计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(5): 1164−1189, 2025

大模型时代的混合专家系统优化综述

史宏志　赵　健　赵雅倩　李茹杨　魏　辉　胡克坤　温东超　金　良
（浪潮电子信息产业股份有限公司　济南　250101）
（浪潮（北京）电子信息产业有限公司　北京　100095）
（shihzh@ieisystem.com）

Survey on System Optimization for Mixture of Experts in the Era of Large Models

Shi Hongzhi, Zhao Jian, Zhao Yaqian, Li Ruyang, Wei Hui, Hu Kekun, Wen Dongchao, and Jin Liang
（*IEIT SYSTEMS Co., Ltd., Jinan* 250101）
（*Inspur* (*Beijing*) *Electronic Information Industry Co., Ltd., Beijing* 100095）

Abstract　In recent years, large models have made unprecedented progresses in variety of domains, such as natural language processing and machine vision. Mixture of experts (MoE) has emerged as one of the most popular architectures for large models due to its distinct advantages in model parameter scalability, computational cost control and complex task processing. However, with the continuous increase of the parameter scale, the execution efficiency and scalability of the system are becoming increasingly challenging to meet the demand, and must be addressed urgently. The system optimization approach is an effective solution to solve this problem, which has become a hot research area. In light of this, we review the present research status of MoE system optimization techniques in the era of large model in this paper. To begin, we describe the present development state of work for MoE large model, and analyze the performance bottlenecks it faces on the system side. Then, we comprehensively sort out and deeply analyze the most recent research progress from four system core dimensions, ranging from memory occupation, communication latency, computational efficiency to parallel scaling, and compare and elaborate on the key technologies, application scenarios and optimization directions; finally, we summarize the current research state of MoE system optimization and outline some future research directions as well.

Key words　large model; mixture of experts; memory offloading; hierarchical communication; expert placement; expert activation prediction; adaptive parallelism

摘　要　近年来，大模型推动自然语言处理、机器视觉等众多领域取得前所未有的进展. 混合专家（mixture of experts, MoE）凭借在模型参数扩展、计算成本控制和复杂任务处理等方面的独特优势成为大模型的主流架构之一. 然而，随着参数规模的持续增长，系统的执行效率和可扩展能力愈发难以满足需求，亟待解决. 系统优化方法是解决这一挑战的有效途径，日益成为研究热点. 故综述大模型时代 MoE 系统优化技术的研究现状，首先介绍 MoE 大模型的发展现状，并分析其在系统端面临的性能瓶颈；然后从内存占用、通信延迟、计算效率和并行扩展 4 个系统核心维度对最新的研究进展进行全面梳理和深入分析，并对其中涉及的关键技术、适用场景和待优化方向进行详细对比阐述；最后总结 MoE 系统优化的研究现状，并展望未来研究方向.

关键词　大模型；混合专家；内存卸载；分层通信；专家放置；专家激活预测；自适应并行

收稿日期：2024-01-12；**修回日期**：2024-09-18
基金项目：山东省自然科学基金项目（ZR2020QF035）
This work was supported by Shandong Provincial Natural Science Foundation (ZR2020QF035).
通信作者：赵健（zhao_jian@ieisystem.com）

中图法分类号 TP391

DOI: 10.7544/issn1000-1239.202440016 CSTR: 32373.14.issn1000-1239.202440016

近年来，深度学习模型在多个应用领域取得了令人印象深刻的成果. 然而，随着深度学习在各领域应用的持续深入，其面临的问题也越来越复杂，人们对具有更高能力模型的期待也愈加迫切.

扩展模型参数规模是增强模型容量和提高模型能力最简单、最有效的方式之一[1]，因此，近年来大模型的参数规模呈现爆炸性增长趋势. 2019 年发布的 T5[2] 仅有 110 亿个参数，但到 2020 年发布的 GPT-3[3] 的参数量就达到了 1 750 亿，2021 年发布的 Gopher[4] 参数量则继续增长到 2 800 亿规模，而 2022 年发布的 PaLM[5] 参数量已经达到惊人的 5 400 亿. 在传统稠密模型，几乎所有参数都参与计算，计算成本随着模型参数规模增长线性增加. 持续增长的模型参数为模型训练和推理的计算开销带来严峻挑战，严重阻碍了模型规模的进一步扩展，亟需新的模型架构解决这一挑战.

混合专家（mixture of experts, MoE）成为突破这一困局的有效方法之一，其概念最早在文献 [6] 中被提出，旨在通过分治思想将复杂问题分解为多个子问题单独求解. MoE 架构将模型划分为多个规模更小的“专家”模型，并通过可学习门控网络将输入样本路由给不同专家组合分别处理. 区别于传统集成学习模型输入样本会训练所有子模块，MoE 的专家模型专注于特定输入样本/任务，充分发挥不同专家的特长. 多个可选专家以及专家稀疏激活设计，使 MoE 在计算量次线性增加的前提下有效提高了模型的容量和性能. MoE 的优势主要体现在 4 个方面：

1）灵活可扩展. 根据问题的复杂度灵活设置专家数量、专家规模，甚至专家架构，MoE 模型规模可以很容易扩展至万亿级别.

2）低计算开销. 专家稀疏激活机制保证了 MoE 模型的计算复杂度随模型规模的扩展而次线性增加，与同等规模的传统稠密模型相比，计算开销显著降低.

3）强专业能力. 门控网络能够准确捕获输入样本的特征差异，进而分配给相应专家处理，通过对特定数据的针对性训练，专家强化对特定领域任务的预测能力，展现出高度的专业性.

4）强泛化性. 路由机制根据输入样本动态激活专家，组合多个专家预测结果，修正专家预测偏差，提高了模型的泛化能力和可靠性.

因具有上述优势，MoE 目前已经广泛应用于计算机视觉[7-10]、多模态[11-13]、语音识别[14-16] 以及推荐搜索系统[17-18] 等众多 AI 细分领域，并取得巨大成功. 特别是在自然语言处理领域[19-25]，MoE 已经成为大模型降低计算成本的有效解决方案之一. 采用 MoE 架构的 GLaM[24] 具有最多 1.2 万亿个参数，虽然参数规模是 GPT-3 的 7 倍，但计算量只有其 1/3，并在 29 个自然语言处理基准任务上取得了超越 GPT-3 的表现. 其他大模型如 Switch Transformers[19]，M6[26]，M6-10T[27]，PANGU-Σ[28] 等亦采用了 MoE 架构并取得令人瞩目的成果. 基于 MoE 架构的 Mixtral-8x7B[29] 在开源大模型领域取得领先性能，特别是在数学、代码生成和多语言基准测试中表现突出. 这一显著成就不仅彰显了 MoE 架构的巨大潜力，也为 MoE 的发展前景带来更多期待和启示.

虽然 MoE 在众多应用领域取得了显著成功，但仍然存在内存占用大、通信延迟高、计算效率低和扩展困难的关键性能问题，给 MoE 模型训练和推理的高可扩展性和高效性带来严峻挑战，严重阻碍了 MoE 模型的进一步发展. 对 MoE 系统的优化研究已经如火如荼地开展，特别是在近 2 年，涌现出大量杰出工作. 然而，目前尚缺乏对该领域研究进展进行全面回顾的文献. 据统计，目前仅有少数几篇专门介绍 MoE 的综述文献. 其中，文献 [30−32] 概述了 2017 年之前的 MoE 早期发展，时间跨度较大，其内容已不能反映当前的研究前沿. 文献 [33] 系统介绍了 MoE 模型近年来的整体发展情况，特别是与深度学习模型结合后取得的长足进展，但其并未特别关注 MoE 系统优化研究. 文献 [34] 则对计算机视觉任务领域的 MoE 模型路由门控研究进展进行了系统论述. 在 2024 年文献 [35] 对 MoE 模型的研究进展进行了全面梳理，但其重点关注算法方向. 据笔者所知，本文是系统介绍大模型时代背景下 MoE 系统优化研究的综述性文章.

本文全面回顾了 MoE 系统优化技术的研究进展，并从内存、通信、计算以及并行 4 个系统核心方向对这些工作分别展开介绍，如图 1 所示. 本文讨论了这些 MoE 系统优化技术的核心设计思想、关键技术特征以及优势和局限性. 本文能够帮助该领域的研究人员快速全面追踪这一领域的研究进展，激发、推动该方向的研究创新.

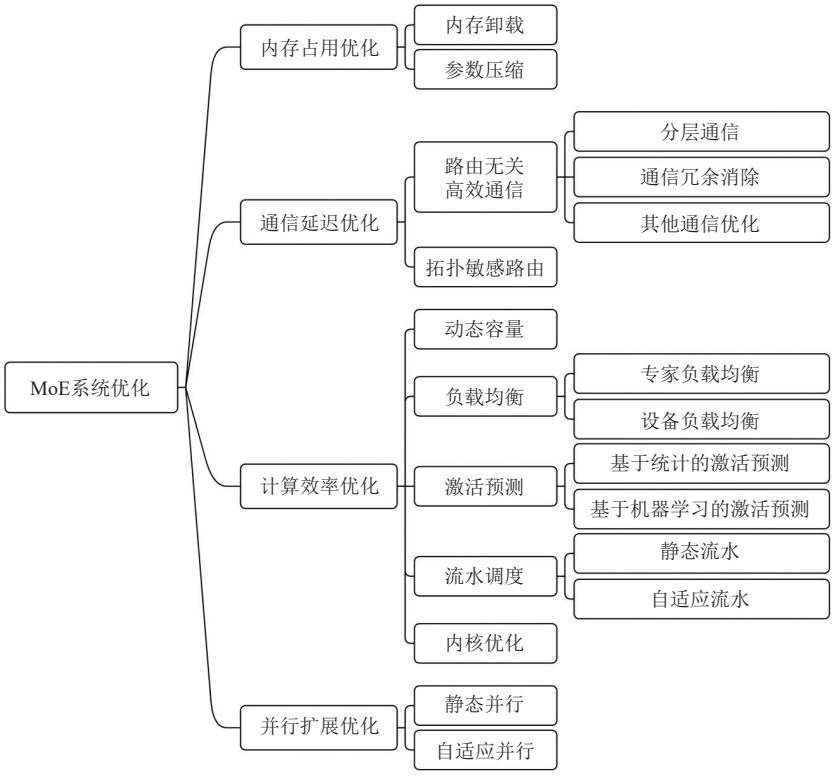


Fig. 1 Taxonomy of MoE systems optimization techniques

图 1 MoE 系统优化技术分类

1 MoE 概述

MoE 本质上是一种集成学习技术, 使用多个子模块分别处理原始问题的不同子集, 这些子模块被称为“专家”. MoE 并非一个新概念, 早在 20 世纪 90 年代便被提出, 并应用于机器学习领域. 对大模型以及多任务模型研究的持续高涨引发了 MoE 研究的新一轮热潮. 下面将概述 MoE 的核心概念、MoE 大模型的发展及其在系统端面临的问题与挑战.

1.1 MoE 核心概念

本节将介绍 MoE 涉及到的 4 个核心概念, 包括 MoE 的模型结构、专家容量、专家并行及 MoE 与集成学习.

1)模型结构. 文献 [20] 提出了使用稀疏激活专家处理输入样本的稀疏门控 MoE 模型, 在扩展模型参数规模的同时控制了计算复杂度, 奠定了后续 MoE 模型的基本结构. 如图 2 所示, MoE 模型由一个门控网络和多个并行的子模块组成. 门控网络根据输入样本的特征将其路由到特定专家, 并为专家预测结果分配权重. MoE 模型的最终预测结果由各专家预测输出加权获得. MoE 模型的输出如式(1)所示:

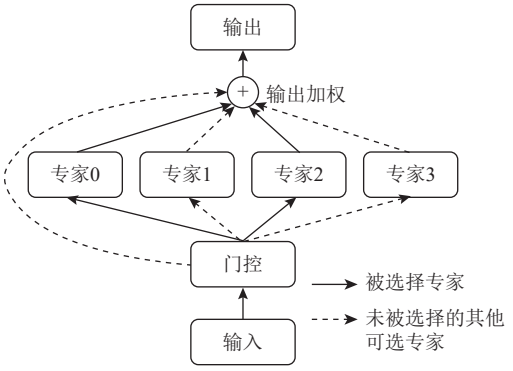


Fig. 2 The architecture of MoE model

图 2 MoE 模型架构

$$y=\sum_{i=1}^{n}G(x)E_i(x), \tag{1}$$

其中 $E_i(x)$ 表示专家 i 对输入 x 的输出, $G(x)$ 表示门控网络对输入 x 的输出.

当使用 Top-k[20] 门控网络时, 门控网络从 n 个专家中选择 k 个专家处理输入样本. *Softmax* 函数被广泛用于门控网络, 门控网络的输出如式(2)所示:

$$G(x)=softmax(Topk(xW_{\text{gate}},k)), \tag{2}$$

其中 W_{gate} 为门控网络参数. 使用 $Topk$ 操作保留门控输出的前 k 个值, 并设置其余专家输出为负无穷, 从而通过 *Softmax* 只评估前 k 个专家.

2)专家容量. 为满足通信和计算的系统级约束，GShard[36]提出了专家容量(expert capacity, EC)概念，指定专家处理的样本数量，其定义如式(3)所示：

$$EC = k \times f \times \frac{T}{E}. \tag{3}$$

其中 k 表示使用 Top-k 路由策略为每个样本分配的专家数量，$k \geq 1$；T 表示一个批次数据中的样本数量；E 表示全局专家数量；f 表示容量因子，是一个可调超参数. $f > 1$，表示专家容量具有余度，为热门专家处理更多样本提供空间. f 值越大，样本路由越不均衡，通信和计算浪费越严重. 当专家分配的输入样本超过专家容量时，则会丢弃多余的样本；反之，当专家分配的样本数量未达到专家容量时，则会用“0”数据填充剩余容量空间. 专家容量设计被后续 MoE 模型广泛采用，成为一种典型配置.

3)专家并行. 专家并行概念在 GShard[36]中被提出，其将不同专家放置在多个设备上并行执行，而非专家层则以数据并行方式执行. 由于专家位于不同设备上，需要执行 All-to-All 通信将输入样本分发给相应专家，并在专家处理后将专家输出恢复到样本原始位置. 专家并行策略成为 MoE 模型的典型并行模式，被后续众多工作采用.

专家并行的计算流程如图 3 所示. 其中，MoE 层集成了 3 个专家，采用 Top-1 门控机制且专家容量设置为 1，专家通过专家并行分布在 3 个 GPU 设备上并行执行.

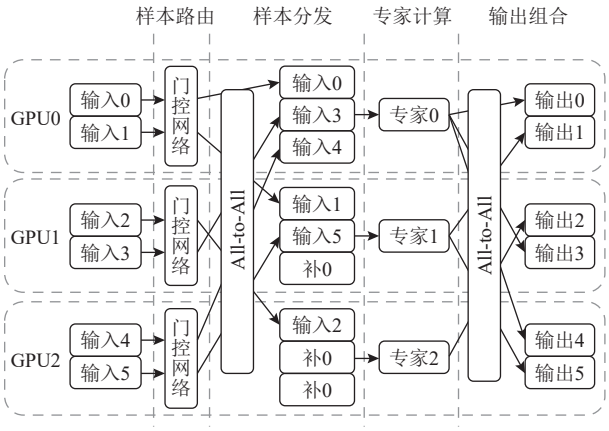


Fig. 3 An example of expert parallel execution flow of MoE

图 3 MoE 专家并行执行流程示例

采用专家并行的 MoE 模型计算流程如下：

①样本路由. 门控网络根据路由策略生成样本—专家映射以及每一对映射的置信概率. 常用的路由策略如 Top-k.

②样本分发. 由样本—专家映射生成专家—样本映射，通过 All-to-All 通信对样本重新排列，将分配给同一专家的样本放置到连续的内存空间. 专家容量溢出则丢弃多余数据，专家容量有剩余则用“0”填充.

③专家计算. 对分配到的样本执行专家计算.

④输出组合. 通过 All-to-All 通信重塑专家计算结果以满足原始输入样本排序，专家计算结果经置信概率加权调整后得到最终输出结果.

专家稀疏激活是 MoE 最核心的特征. 一个可训练的门控网络根据输入样本特征，动态地将样本分配给不同专家.

4)MOE 与集成学习. MoE 作为集成学习的一个分支，通过专业化的专家网络和动态门控机制，为处理复杂和多样化任务提供了一种高效策略. 而传统集成学习方法则提供了一种更广泛和通用的途径，适用于多种机器学习任务. 尽管两者都致力于提升模型精度和泛化能力，但其在实现机制和应用场景上各有侧重，具体差异和联系如表 1 所示.

Table 1 Comparison of MoE and Ensemble Learning

表 1 MoE 与集成学习对比

类别	MoE	集成学习
模型集成	都涉及整合多个模型结果以提高预测精度，都利用不同模型的优势来解决复杂问题.	
应用目的	不仅提高模型预测精度和泛化能力，还提高模型收敛速度和执行效率.	提高模型预测精度和泛化能力.
模型结构	由多个专家和门控网络组成，每个专家负责处理模型的一部分子任务，门控网络决定专家和子任务的映射关系.	通常由多个独立训练的相同或不同算法模型构成.
任务分解	将复杂任务分解为相对简单的子任务，专家网络专注于处理特定子任务.	通常不涉及任务分解，多个学习器同时处理整个任务，通过投票、加权或平均等方法整合结果.
训练方法	门控网络学习将任务分配给相应专家网络，专家网络专注于特定任务学习.	各学习器独立训练，可使用不同的数据或算法，不涉及可学习的任务分配网络.
应用场景	适用于复杂且任务可以分解为不同子任务的场景.	适用于提升模型泛化能力和鲁棒性，尤其在数据或者特征有限的情况.
稀疏性	只有少量专家在给定的时间被激活，提高计算效率.	通常不具有稀疏性，所有模型都参与最终的预测.
动态性	根据输入数据动态激活最适合的专家网络.	通常不具有动态性，模型的集成是静态的，且在训练过程就已确定.

MoE 凭借专业化的专家网络集成和动态门控机制设计，不仅可以显著提升模型的精度和泛化能力，还可以极大提高模型训练和推理的效率，展现出巨大潜力.MoE 是集成学习理论的一次重要创新，不仅丰富了集成学习的技术优势，而且极大扩展了集成学习的应用领域，特别是在自然语言处理领域，MoE 已经成为大模型的主流架构之一.

1.2 MoE 大模型

MoE 模型的专家非线性特性与自然语言处理领域潜在的问题聚类结构完美契合，赋予 MoE 模型在处理复杂语言任务时的独特优势，并在提高模型容量、降低计算成本等方面展现出巨大优势. 这使得 MoE 大模型成为自然语言处理领域一个备受瞩目且具有广泛应用前景的解决方案. 随着研究的深入，各种 MoE 大模型相继问世，不断刷新各领域的性能记录，推动大模型技术的发展.

MoE 大模型发展初期，高可扩展性和低计算开销优势是其被应用于大模型的主要初衷，而这得益于文献 [20] 创造性提出的稀疏门控 MoE 模型. 其通过稀疏激活专家设计，在扩展模型参数规模的同时显著降低了计算成本，并在自然语言处理领域的多项任务上取得了明显的性能提升. 文献 [20] 将 MoE 与 LSTM 模型相结合，随着自然语言模型逐渐发展到以 Transformer 架构为基础，GShard[36] 进一步将 MoE 引入其中，利用专家网络替代注意力中的前馈层，开启了 MoE 和 Transformer 模型结合的大门，如图 4 所示. 该模型参数量达 6 000 亿，并以更低的计算开销、更短的训练时间实现比以往稠密模型更优的翻译质量. 此外，Switch Transformers[19] 构建了高达万亿参数的 MoE 大模型，在多种自然语言处理任务上，相比 T5 基线模型，无论是训练速度、推理速度还是模型精度都取得显著提升，展现出 MoE 在处理多语言复杂任务方面的专长. 此外，其他一些工作[24,37-38] 也进一步挖掘了 MoE 大模型的低计算开销和高可扩展优势在自然语言处理领域的潜力.

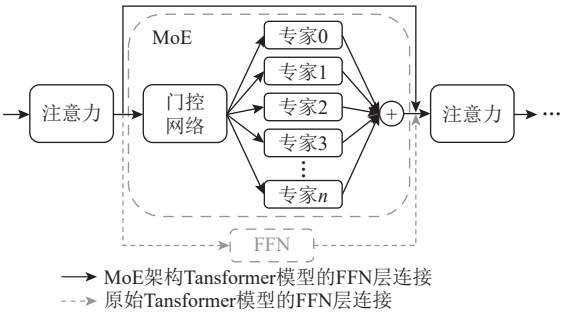


Fig. 4 MoE-architectured Transformer model

图 4 MoE 架构 Transformer 模型

随着 MoE 大模型的发展，MoE 的强专业能力优势也被继续挖掘，以进一步提高模型精度. ST-MoE[21] 利用编码器中存在的词典级别的专家专业化优化模型表现. 进一步地，Mixtral-8x7B[29] 探索了专家选择时的结构化语法行为，实现在所有评估的基准测试中表现优于或匹配稠密的 Llama2-70B 和 GPT-3.5. 特别是在数学、代码生成、多语言和偏见等特定领域展现出显著优势. 为了充分发挥 MoE 大模型专家的专业化能力，LLaMA-MoE[39] 通过神经元独立和神经元共享的专家划分以及持续训练方法，提高模型的表示能力和灵活性，并为从现有稠密大模型转换为 MoE 大模型提供了一个高性能、低成本的可行框架. 一种更为极端的专家专业化方法是 DeepSeekMoE[40] 提出的细粒度专家分割和共享专家隔离技术，旨在解决知识混合和专家冗余问题，以提高模型精度和计算效率. 更为全面、深入的专家专业化研究工作是 OpenMoE[41]，其对不同层次的专家专业化进行了深入分析，发现相同 ID 的样本，无论上下文如何，总是被路由到相同专家，且这种专家选择偏好在训练早期就已形成. MoE 大模型的专业化利用正向着细粒度方向持续发展.

近 1 年来，MoE 大模型领域经历了井喷式发展，众多 AI 厂商和研究机构纷纷推出自研 MoE 大模型. xAI 的 Grok-1[42]、Google 的 Gemini 1.5 Pro[43]、Snowflake 的 Arctic[44] 以及 Databricks 的 DBRX[45] 等 MoE 大模型在多个领域取得了令人惊喜的成绩.

MoE 大模型的不断涌现不仅推动了大模型的快速发展，也加速了生成式人工智能(AIGC)领域的研究进程. 然而，专家稀疏激活特性是把双刃剑，既有益于降低模型计算开销、提升模型专业能力，又对模型的高可扩展和高效性带来极大挑战.

1.3 问题与挑战

由于模型参数规模较大，MoE 大模型分布式训练和推理成为一种必然选择. 这意味着需要低效的 All-to-All 同步通信实现输入样本和专家输出在不同设备间的分发和收集. 此外，不同专家对样本的喜好差异也带来了专家和设备负载不均衡问题. 这些问题给 MoE 大模型系统端的高可扩展和高效性提出了严峻挑战.

1)内存占用大，硬件需求较高. 一方面，MoE 大模型的巨大内存需求与 GPU 的内存容量差距明显，且 GPU 内存的增长速度远落后于 MoE 大模型参数规模的增长速度，加剧了对 GPU 海量内存资源的需求以及降低内存占用技术的需求. 另一方面，虽然 MoE 大模型以计算量次线性增加的代价扩展模型参数规模，但模型精度往往并不比具有同等计算量的对应稠密模型更高，MoE 大模型参数效率较低，这进一步加剧了对 GPU 内存需求的压力. MoE 大模型内存占用较大对训练和推理的硬件资源提出较高要求.

2)通信延迟高，模型执行低效. 首先，MoE 大模

型分布式训练和推理中所有专家都需要从其他设备获取输入样本，这涉及节点间的同步 All-to-All 通信交换数据，延迟较高. 其次，考虑到异质的通信链路，这种同步通信在带宽受限的节点间以及路径较长的通信链路将产生更高延迟，滞后专家计算执行. 最后，样本分发与通信带宽不匹配、All-to-All 与 All-Reduce 等通信并发执行竞争通信带宽、通信中存在冗余数据进一步加剧了 All-to-All 通信延迟.MoE 大模型通信延迟较高，极大地降低了模型执行效率.

3)计算效率低，模型执行耗时. 一方面，通用的固定专家容量策略会引发专家补 0 问题，导致计算资源浪费. 另一方面，专家的稀疏激活特性决定专家负载不均衡，进而导致热门/冷门专家驻留设备负载过高或过低. 更为严重的是，这种负载不均衡动态变化，严重影响计算资源利用率和模型计算效率. 此外，专家选择、样本通信和专家执行同步执行，导致专家计算执行滞后，模型执行低效.MoE 大模型计算低效直接增大了模型执行延迟.

4)并行扩展难，动态负载性能差. 首先，MoE 大模型参数规模的覆盖范围广泛，通过分布式训练和推理，需要确保并行策略具有良好的可扩展性，以适应不同规模的模型和集群资源. 其次，专家稀疏激活特性决定一种并行策略难以为 MoE 大模型的动态专家负载提供最优性能，并行策略需根据专家负载动态调整，以保证高性能表现. MoE 大模型并行扩展困难及动态负载性能差，严重影响模型执行效率.

上述 MoE 大模型在训练和推理过程中面临的系统端挑战严重阻碍了其进一步发展. 近年来，MoE 系统优化技术受到研究学者的广泛关注并取得重要进展. 内存占用、通信延迟、计算效率以及并行扩展优化是解决 MoE 系统面临的高可扩展和高效性挑战的 4 个核心研究方向，其对应关系如图 5 所示，本文后续章节也将分别从这 4 个研究方向综述相关系统优化技术的进展.

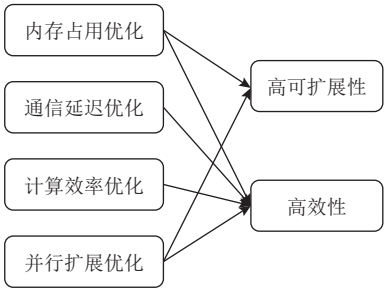


Fig. 5 Corresponding relationship between core research directions and challenges

图 5 核心研究方向与挑战对应关系

特别指出，鉴于目前广泛认知的 MoE 模型主要就是指 MoE 大模型，因此，为简化叙述并保持一致性，本文后续将不再对“MoE 模型”和“MoE 大模型”进行区分，而统一使用“MoE 模型”这一术语.

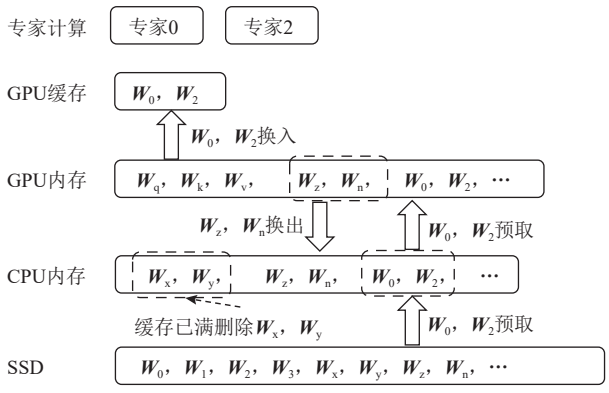
2 MoE 内存占用优化

MoE 模型的大规模参数对加速器内存容量提出了严峻挑战. 在过去的 3 年里(2021—2023 年)，高端 GPU 的最大内存容量仍保持在 80 GB[46-47]，而 2021 年发布的 Switch Transformers[19] 参数量就已达到了 1.6 万亿. 动辄需要数百甚至数千张最高配置高端加速器才能实现 MoE 模型高效运行的极高需求令人望而却步[36]. 现有文献通常通过以下方式实现 MoE 模型的内存占用优化：内存卸载和参数压缩.

2.1 内存卸载

内存卸载[48-52] 利用 CPU 内存和 SSD 存储扩展 GPU 有限的内存容量，同时将非活跃张量卸载到“扩展内存”，并在需要时提前预取到 GPU 内存. 基于 CPU 内存和 SSD 存储在容量和价格方面的显著优势，以及 GPU 和 CPU 通信、CPU 和 SSD 通信的通信带宽能够支撑计算和张量传输的并行处理，内存卸载已经成为 MoE 模型突破 GPU 内存限制的有效方法.

图 6 展示了典型 MoE 模型参数卸载系统的示意图. 其中，GPU 内存、CPU 内存和 SSD 存储构成一个 3 级存储系统. CPU 内存作为 SSD 缓存，通过预取技术将一些热门专家的参数预取到 CPU 内存. 此外，当前计算所需的参数(专家 0 和专家 2)也通过预取技术预先换入到 GPU 内存.



注：$W_{0\text{-}3}$为专家参数，其他为注意力等参数.

Fig. 6 Illustration of parameter offloading in MoE models

图 6 MoE 模型参数卸载示意图

专家参数在 MoE 模型执行过程的内存占用中占据主导地位. 因此，专家参数卸载成为 MoE 模型内存

卸载的核心研究方向. 此外, 针对模型执行功能过程中临时变量的内存占用优化以及利用分离式内存技术扩展加速器内存方法也广泛开展.

1)专家卸载. 推理场景下, 专家激活呈现出显著的时间局部性, 这意味着被当前批次数据激活的专家在随后的几个批次中很可能会再次被激活. 只将这些热门专家保留在内存中, 将极大缓解加速器的内存压力. 最简单的专家卸载方法是将专家分为热门专家和冷门专家并分别存储在 GPU 内存和卸载到 CPU 内存中[53-54]. 与文献 [53] 通过门控网络确定冷门专家和热门专家不同, PC-MoE[55] 根据输入样本、门控网络和专家参数幅值的大小估计专家的重要性, 将一些重要专家和非重要专家分别存储在 GPU 内存和卸载到 CPU 内存中, 并仅使用一些重要专家子集进行推理. 对于 GPU 内存中缓存专家的维护策略, 文献 [53] 采用了懒惰的后进先出专家驱逐策略为新专家提供所需内存空间, 文献 [54] 则采用 LRU 缓存策略卸载专家. 而 PC-MoE 则通过定期更新重要专家子集保持使用中的最重要专家, 同时重要专家以层的方式周期地、异步地交换到 GPU 内存中以平摊专家交换以及通信开销. 对于冷门专家的处理两者也截然不同. 当所需专家没有在 GPU 内存缓存时, 文献 [53] 采用将冷门专家从 CPU 拷贝到 GPU 的标准方法, 并将专家传输与输入样本传输并行处理, 以隐藏专家传输延迟. 但这不可避免地会与样本竞争通信带宽, 导致通信延迟.PC-MoE 则通过跳过或与重要专家交换的方式以最小的精度损失执行非重要专家的计算请求, 但该方法本质上是另一种形式的模型剪枝, 不可避免地会牺牲模型精度.

非专家参数与输入无关且总是会被使用, 而专家参数尽管占据 MoE 模型参数量的绝大部分, 但在推理过程中只有少部分专家被激活. 基于此, Pre-gated MoE[56] 将非专家参数和专家参数分别存储在 GPU 内存和卸载到 CPU 内存. 而 SE-MoE[57] 则进一步将卸载对象扩展至 SSD, 其为减少从慢速 SSD 迁移专家导致的延迟, 设计了 2 维预取调度方法以重叠专家计算和专家参数加载, 并在 CPU 内存为 SSD 建立类似 LFU 的缓存机制以减少计算等待时间. 该方法利用 NVLink 和 PCIe 这 2 个维度的带宽, 同时预取稠密和稀疏参数, 减少异构存储带来的调度间隙以提高模型执行效率. Janus[58] 同样利用专家预取策略支持细粒度的专家获取请求. 其在自注意力计算过程中利用节点间和节点内带宽空闲时段预先获取所有外部专家和内部专家分别存储到本地的 CPU 和 GPU 内存. 该方法不仅避免了模型计算等待专家参数获取的延迟, 同时还可以平滑突发通信并提高带宽利用率. 而 Pre-gated MoE 并不需要专门的专家预取策略, 这得益于其预先门控设计完全解耦了本层门控专家选择和专家执行的顺序依赖关系, 允许本层专家执行和下一层激活专家加载并发执行. 同样不需要专家预取的还有文献 [59] 所描述的方法, 其通过单个专家粒度的精细调度使专家参数加载、专家计算和专家参数优化流水并行执行以隐藏专家参数加载延迟.

与 Pre-gated MoE 利用前一 MoE 层输出激活当前层专家类似的工作还有文献 [54] 中的方法和 EdgeMoE[60]. 两者区别在于, 文献 [54] 利用当前层门控, 直接以前一 MoE 层输出作为输入预测激活专家, 门控不需要微调, 而 EdgeMoE 则采用离线阶段构建的统计模型作为门控. 另一种基于统计的专家预取方法由 MoE-Infinity[61] 提出. 其通过专家激活矩阵记录推理过程中每个生成序列的激活专家选择, 并利用聚类方法选择具有代表性的专家激活矩阵集合. 在推理过程中, 根据当前的专家激活矩阵及其集合预测激活专家, 实现专家预取. 然而, 该方法引入了额外的存储开销, 一定程度上削弱了专家卸载的内存优化收益.

Fiddler[62] 采取了一个独特的研究视角, 探索利用 CPU 执行卸载到 CPU 的专家计算, 以减少 CPU 和 GPU 间的专家参数移动. 但专家计算通常是计算受限的, GPU 仍需保留足够多的专家参数才能最大化两者的并行处理速度, 显然其更适合模型规模较小或输入数据批处理量较小的单 GPU 推理场景, 应用范围存在局限性.

2)激活值和临时缓存内存占用优化. 除专家参数外, 激活值和临时缓存也在 MoE 模型执行过程中占用大量内存, 研究人员对此问题也开展了广泛研究. MPipeMoE[63] 通过沿着输入序列维度切分张量, 且为张量的不同切分块共享相同缓冲区以减少激活值和临时变量内存占用. 其同时设计了考虑包括计算、通信、内存拷贝速率等硬件能力和包括通信、内存拷贝速度衰减因子等模型特性的内存重用策略性能模型, 在运行时评估对中间激活值和分发样本分别应用重计算[64-67]/卸载和通信/卸载的 4 种内存重用策略, 配置最优的内存优化方法. 为解决数据并行中因模型参数在数据并行组内重复拷贝导致的内存浪费问题, 文献 [68] 利用 ZeRO-Offload[48] 卸载优化器和梯度状态参数到 CPU 内存以突破 GPU 内存限制, 扩

展模型规模. DeepSpeed-TED[69] 则提出了一个分块版本的优化器, 以固定大小的块处理专家模型梯度参数, 而不是一次性处理所有专家模型梯度参数, 并通过参数块重用降低 GPU 峰值内存占用.

3)利用分离式内存卸载. 1)和 2)中讨论涉及的卸载设备都局限在统一内存, 然而, 文献 [70] 表明, 利用分离式内存(disaggregated memory)扩展 GPU 内存来卸载 MoE 模型参数是解决加速器内存限制的一种有效方法. 其提出 4 种利用分离式内存扩展 GPU 内存的方法, 不同程度地卸载 MoE 模型参数. 实验结果揭示, 当内存带宽需求较低时, 通过为每个 GPU 分别扩展内存的架构能够提供与高端集成芯片系统相当的性能, 为后续通过分离式内存扩展 GPU 内存系统实现 MoE 模型高效执行奠定了基础.

MoE 内存卸载技术的关键是预取方法的效率, 这涉及对预取数据以及预取时机的考量. 上述研究工作虽然已经显著改善了 MoE 模型的内存占用过大问题, 但在完全隐藏专家或数据传输延迟方面尚存在不足, 主要原因在于专家缓存命中率还不够准确, 不能将计算所需的专家参数全部提前预取到 GPU 内存. 此外, 目前内存卸载方法难免会影响模型的处理速度, 对于性能极致需求的场景其应用也受到限制.

探索激活专家对序列样本的时间局部性、不同层专家对相同样本的相关性以及在特定任务和场景下的专家激活模式, 并通过神经网络模型预测激活专家, 以实现专家预取是内存卸载技术中一个有价值的研究方向. 此外, 结合参数共享等方法, 不依赖专家预取, 而是利用本地存储的低秩矩阵和其他专家来恢复目标专家参数, 也是值得探索的方向.

2.2 参数压缩

参数压缩是降低模型内存占用的另一种流行方法. 通过专家网络扩展模型参数并不能总是有效提升模型容量, MoE 模型精度往往并不能优于具有同等参数量的相应稠密模型, 参数存在冗余. 传统稠密模型的知识蒸馏[71]、剪枝[72-73]、量化[74-75] 和参数共享[76-77] 等参数压缩技术同样可以解决 MoE 模型的参数低效问题.

1)知识蒸馏. 通过知识蒸馏从 MoE 模型中提取较小的模型是压缩 MoE 模型冗余参数的一种直接方法. DeepSpeed-MoE[78] 设计了混合同学方法, 利用阶段性知识蒸馏压缩残差金字塔 MoE 模型. 除专家网络的深度外, 还可以进一步利用知识蒸馏方法压缩专家数量, 如 SE-MoE[57].

2)剪枝. 通常 MoE 层只会稀疏激活数量有限的几个专家生成输出, 因此利用剪枝移除使用率极低的专家可以降低模型参数, 而不会或只轻微降低模型精度. 文献 [68] 利用剪枝方法基于随机选择和专家 Top-k 利用率移除模型中的不重要专家. 类似的工作还有针对多语种机器翻译的 MoE 模型 NLLB-200[79], 其根据多种统计粒度评估专家重要性, 并结合多种剪枝策略, 实现了高达 80% 的专家剪枝率. 文献 [80] 介绍了一种任务无关和特定于任务的训练后专家剪枝方法, 然而, 该方法基于专家组合的枚举搜索, 时间复杂度过高, 不适合应用于专家数量较多的模型. 类似地, 还可以通过任务级路由从 MoE 模型中提取小的子网络, 利用任务标识将属于特定任务的所有数据路由到同一类专家[23]. 此外, SEER-MoE[81] 提出了基于重计数指导的剪枝方法, 并利用正则化的微调来恢复剪枝过程中的精度损失. 上述方法通过在微调阶段或微调后的验证数据集上统计每个专家接收样本的总数来确定非重要专家, 属于经验性方法. 而文献 [82] 则在理论上证明了优先剪除在微调阶段路由门控参数 L2 范数变化较小的专家的合理性. 庞大的搜索空间导致确定适合的启发式剪枝方法困难, 依赖梯度的剪枝方法开销巨大. 为应对这个挑战, EEP[83] 利用进化策略搜索路由映射矩阵以保留最重要专家, 并通过专家聚合矩阵检索专家知识并将其合并到保留的专家.

3)量化. 量化方法通过减少张量表示所需的位宽来减少模型的存储开销, 已经在传统神经网络模型领域被广泛讨论, 同样该方法也被应用到 MoE 模型. 如 MoQE[84] 采用 2b 量化专家网络参数以减少内存占用, 而 QMoE[85] 则采用自定义格式进一步将万亿参数的 MoE 模型压缩到每个参数占用不到 1b. 此外, 还有一些量化工作探索在更高的位宽上量化 MoE 模型[60,86-87], 以及动态确定量化专家的数量以在资源受限环境实现自适应服务[88]. 然而, 与针对传统稠密神经网络模型的量化方法相比, 针对 MoE 模型的量化方法并未展现出独特性. 例如, 同时面向 MoE 模型和传统稠密模型, 仅利用预训练模型参数的量化方法 FineQuant[87].

4)参数共享. 通过利用低秩矩阵近似大矩阵的矩阵分解或其他技术共享不同专家的参数也是压缩 MoE 模型参数的一种有效方法. WideNet[10] 提出跨 MoE 层共享专家以减少专家参数, 实现参数效率和模型效果的共同提升. 随后的 MPoE[89] 并没有简单地共享现有的专家, 而是通过结构化矩阵分解方法共享专家参数. 具体而言, 该方法设计了矩阵乘积算子分解

专家参数矩阵为专家共享的全局中心张量以及特定于专家的辅助张量. 由于中心张量有效重组、聚集原始参数矩阵的重要信息, 包含了大部分矩阵乘积算子分解参数, 显著降低了模型参数量. 注意到 MoE 模型推理的计算代价随激活专家数量线性增加. 因此, 将激活专家合并成一个专家, 然后再执行专家计算可以极大提高计算效率. 基于此, 文献 [90] 提出按照门控网络激活值合并多个激活专家的加权模型融合方法, 但该方法显然只对门控网络专家激活数量大于 1 的模型才有效. 此外, 最近的工作[91] 通过对输入特征调制来共享 1 个专家, 其部分专家功能由调制单元代替实现, 是传统 MoE 架构的一种变体.

为进一步压缩冗余参数, 一些工作还尝试直接将稀疏 MoE 模型压缩为稠密模型. 文献 [92] 根据下游任务一边训练一边逐步剪枝非专业专家, 不需要微调即可将 MoE 模型简化为单专家密集模型. 但这类任务相关的提炼模型方法一般只适用于多语言/多任务等特定 MoE 模型. Switch Transformers[19] 则通过知识蒸馏来实现 MoE 模型稠密化. 沿着这个方向, 文献 [93] 设计了一个包括知识收集和知识提炼的通用训练框架蒸馏稠密的学生模型, 并提出了基于 Top-k 和奇异值分解的知识收集方法.MoE 模型的稠密化丧失了多专家的专业性, 通常只能在特定的子任务部署时发挥良好作用.

参数压缩方法可以有效降低模型的参数量, 提高模型的执行效率, 但不可避免地会对模型精度产生负面影响. 因此, 这类方法通常都需要额外的微调过程, 以弥补模型的精度损失. 然而, MoE 模型训练成本高昂, 这些巨大开销限制了此类方法的应用场景. 但对于一些内存受限的加速器部署环境, 其依然提供了一个相对有效的解决办法.

3 MoE 通信延迟优化

专家并行的 MoE 模型单次前向传播需执行 2 次 All-to-All 通信以交换数据, 其通信延迟随通信节点数线性增长, 而 MoE 模型依赖大量加速器高效并行执行, 加剧了 All-to-All 通信延迟问题. All-to-All 通信是 MoE 模型执行最耗时的步骤之一, 已经成为性能瓶颈[58,94]. 当前对此问题的研究集中在路由无关通信优化和拓扑敏感路由 2 个方向.

3.1 路由无关通信优化

路由无关通信优化在保持路由策略不变的前提下, 专注于通信方法的改进以提高 All-to-All 通信性能. 一般来说, 该技术通过分层通信、通信冗余消除以及诸如小批量数据频繁通信优化等其他优化实现.

3.1.1 分层通信

MoE 模型通过专家并行执行时, All-to-All 通信存在因通信数据量较小而导致带宽利用率不充分、跨多节点通信路径过长引发通信延迟增加, 以及由于非连续内存访问造成额外延迟等问题. 一种最直接的优化方法是利用异质拓扑网络的通信带宽差异, 先在高带宽拓扑层聚合局部信息, 然后在低带宽拓扑层再统一交换全局数据, 以降低通信跳数, 提高通信效率, 这被称为 All-to-All 分层通信. 表 2 总结了本文经常使用的符号.

Table 2 Description of Symbols

表 2 符号描述

符号	描述
G	每个节点内 GPU 的数量
N	节点总数
P	GPU 设备总数 (P=GN)

All-to-All 分层通信的一种典型实现方法是利用节点内的一个 GPU 收集节点内所有 GPU 的通信数据, 进行数据布局转换后再由该节点启动 All-to-All 通信, 并在 All-to-All 通信完成后再通过相应的数据布局转换和 scatter 通信分发数据给相应专家, 如 HetuMoE[95]. 该方法将节点间数据量扩大为原始通信实现的 G^2 倍, 但将节点内的所有数据分配给一个加速器统一进行通信, 可能导致这个加速器负载过高.

上述方法侧重于减少节点间的通信轮次, 以充分利用节点内和节点间的通信带宽. 另一类方法则着眼于降低通信跳数, 避免不同 rank GPU 跨节点通信导致的不必要开销. DeepSpeed-MoE[78] 先进行 1 次数据布局转换以及节点内 GPU 的 All-to-All 通信, 再进行 1 次相同的数据布局转换以及节点间相同 rank GPU 的 All-to-All 通信, 如图 7 所示. 虽然通信量变为原始方法的 2 倍, 但通信跳数从原始的 $O(P)$ 降低为 $O(G+P/G)$. 此外, Tutel[96] 和 SE-MoE[57] 也实现了类似的分层通信机制. 这种设计对受限于延迟而非带宽的小批量数据通信扩展性更好.

不同于上述从通信方法角度着手的分层通信优化方法, SMILE[97] 进一步细化了单步路由机制, 将其拆分为节点间-节点内 2 级路由架构. 数据首先被节点间门控路由到目标节点, 然后再通过节点内门控分发到该节点内的特定 GPU 设备, 将 All-to-All 通信的复杂度从 $O(GN)$ 降低为 $O(G+N)$. 该方法充分利用

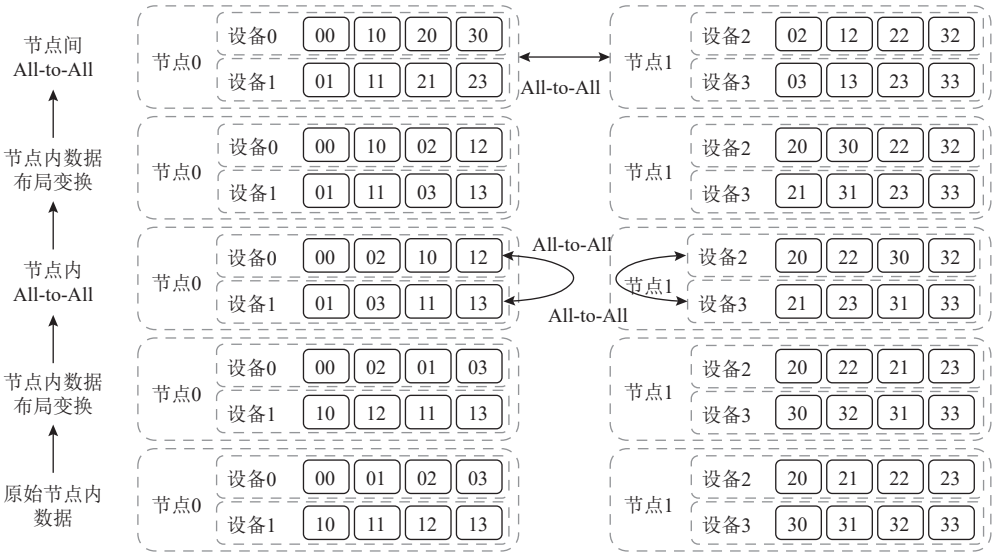


Fig. 7 A typical hierarchical All-to-All communication approach

图 7 一种典型的分层 All-to-All 通信方法

了异质网络拓扑的带宽差异提高通信效率，降低节点间的通信带宽竞争.

与 HetuMoE 类似，Janus[58] 为减少低带宽节点通信数据量也精心设计了分层通信方法，由于其通信对象是专家参数，因此特别设计了高效的专家缓存机制. 具体而言，在 CPU 上缓存从其他节点获取的专家并共享给本地其他设备，同时收集、合并同一节点不同设备的相同外部专家请求以减少跨节点通信量. 在接收到专家请求后按照本地设备、CPU 缓冲区和其他节点的优先顺序请求所需专家.

3.1.2 通信冗余消除

All-to-All 通信的冗余数据主要存在于 2 个方面：首先，一些样本的处理结果对最终预测结果影响微乎其微，删除这些不重要样本可以降低通信开销；其次，张量并行组内设备的激活值经聚合后完全相同，全部参与通信会导致通信数据冗余. 消除 All-to-All 通信中的冗余数据将有效降低通信延迟.

通过稀疏化技术在通信前删除低贡献样本来压缩数据是提高通信效率的有效手段. Sparse-MLP[25] 根据数据与专家间的重要性得分，在路由前排除 10% 具有最低得分的数据. 类似地，V-MoE[7] 的数据块优先路由机制则通过与 Sparse-MLP 重要性得分类似的优先级得分丢弃无用数据块. 但上述的稀疏化方法通常会损害模型精度.

模型通过张量并行执行时，张量并行组设备需要聚合局部信息以获取完整激活值，作为后续算子的输入. 然而，对于 MoE 模型，聚合完整的激活值将导致后续的 All-to-All 通信数据存在冗余. 如图 8 所

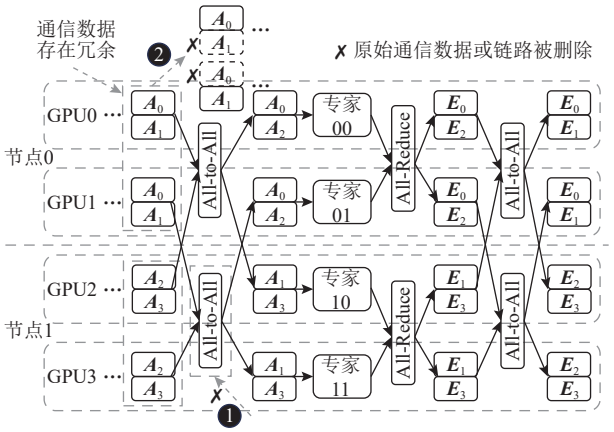


Fig. 8 Elimination of redundant communication data in tensor parallel group

图 8 张量并行组冗余通信数据消除

示，专家 0 和专家 1 通过张量并行方式分别部署在 GPU0 和 GPU1 以及 GPU2 和 GPU3 上. 专家 0 的张量并行组设备 GPU0 和 GPU1 的输入数据都是 $\boldsymbol{A}_0$ 和 $\boldsymbol{A}_1$，当这 2 个 GPU 全部参与通信时，通信数据中不可避免地会产生冗余. 为此，DeepSpeed-MoE[78] 限制张量并行组中只有 1 组具有相同 rank 的 GPU 参与 All-to-All 通信（如图 8 中❶所示），实现以张量并行度的比例减少参与通信的 GPU 数量. 而 DeepSpeed-TED[69] 则在 Megatron-LM[98] 张量并行组中删除重复数据以确保输出激活值没有冗余（如图 8 中❷所示），再通过一个 All-Gather 通信在张量并行组内组装完整的输入激活值. 此外，由于激活检查点技术需要一次附加的前向传播过程，会导致额外的通信开销，DeepSpeed-TED 通过存储第 1 次前向传播过程中 All-Reduce 和

All-to-All 通信的输出来解决这个问题.

一种新颖的后处理方法由 ExFlow[99] 提出. 该方法通过在专家并行的全部 GPU 上存储所有样本的上下文信息, 以移除专家并行的第 2 个 All-to-All 通信, 并根据跨层专家亲和性, 优先在 GPU 内和节点内逐级放置专家, 进一步减少低带宽拓扑链路的数据交换. 但这需要在推理前及每次迭代后使用 All-Gather 通信来确保 GPU 上所有样本上下文的一致性. 此外, 这也增加了大量额外的内存开销.

3.1.3 其他通信优化

除上述介绍的优化方向外, 路由无关通信延迟优化的其他方向还包括针对小批量数据频繁通信以及针对数据切分维度等通信优化.

连续的小批量数据通信调用开销较大, 为此, SE-MoE[57] 在通信前将参数小切片合并为一个较大切片, 并在通信后根据切片索引重新划分为小切片. 此外, 该方法还在预定数量的参数完成反向传播后再统一通信梯度, 进一步降低了通信频率.

沿设备数量维度切分输入数据将使 All-to-All 通信被划分为多个低效的点对点通信. 针对这个问题, MPipeMoE[63] 采用沿输入序列维度切分数据, 实现 All-to-All 通信的细粒度拆分, 以充分利用 NCCL 提供的通信优化. 此外, 该方法还缓解了同步过程中高带宽设备的资源浪费问题.

Janus[58] 则设计了以交错顺序获取其他设备的内部专家, 并确保同一时间只有 1 个设备从同一设备获取专家, 以缓解设备出口带宽竞争. 此外, 伙伴设备以交织顺序并行通过 PCIe, 从 CPU 缓存获取不同外部专家, 同时, 利用 NVLink 获取已驻留在伙伴设备上的其余专家. 伙伴设备并行获取不同专家, PCIe 和 NVlink 链路流水传递相同专家, 缓解了 CPU 到 GPU 的带宽瓶颈.

目前, 路由无关通信优化仍是 All-to-All 通信延迟优化的主流方法, 分层通信方法更是几乎成为 MoE 系统标配的 All-to-All 通信优化实现. 尽管这些方法已经在降低通信延迟方面取得了显著进步, 但受限于样本专家的选择偏好, 通信链路数据容量不均衡问题并未得到根本解决, 仍然存在通信拥塞现象. 此外, 值得注意的是, 近期该方向研究成果表明在通信延迟性能上的提升已经逐渐降低.

如果通信拥塞不可避免, 探索将热门专家和冷门专家与不同带宽的拓扑网络有效结合以降低通信延迟是一个有意义的研究方向. 特别是在推理场景下, 利用在模型训练过程收集到的专家热度、不同样本对专家的敏感性等数据, 通过精心设计的专家放置和调度策略, 有望实现路由无关通信方法难以企及的效果.

3.2 拓扑敏感路由

考虑到路由策略与通信网络拓扑间的不协调会导致严重的网络带宽竞争, 设计针对性的路由策略辅助 All-to-All 高效通信是一种朴素的方法. 拓扑敏感路由方法根据通信网络的拓扑结构和带宽特性自动调整通信链路的数据量, 避免网络拥塞. 本文总结了 3 种实现方法.

1)强制限制节点间通信量. 直接限制路由到其他节点设备专家的样本数量, 将样本优先分发到高带宽的节点内设备专家是最直观的拓扑敏感路由方法. 该思想首先在 Gating Dropout[100] 中提出, 通过设置路由到其他节点设备专家的概率来减少节点间通信开销. 如图 9 所示, 节点 0 的输入 0 和输入 2 以及节点 1 的输入 3 和输入 4 均依据概率被重新映射到局部节点内.

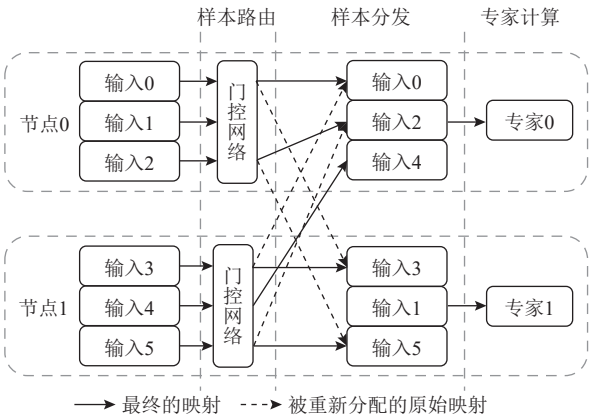


Fig. 9 Method for forcibly limiting the amount of data communication between nodes

图 9 强制限制节点间通信的数据量方法

沿着这个方向, FasterMoE[101] 通过对节点内和节点间 All-to-All 通信数据量分析, 形式化表示节点间通信的样本数量, 进一步优化了 Gating Dropout 对模型精度的影响. Gating Dropout 根据概率将样本分发到本地设备和其他节点设备驻留专家, 不考虑样本和专家间的匹配得分, 且这个概率需根据实验效果恰当选择. FasterMoE 则在节点间通信样本数量超过限定数量 n 时, 将具有最高匹配得分的 n 个样本路由到其他节点专家, 而将剩余样本重新路由到本地专家.

2)自适应调整链路通信量. 根据网络拓扑的带宽特性自适应动态调整通信链路的通信量则是对 1)中方法的一种自然改进. TA-MoE[102] 将样本分发抽象

为最小化多跳通信中最低带宽链路的通信开销问题. 具体而言，该方法根据设备间通信的最短路径所需经过的交换机数量将对称树拓扑结构的设备分组，并平滑组内设备的通信模型系数，从而简化了原始异质网络拓扑的通信模型. 此外，环形拓扑和不规则非对称拓扑结构具有或可转化为对称拓扑结构的最优解模式. 但该方法需要对通信链路精确建模，否则将影响最终结果的准确性.

3)融合方法. 通过机器学习方法选择样本的通信拓扑链路则进一步优化了上述的数值模型方法. 基于上述研究进展，SCoMoE[103] 提出了一种融合强制限制节点间通信量和自适应调整链路通信量这 2 种策略的结构化通信 MoE 架构. 具体而言，该方法在序列维度采用与 FasterMoE 类似的强制限定策略，优先通过加速器内或节点内通信传输与本地节点专家具有最高得分的预设数量数据，而将剩余数据通过节点间通信处理. 在特征维度，则采用线性网络自适应地将特征映射为加速器内、节点内、节点间通信数据. 然而，该方法引入了新的超参数，需精细调整以优化模型性能. 此外，其自适应通信方法增加了模型的复杂性和计算开销，引入了额外的执行延迟.

拓扑敏感路由是一种融合了通信延迟优化的改进路由方法. 显而易见，对比路由无关通信优化方法，该类方法的通信延迟性能表现通常更优异，但同时也伴随着影响模型精度的风险. 此外，此方法通常只能应用于训练场景，且由于对通信拓扑敏感的固有属性，需要特殊设计损失项以辅助损失函数收敛，这给模型训练增加了困难.

4 MoE 计算效率优化

计算效率是决定模型执行效率的最直接因素. 现有文献主要从专家动态容量、专家和设备负载均衡、专家激活预测、计算和通信流水调度以及计算内核优化 5 个方向优化 MoE 模型的计算效率.

4.1 动态容量

为满足 All-to-All 通信和专家计算的系统性约束，固定专家容量已经成为 MoE 模型的标准配置. 但这不可避免地需要在模型精度和计算效率间折中选择. 一种新兴的解决方案是专家动态容量方法，其根据专家的实际需求为其分配不同容量.

专家动态容量因避免处理冗余的填充数据，从而提升通信和计算效率. 但其面临着因数据形状不一致引发的低效编译、低效数据分发和低效批量矩阵计算所带来的挑战，如图 10 所示.

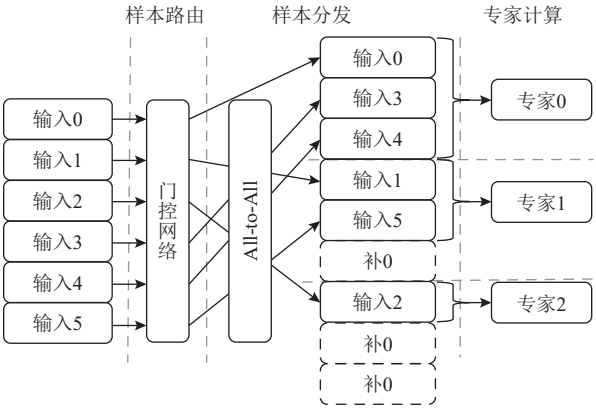


Fig. 10 Dynamic capacity of experts

图 10 专家动态容量

由于当前深度学习框架在处理动态数据流方面存在的低效瓶颈，为支持专家动态容量，用户只能采用一些变通方法，这不仅带来显著的运行时开销，甚至还会牺牲模型精度. 为了解决这个问题，DynaMoE[104] 引入动态重编译机制，按需为动态张量重新分配资源，避免了完全重新编译整个计算图带来的巨大开销，并基于此机制按需动态调整专家容量. 对专家动态容量的编译支持本质上属于 AI 编译研究领域，该领域对动态张量问题取得的系列研究成果[105-108] 均可为本问题的解决提供参考和借鉴.

专家动态容量导致专家接收的样本数量不一致，现行的门控样本分发机制不能高效处理此类动态样本分发. 为此，文献 [53] 提出支持专家动态容量的门控样本分发优化方法，通过排序、计数和索引取代原始的矩阵乘操作不仅降低了数据分发的复杂度，也减少了内存占用. 类似地，FastMoE[109] 也设计了支持专家动态容量的专用数据通信内核来实现高效的样本分发和组合.

专家动态容量还意味着同一设备上的多个专家需要处理不同形状的输入数据，这要求高效的可变尺寸批量矩阵乘法实现这些专家的并行计算. 为此，MegaBlocks[110] 提出了高效的块稀疏可变尺寸批矩阵乘法内核重写专家计算. 具体而言，该方法将可变尺寸批量矩阵乘法分解为多个 128×128 固定尺寸的基础矩阵乘法块批量处理. 同时设计了块压缩稀疏行–坐标(CSR-COO)混合格式，配合非零块行索引和转置索引等数据结构设计，优化对稀疏矩阵的访问和转置操作. 但该方法主要针对在单一设备上放置多个专家的场景，对于专家数量较多的模型，可能会面临挑战. 类似地，PIT[111] 利用置换不变变换巧妙地将

多个稀疏分布的微小张量块转换为 GPU 执行高效的稠密块, 并在运行时通过精心设计的数据重排原语来快速执行这些变换, 从而几乎不产生额外的运行开销. 需要强调的是, 该方法并不局限于解决低效的可变尺寸批量矩阵乘法问题, 还为张量计算中的动态稀疏性问题提出了普适的编译优化解决方案.

上述方法将可变尺寸批矩阵乘法问题转化为稀疏矩阵形式, 这涉及将索引转换为稀疏矩阵的块偏移操作, 从而导致额外的计算开销. 此外, 对输入执行从分散到组合的初始拷贝也会导致额外的内存开销. 为应对这些挑战, ScatterMoE[112] 通过特殊设计的线性模块在分散组高效执行分组矩阵乘法. 该模块可以实现多种 MoE 变换的不同组合, 同时还允许输入张量和输出张量以组合或分散的形式存在. 更重要的是, 通过融合分散和组合以及线性操作, 避免了对输入的拷贝和补 0. 此外, 该方法还具有高度的可扩展性, 可以轻松将现有的 MoE 方法扩展到其他类型的专家模块.

此外, Tutel[96] 通过在不同迭代过程动态调整容量因子以最大程度保证样本不被丢弃. 然而其并不支持为不同专家配置不同容量, 这导致其无法避免因填充样本而产生的无效通信和计算.

专家动态容量与专家动态激活的固有属性相契合, 但这也给系统性能带来严峻挑战. 尽管上述工作已取得一定进展, 但与固定专家容量相比, 专家动态容量仍然在模型执行效率方面存在不足. 究其原因, 现有的加速器设计更适合处理形状规则、统一的张量, 对不规则形状张量的高效处理仍是机器学习系统领域亟待解决的一个难题.

4.2 负载均衡

专家的选择偏好会导致热门专家负载过高, 而冷门专家负载偏低. 这种负载不均衡使得热门专家驻留设备成为训练和推理的系统性能瓶颈, 而冷门专家驻留设备则资源利用不充分, 进而降低了模型的吞吐性能并增加了执行延迟. 现有的负载均衡研究方法主要包括专家负载均衡以及设备负载均衡.

4.2.1 专家负载均衡

设备负载不均衡源于专家负载不均衡. 专家负载均衡技术直接从 MoE 模型负载不均衡问题的根源出发, 避免了仅从系统角度出发的设备负载均衡方法的局限性. 目前, 专家负载均衡研究主要集中在负载均衡损失以及路由均衡门控.

1)负载均衡损失. 通过在损失函数中引入辅助损失来抑制专家负载不均衡是一种常见的方法[7,19-21,36,113]. 但负载均衡损失的权重需要精心设计以免影响主损失函数. 即便如此, 由于不可避免的样本丢弃仍会导致模型精度的下降[110,114]. 该类方法在一定程度上抑制了专家负载不均衡, 但其属于单纯的算法设计方法, 并不属于本文主要关注的系统优化工作, 在此不过多介绍.

2)路由均衡门控. 另一种实现专家负载均衡的方法通过特殊的门控网络设计实现样本路由均衡. 相比负载均衡损失方法, 路由均衡门控方法直接调控样本—专家映射, 而非依赖于后续调整, 提供了一种更直接且更高效的解决方案.

改进传统门控、优化专家负载的最经典方法是固定专家容量[36], 其通过限制允许专家接收的最大输入样本数量来抑制专家负载不均衡. 类似地, 直接在门控网络中加入高斯噪声也可以鼓励专家负载均衡[20]. 值得注意的是, 文献 [115] 采用后处理方法矫正 Top-k 路由. 其通过将丢弃样本重新分发到样本所在 GPU 的其他专家, 并将专家的填充样本替换为以该专家为次最优专家选择的其他样本, 从而取得有益效果.

上述工作虽然在专家负载均衡方面取得了一定成效, 但仍未实现每个专家接收相同数量输入样本的绝对均衡. 鉴于此, 探索更高效的均衡门控设计成为研究热点.Base layers[116] 通过将输入—专家映射建模为带有约束的线性分配问题, 从而解决了这个问题. 在此基础上, S-BASE[117] 采用最优传输问题建模专家分配, 并通过 Sinkhorn 算法求解专家选择的近似匹配, 优化了 Base layers 的分配策略. 沿着这个方向, 更进一步地, 通过带有稀疏约束的最优传输问题, 确保最多只有预设数量的输入样本与 1 个专家匹配[118], 优化了 S-BASE 方法. 作为对传统可学习门控方法的补充, 研究人员还将均衡门控的研究范围扩展到非可学习门控. 例如, 使用哈希函数而不是经训练学习的通用门控网络决定输入样本—专家映射, 以解决专家负载均衡问题[119]. 而 THOR[120] 则进一步松弛了专家和样本的映射关系, 通过随机路由策略缓解负载不均衡并取得了优异的模型精度. 与上述输入样本选择专家路由策略形成鲜明对比的是专家选择输入样本路由策略[114]. 该方法根据样本对专家的重要程度为每个专家分发数量相等的输入样本, 但仍存在丢弃样本的问题. 一种新颖的软路由机制[121] 基于可学习参数将所有输入样本的不同线性组合交由不同专家处理, 从而解决了专家负载不均衡问题. 值得一提的是, 该方法还是完全可微的, 避免了样本路由

过程的低效稀疏操作. 然而, 该方法只适用于具有固定输入长度的编码器模型, 在自回归解码器模型的应用存在挑战.

专家负载均衡方法因其实现简单且通常对系统无额外副作用而广受青睐, 尽管该类方法在一定程度上实现了均衡负载, 但存在以下局限: 首先, 其设计理念与专家的选择偏好固有属性存在潜在冲突, 可能会削弱专家的专业性. 其次, 特殊的路由门控设计如果片面强调负载均衡, 难免会对模型精度造成负面影响[122].

此外, 上述专家负载均衡方法中的路由门控通常都是离散、稀疏的, 这会导致样本—专家映射操作的低效. 虽然稠密的路由设计研究已广泛开展[123-125], 但这些工作通常没有考虑专家负载均衡问题. 因此, 稠密的专家负载均衡路由设计是一个值得探索而未被充分研究的课题.

4.2.2 设备负载均衡

负载均衡的另一个研究方向是通过调整专家放置策略(专家—设备映射)来均衡设备负载. 该类方法的核心思想是为高负载专家分配更多资源, 而为低负载专家分配较少资源, 以提高设备资源利用率.

图 11 提供了专家放置策略对设备负载均衡的有益效果示意图. 在图 11(a)中, 各专家的负载并不均衡. 图 11(b)展示了模型运行时未考虑专家放置策略, 会导致 GPU0 的设备负载过高, 而 GPU1 的设备负载偏低. 图 11(c)则展示了经专家放置优化后的设备负载均衡得到显著改善.

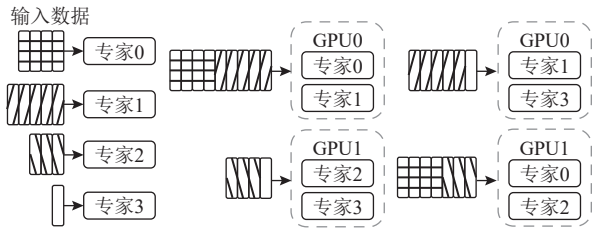


Fig. 11 Illustration of expert placement strategy

图 11 专家放置策略示意图

文献 [53] 工作和 SmartMoE[126] 是将组合高、低负载专家放置到相同设备以实现负载均衡方法的 2 个典型代表. 文献 [53] 将负载均衡问题简化为多路数划分问题, 并为非相关激活专家和相关激活专家分别设计了贪婪均衡方法和反相关均衡方法. 具体而言, 对于非相关激活专家, 根据历史负载降序依次分配给负载最小的 GPU. 而对于相关激活专家, 则在计算工作负载时考虑历史数据中专家间的皮尔逊相关系数. 但为了简化负载均衡问题, 文献 [53] 的方法假定每个 GPU 上都驻留相同数量的专家, 这限制了其应用范围. SmartMoE 在离线阶段聚类多组相互转换代价低且对动态工作负载不敏感的并行策略, 其中只有专家放置是唯一可变策略, 并通过性能模型枚举筛选出最优并行策略. 在运行时阶段, 根据当前工作负载在选定的并行策略上调整专家放置. 然而, 依赖门控参数估计专家负载的方法在局部时间内可能并非最优.

上述方法仅调整专家的放置顺序, 并未改变专家的资源分配. 对于设备的负载均衡, 这常常并非是最佳选择. 另一种更为流行的解决方案是根据专家的实际负载情况动态分配资源. FasterMoE[101] 通过将热门专家复制到多个设备上以减少热门专家驻留设备的工作负载, 但其并不会减少冷门专家的资源分配, 是一种简化的专家放置策略. 专家热度是数据依赖且动态变化的, 该方法通过使用性能模型在运行时评估模型端到端延迟收益以决定是否复制专家到其他设备. Lina[94] 通过专家选择分布估计调整专家资源分配. 该方法设计了一种 2 阶段调度方法, 当实际的专家受欢迎程度偏离估计较多时, 再微调专家资源分配. 但专家选择分布估计依赖训练过程中的数据收集, 这给没有训练过程统计信息的推理任务带来困难. 考虑到 SmartMoE 利用门控网络参数通常只能预测专家负载极限值的局限性, FlexMoE[127] 采用了后验统计方法实时监测专家负载. 基于训练过程中专家的偏好分布变化连续且平稳, 该方法在运行时实时监测专家负载趋势, 并在专家负载超过阈值后, 微调专家放置策略. 为缩小调度空间, 该方法还设计了最小调度单元的抽象表示和专家放置专用调整指令来辅助专家放置策略调整.

文献 [53] 和 Lina 均采用基于统计信息的先验方法估计专家负载, 区别在于: 文献 [53] 直接利用专家历史负载信息估计当前专家负载; 而 Lina 则利用专家对不同输入的选择偏好, 统计输入样本在连续 l 层窗口选择相同专家的概率, 并直接用于推理过程的专家负载预测.

专家放置策略是数据依赖的, 运行时动态预测专家负载, 调整专家放置策略需要一个轻量级的高效方法以满足模型运行的实时性要求. SmartMoE 通过离线时构建并行策略和运行时调整专家放置策略的 2 阶段方式, 放宽了联合搜索的时间开销, 并采用一个轻量级贪婪算法高效评估转换策略以满足运行

时严苛的时间限制. FlexMoE 则通过精确建模 MoE 层的执行过程预先估计计算和通信代价, 利用代价模型预测放置策略调整的成本和收益以决定最优调整策略. 但其通信模型偏理想化, 没有考虑通信与计算、All-to-All 与 All-Reduce 通信并发执行的相互影响, 因此模型精度有限. 此外, FlexMoE 采用同时为高/低负载专家增加/减少资源的简单高效资源调整方法实现专家放置策略调整.

类似的工作还有 SE-MoE[57], 其提出了针对多任务训练场景下的负载均衡方法. 该方法根据预先估计的各任务工作量, 灵活调整各训练任务节点数量, 保证各节点负载均衡, 但通用性不强.

设备负载均衡方法具有 3 个显著优势: 首先, 它保持原始门控的样本—专家映射关系不变, 这通常意味着更优的模型精度; 其次, 虽然其负载均衡效果可能略逊于一些专家负载均衡方法, 但幸运的是两者差距并不显著, 且其性能也足以满足绝大多数应用场景的需求; 最后, 对于那些并未考虑专家负载均衡或实际效果不佳的路由策略, 设备负载均衡方法更是提供了唯一的有效解决方案. 然而, 受限于实际负载的复杂性, 设备负载均衡方法通常需要复杂的设计才能实现与专家负载均衡方法相匹配的均衡效果. 加之专家负载在模型执行过程中动态波动, 且专家在运行时被动态激活, 进一步加剧了设备负载均衡的困难. 此外, 专家放置策略在线调整依赖于模型运行时的专家选择, 这需要高效的评估方法来综合衡量调整成本和性能收益.

负载均衡是 MoE 模型高效执行的一个关键要素. 鉴于样本分布的不均匀性决定了专家选择的固有非均等分布属性, 进一步探索基于专家负载均衡设计并结合神经网络模型预测专家负载的设备负载均衡方法, 是一个值得深入研究的方向.

4.3 激活预测

专家选择、样本通信和专家计算的同步执行严重阻碍了 MoE 系统优化的潜力. 专家激活预测通过在门控网络选择专家前预测激活专家, 为专家资源分配以及上述流程并行执行提供优化空间. 根据专家激活分布估计的方式, 本文进一步将相关方法分为基于统计和基于机器学习的专家激活预测方法.

4.3.1 基于统计的激活预测

根据专家选择的统计信息可以估计专家选择模式. 一般而言, 基于统计的专家激活预测方法对输入样本个例的专家选择预测不够准确, 只限于估计专家的选择分布. 因此, 该类方法通常只应用于系统资源分配,

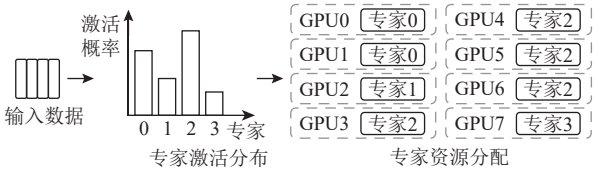


Fig. 12 Resources allocation utilizing statistical-based activation prediction method

图 12 利用基于统计的激活预测方法分配资源

因为此时输入样本个例的专家激活预测是否准确并非至关重要, 只要统计意义上准确即可满足需求.

图 12 为根据专家激活分布调整计算资源的示意图, 根据输入数据的专家激活分布预测结果, 将专家 2 和专家 0 的计算资源由原始平均分配的 2 个 GPU 分别增加到 4 个 GPU 和 2 个 GPU, 以平衡设备负载.

在基于统计的专家激活预测方向上做出初步探索的是 DynaMoE[104], 其研究了基于样本分发缓存预测专家激活模式的方法. 基于门控网络在训练中早期收敛的特性, DynaMoE 通过度量连续 2 个训练历元间一个小批量样本中被重新分发给不同专家的数量来检测门控是否收敛. 如果门控收敛, 便缓存样本分发结果并用于创建后续批量专家计算.

一些门控网络参数设定了通信和专家计算负载的上限, 这为我们提供了一种估计专家负载极限情况的方法. SmartMoE[126] 根据负载均衡门控网络容量因子估计最高负载专家的计算开销上限, 并根据拓扑敏感门控网络的通信约束估计设备间的最大通信开销, 从而得到专家选择分布, 进而利用以计算负载、通信负载及专家选择分布作为参数的性能模型搜索出最优的并行策略. 但利用门控网络参数估计专家选择的统计方法略显粗略, 该方法的并行策略性能预测效果还需进一步验证.

注意到 MoE 模型在前一层选择相同专家的样本在后一层同样倾向于选择相同专家, Lina[94] 利用相邻层专家的选择模式实现专家选择分布估计. 具体而言, 在训练时, 收集专家选择结果, 对连续 l 层选择相同专家的样本分组, 并对每组样本统计 l+1 层的专家偏好分布. 在推理时, 在第 l+1 层根据之前 l 层的样本分组以及训练过程统计的各组样本专家偏好估计专家选择分布, 进而为专家分配资源. 但该方法中的参数 l 需要在预测精度和统计开销中折中选择, 且浅层专家的选择偏好并不十分突出, 预测效果有限.

基于统计的激活预测方法实现较为简单, 预测精度也基本能够满足资源分配应用场景的需求. 但其存在 2 个不足: 首先, 严重依赖训练过程中的大量

数据收集，给没有经历训练过程的推理任务带来困难. 其次，该方法略显粗糙，对输入样本个例的专家选择预测，其结果也不够准确.

4.3.2 基于机器学习的激活预测

提高专家激活预测准确性的流行方法是利用机器学习模型学习专家的样本选择分布，进而精准预测输入样本的专家选择. 这意味着原始门控网络的功能被取代，解耦了专家选择和专家执行的依赖关系. 基于机器学习的激活预测方法，其本质上是一个数据回归过程，相对于前述的统计方法更智能也更准确.

与传统 MoE 模型本层门控网络选择本层激活专家不同，Pre-gated MoE[56] 引入了预先门控设计，当前 MoE 层门控网络会预先为下一 MoE 层选择激活专家，从而完全消除了专家选择与专家执行间的依赖. 该模型的第 1 层 MoE 层包含传统门控和预先门控 2 个门控网络. 此外，最后一层 MoE 层没有门控网络. SiDA[128] 则摒弃了 Pre-gated MoE 采用的模型内置门控网络，转而使用了一个独立的、离线训练的机器学习模型来提前预测输入样本的激活专家，并将预测结果存储在哈希表中，如图 13 所示. 这种设计允许不同批次的输入样本推理、激活专家获取和参数加载并行执行. 由于模型的推理速度慢于哈希表构建，完全隐藏了激活专家选择和加载过程，因此，推理过程完全取消了传统门控网络.

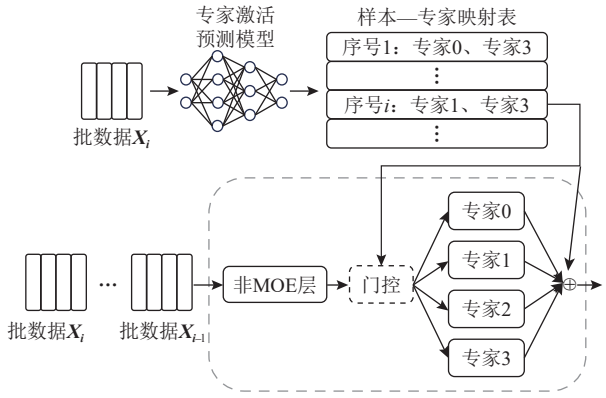


Fig. 13 Illustration of machine learning-based expert activation prediction method

图 13 基于机器学习的专家激活预测方法示意图

上述 2 种方法的区别主要在于：Pre-gated MoE 利用上一层而非本层输入特征选择本层激活专家，专家选择的合理性有待进一步理论验证；SiDA 则使用 LSTM 模型学习门控网络的样本—专家映射过程，本质上是门控网络的离线实现.

相比于基于统计的激活预测方法，基于机器学习的激活预测方法提高了对输入样本个例的预测准确性. 利用基于机器学习的专家选择预测模型取代门控网络，为并行专家选择和专家计算提供了可能，有助于提高硬件资源利用率. 需要指出的是，尽管基于机器学习的激活预测方法对专家选择的预测较为准确，但目前尚与原始门控网络存在一定差距，这将在一定程度上影响模型精度.

受 ExFlow[99] 利用不同层专家的亲和性消除 All-to-All 通信的启发，探索利用专家间亲和性和专家专业性等特性，根据当前输入样本、当前 MoE 层激活专家预测下一 MoE 层激活专家的专家激活预测方法是一个有价值的研究方向.

4.4 流水调度

流水调度通过将连续执行的算子切分为细粒度子集并分配给不同执行单元流水并行执行，以提高算子并行性，是模型训练和推理的一种经典优化技术[129-134]. 流水调度能够充分挖掘 MoE 模型中 All-to-All 通信和专家计算的并行优化空间，提高模型执行效率. 根据流水设置方法，本文将流水调度方法分为静态流水和自适应流水.

4.4.1 静态流水调度

静态流水调度通过人工确定流水粒度，拆分 All-to-All 通信、专家计算以及 All-Reduce 通信、自注意力计算为细粒度算子，并精心排布这些细粒度算子的流水执行过程，以减少流水气泡，最大程度隐藏通信延迟. 计算和通信的交织执行充分利用不同硬件资源以降低模型执行延迟.

MoE 层前向传播过程中通信和专家计算流水并行执行的时间线概略示例如图 14 所示，可以看到细粒度的算子拆分并未带来预期的性能提升，揭示出算子拆分粒度和微算子精细排布的重要性.

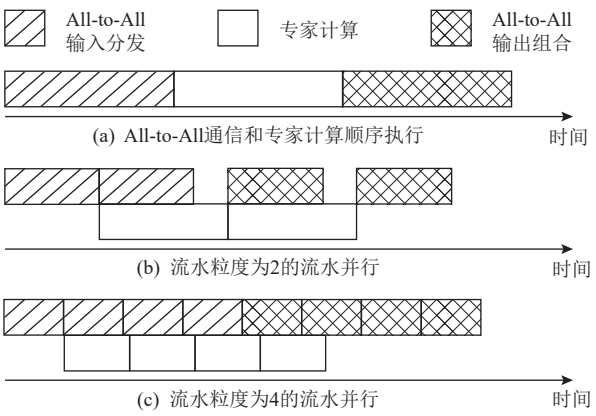


Fig. 14 An example timeline of forward propagation pipeline parallelism in an MoE layer

图 14 MoE 层前向流水并行的时间线示例

FasterMoE[101]将设备划分为细粒度的组并拆分All-to-All通信,使用分组成对交换算法执行.具体而言,这些组形成一个大小为n的环,以0~n−1的递增步长向其他组发送数据,并采用启发式方法将连接紧密的设备放在同一组,以加快同组设备间通信.通过将执行最快的2个发送和接收输入样本和专家输出操作安排在流水队列的起始和末尾以减少通信开销.

反向传播过程中,All-to-All与All-Reduce通信因竞争网络带宽而导致执行延迟被延长,进而导致后续专家计算的阻塞时间增加,影响MoE模型通信和计算效率.为解决这个问题,Lina[94]通过拆分All-to-All、All-Reduce和专家计算为细粒度算子,同时并行这些细粒度算子、梯度计算和门控计算,并优先于All-Reduce执行All-to-All,以改进All-to-All通信的带宽利用率.此外,该方法还通过打包专家计算来减少因细粒度专家计算执行快于相应通信而导致的流水气泡,最大化流水效率.

文献[94,101]工作聚焦于MoE层计算和通信的调度优化,而忽视了与自注意力层联合调度优化的机会.鉴于此,AMPipe[135]将自注意力层与MoE层的计算和通信整合为一个统一的调度优化空间,并通过将自注意力计算、专家计算和All-to-All通信拆分为细粒度子块,流水执行以减少模型执行延迟.该方法对长序列语句的处理速度提升尤其显著,而这也是大语言模型重要的现实应用场景之一.

进一步地,Lancet[136]深入到模型的训练图级探索更广泛的优化空间,以便重叠通信和计算.具体而言,在前向传播过程,通过细粒度算子拆分流水非MoE计算和All-to-All通信.在反向传播过程,无需拆分算子,通过调度重叠梯度计算和All-to-All通信.值得注意的是,基于编译优化方式的优势使其便于精确控制算子执行,并简化了算子调度和算子拆分实现的复杂度.

一般而言,静态流水方法需要对不同模型进行大量实验以确定各自的最优流水粒度,这无疑使其通用性大打折扣.值得庆幸的是,研究表明最优流水粒度的取值范围相对有限,且所需的实验成本也尚可接受.此外,由于专家工作负载的动态变化,静态流水方法并不能保证其流水配置在模型运行过程中始终处于最优.

尽管存在这些局限,静态流水方法因其简单性和易实现性而被众多MoE模型广泛应用.与MoE模型或神经网络框架的良好兼容性也是其另一显著优势.此外,尽管静态流水方法所确定的流水配置未必最优,但相比最优流水配置的性能差距并不显著,这使其在非极致性能追求的应用场景中成为一个实用的选择.而这也是静态流水方法在MoE模型中仍广泛应用的一个重要因素,凸显了其重要价值.

4.4.2 自适应流水调度

自适应流水调度能够根据硬件配置和模型任务自动调整最优流水调度配置,避免了人工设置流水调度的不足.通过暴力搜索和性能分析模型确定最优流水配置是当前主流的自适应流水方法.

一般而言,流水粒度会随着批处理量的增加而单调增加.受此启发,Tutel[96]和MPipeMoE[63]都采用了对搜索变量划分区间的方法,以降低搜索代价.MPipeMoE将批处理量的范围分割为不相交区间,并将此区间与最优流水粒度的映射存储在查找表中.当不能在查找表中查询到当前批处理量信息时,便通过实验搜索出最优流水配置并更新查找表.与MPipeMoE搜索不同批处理量相对应的最优流水粒度不同,Tutel搜索的是不同专家容量对应的张量并行度、流水粒度和All-to-All通信算法的最优配置组合.其提前通过实验,利用字典记录上述映射关系,并在运行时查找字典以获取所需的最优配置.

除采用暴力搜索和查表相结合的方法外,自然地还可以进一步利用数值模型分析方法来实现自适应流水专家计算和All-to-All通信.沿着这个方向,PipeMoE[137]形式化表达以流水粒度为参数的模型执行延迟优化问题.通过构建计算和通信延迟预测模型,将最优流水粒度求解分别转化为数值比较、凸优化和线性规划问题,并在多项式复杂度时间内高效得到最优解.然而,实验结果表明该方法并未实现预期的理论加速效果,虽然探讨了潜在的可能原因,但缺乏量化数据的支持.

PipeMoE可以根据模型配置提前自动获得最优流水粒度,Tutel也可根据预先构建的查找表在运行时以几乎零开销的代价得到最优流水粒度,而MPipeMoE则需通过实验方法在运行时搜索查找表中未存储批处理量所对应的最优流水配置.Tutel的查找表可以提前构建,而MPipeMoE的查找表必须在线构建,二者本质区别在于Tutel是根据专家容量确定最优流水配置,而MPipeMoE是任务相关的.MPipeMoE不同任务的批处理量对应的最优流水配置并不相同,因而不能提前构建.但对于同一任务的相同批处理量,MPipeMoE也仅需在线搜索1次.

相比于FasterMoE的性能模型,PipeMoE的性能模型考虑了计算过程GPU内核启动和通信过程All-to-All启动的时间,模型更加准确.但为了简化最优

流水粒度求解，PipeMoE 限定任意 2 个任务不能占用相同的硬件资源，对于诸如 All-to-All 和 All-Reduce 并行执行的这种涉及同类任务间资源竞争的场景，模型预测结果不够准确. 不幸的是，此类场景在深度学习框架中普遍存在，这限制了其实用性.

上述自适应流水方法存在明显不足：暴力搜索和查表相结合方法是对静态流水方法搜索过程的一种自动化改进，但受搜索空间范围的影响，其搜索速度和模型性能通常不可兼得，需要在两者间权衡选择. 而数值模型分析方法则受限于建模的准确性，特别是在复杂通信场景下，其精度难免会受到不同程度的影响.

创新地探索利用神经网络模型辅助的自适应流水并行调度方法，是一个充满潜力的新优化方向. 此外，当前工作均致力于从现有 MoE 架构挖掘流水调度的潜力. 改变现有 MoE 模型中注意力计算和专家计算串行的基本架构，通过分支连接方式并行这 2 个计算单元，能够最大程度解决目前的流水调度难题. 这是一个充满挑战，同时也蕴藏巨大潜力的研究方向.

4.5 内核优化

除 4.1~4.4 节介绍的方向外，MoE 模型计算效率优化还可通过内核优化实现. 常见的高性能计算库能为通用计算场景提供优异性能表现，但并不适合门控稀疏的样本—专家映射以及不同形状数据的专家并行计算等操作. 通过定制化的内核优化上述计算流程，是解决这些问题的有效手段.

1）门控内核优化. 门控网络的样本—专家映射以及专家—样本映射过程通常涉及稀疏矩阵操作，通用加速器对此并不高效. 为此，DeepSpeed-MoE[78] 融合门控网络的多个算子为一个内核，配合稠密的样本—专家映射表设计，以及数据布局转换替换稀疏的 einsum 操作，显著减少了内核调用次数，并降低了计算和内存开销. HetuMoE[95] 设计了支持多种门控策略及高度优化的内核实现. 其优化了主流 Top-k 算子内核，相比 Pytorch 实现取得 25% 的平均速度提升. 此外，该方法还优化了数据布局转换，设计了优化内核加速映射到同一专家的样本放置到连续内存地址的数据布局处理过程. 类似地，Tutel[96] 也设计了高度优化的编码、解码内核，利用高效的单指令多线程（SIMT）稀疏操作深度优化样本分发和组合过程的 einsum 和矩阵乘操作，最小化非专家计算延迟和降低 GPU 内存占用.

2）其他内核优化. 除上述的门控内核优化方法外，还有一些其他方向的内核优化工作. 例如 FastMoE[109] 为 MoE 模型设计了高度并行优化的 MoE 层内核实现，并通过定制的流管理器并发执行多个专家计算，提高专家计算吞吐性能. 此外，文献 [138] 面向推理场景，也重点优化了计算内核.

5 MoE 并行扩展优化

神经网络框架普遍应用数据并行[139]、张量并行[98,130]、流水并行[131-133]、混合并行[48,98,130,133,140-142] 以及自动并行技术[143-146] 扩展大模型到众多加速器，从而突破加速器的“内存墙”限制，提高训练和推理效率. 然而，这些面向传统稠密模型的并行技术缺少对稀疏 MoE 模型的针对性支持. 探索高效的 MoE 模型并行扩展技术已经成为一个亟待解决的问题，本文从静态并行和自适应并行 2 个方向介绍现有工作.

5.1 静态并行

静态并行在模型运行前根据模型配置确定并行策略，并在运行期间保持不变，静态地执行并行策略. 以深度学习框架成熟的并行策略为基础，结合 MoE 模型专家参数量占比大、专家稀疏激活等特性发展而来的静态并行技术是解决 MoE 模型高效并行扩展的一种直接方法.

专家并行[36] 将不同专家放置在多个设备上并行执行，而非专家层则以数据并行方式执行，是 MoE 模型最朴素的高效并行策略. 专家并行本质上是一种针对 MoE 模型的数据和张量混合并行方法，如图 15 所示. 其中，图 15 (b) 中的专家 00 表示专家 0 的第 1 个张量划分. 专家并行将专家放置在不同设备上，因此，需要执行额外的 All-to-All 通信将输入样本分发给其最匹配的专家，并在专家处理后将专家输出恢复到样本原始位置，引入了额外的通信开销. 专家并行已经成为 MoE 模型的典型并行模式，被后续研究工作广泛采用.

一系列工作[68-69,78,109,147] 并未局限于专家并行策略，而是结合神经网络框架成熟的并行策略和专家放置策略，探索更适合 MoE 模型的并行方法. Fairseq-MoE[147] 采用完全分片数据并行（FSDP）[148] 技术训练具有超万亿参数的 MoE 模型. 使用完全分片数据并行的一个重要步骤是选择被包装的子模块粒度，该方法选取 Transformer 层进行包装，以平衡通信效率和峰值内存占用. 文献 [68] 则精心编排了 5 种混合并行策略，通过支持 ZeRO-2 数据并行优化优化器和梯度状态冗余参数，并结合模型并行支持更大隐藏维

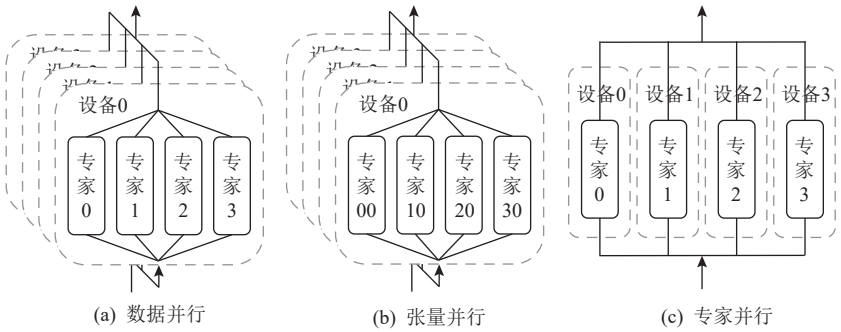


Fig. 15 Data, tensor and expert parallelisms

图 15 数据、张量和专家并行

度规模的模型，将 MoE 模型扩展到数万亿参数规模. FastMoE[109] 对非专家层采用 Megatron-LM[98] 模型并行，对专家层采用专家并行方式，并设计了数据并行通信组标签，通过定制化的数据并行识别不同标签，高效执行参数梯度同步. 该方法通过数据、模型和专家的组合并行方式极大提高了系统的扩展能力，但系统扩展到多节点时，会受通信开销影响而导致单 GPU 性能退化严重.

文献 [68] 提供的并行策略较多，需要用户根据模型自己选择，这无疑增加了用户的使用困难. 针对这个问题，DeepSpeed-MoE[78] 采用张量并行以充分利用 GPU 间的聚合内存带宽，将其支持的多种混合并行策略优化为一种专家、张量和数据混合并行. 具体而言，该方法为专家层和非专家层分别设计了张量和专家混合并行以及张量和数据混合并行方法，解决了张量并行因通信开销和细粒度并行导致的扩展性差问题.

DeepSpeed-TED[69] 则进一步优化了 DeepSpeed-MoE，提出一种结合 ZeRO 数据并行、Mergatron-LM 张量并行、DeepSpeed-MoE 专家并行的 3 维混合并行方法. 其为专家层设计了与张量、数据和专家混合并行方法，为非专家层设计了张量和数据混合并行方法. 相比 DeepSpeed-MoE，其支持的模型规模以张量并行度的比例扩大.

需要指出的是，静态并行方法需要人为确定并行策略，这对用户提出了较高要求. 换言之，用户既需要具备一定的硬件体系结构知识，又需要对模型架构充分了解，才能为模型设置性能优异的并行策略. 同时，静态并行策略不能根据 MoE 工作负载的变化而动态调整，MoE 模型的动态工作负载属性决定其不能保证模型的最优性能. 尽管存在这些局限，静态并行因其使用简单而应用广泛. 通过实验即可确定性能较为优异的并行策略，特别是对于模型频繁重复使用的场景，静态并行方法一旦确定并行配置便可以一直使用，因此，其仍被广泛应用于 MoE 模型的训练和推理.

5.2 自适应并行

不同于 5.1 节讨论的静态并行方法，自适应并行方法能够在运行时根据负载自动调整并行策略，是一种更具潜力的解决办法. 但在运行时动态调整并行策略会导致数据重新布局、数据迁移和 GPU 内存大量消耗等严重开销，这给自适应并行方法带来极大挑战.

如图 16 所示，自适应并行的最大优势是能够为不同批次输入数据在线动态调整并行策略，从而最大化模型的性能表现.

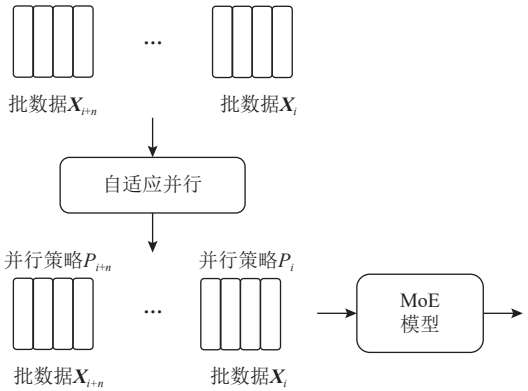


Fig. 16 Overview of adaptive parallel strategy

图 16 自适应并行策略概览

自适应并行的关键是确定最小的可转换并行策略集合和张量数据布局形式，从而以最小的开销代价实现并行策略转换. 基于此，Tutel[96] 提出了动态自适应并行和流水 MoE 系统. 该方法通过剪枝并行策略转换空间，使其只包含数据并行以及专家、数据和模型混合并行 2 种并行策略，并确定了两者相互转换的执行流程. 此外，该方法还为模型参数和输入样本设计了相同的数据布局以实现运行时并行策略动态调整的零开销.

对并行策略转换空间深度剪枝，可以进一步简化自适应并行方法的复杂度. 一种简化的自适应并行方法是在运行时只调整专家放置策略，保持其他并行策略不变[126].

自适应并行方法能够根据 MoE 模型的工作负载变化在运行时动态调整并行策略以提供最佳性能. 但这需要满足模型运行时间的严苛限制，以避免影响模型的正常执行. 因此，需要设计高效的决策机制以评估并平衡搜索开销、转换代价和性能收益，从而决定是否转换到一个新的并行策略.Tutel 开启 MoE 系统自适应并行技术先河，但其并未搜索专家并行粒度维度，因此，可能缺少一些更优的并行策略，这是一个值得继续探索的课题.

6 总结与展望

近年来，MoE 已经成为深度学习领域的重要研究方向之一，但其专家稀疏激活特性有两面性，在以计算开销次线性增加为代价扩展模型参数规模的同时，也给系统端带来前所未有的挑战. 为应对 MoE 系统在高可扩展性和高效性方面的重大挑战，研究学者开展了广泛的系统优化研究，并取得显著进展.

如何平衡存储、通信以及计算开销，以充分利用系统资源，提高系统的可扩展性和计算效率，降低模型端到端延迟是 MoE 系统优化研究的核心. 本文调研分析揭示，当前研究主要通过深入探索内存、通信、计算以及并行 4 个系统核心维度，从而发现潜在的优化机会. MoE 系统优化技术正向着精细化演进，单一维度的执行效率提升已经不足以实现显著的改进. 因此，多数工作采用多维优化策略，全面挖掘系统性能的提升潜力. 表 3 总结了 MoE 系统优化研究的一些代表性工作以及这些技术的适用场景.

尽管 MoE 系统优化工作已取得重要进展，但仍然存在许多值得探索的开放性问题. 下面列举一些未来可能深入探索的研究方向.

1）架构创新. 尽管从系统优化角度出发的 MoE 系统优化方法已取得积极进展，但受限于 MoE 底层架构设计中注意力计算、门控决策、样本分发、专家计算的串行依赖，仍然严重制约着系统优化的潜力. DeepSpeed-MoE[78] 通过改进模型架构，取得令人鼓舞的成果. 进一步探索其他可能的优化架构设计，将赋予 MoE 系统更大的优化潜力. 例如，通过将 MoE 层和自注意力层以分支并行的方式连接；在注意力的前馈层之间间隔设置 MoE 层，并利用非 MoE 层输出作为 MoE 层的输入样本；利用数量有限的专家配合专家参数变换矩阵或者输入特征迁移矩阵，从而实现映射专家的预测效果等设计. 这些架构上的创新将进一步释放 MoE 模型的系统优化潜力，有望解决传统

Table 3 Summary of Representative Works on MoE System Optimization

表 3 MoE 系统优化的代表工作总结

类型	子类	适用场景	文献											
			[69]	[56]	[57]	[58]	[63]	[78]	[109]	[96]	[126]	[94]	[95]	[101]
内存占用	内存卸载	设备内存有限、模型参数规模大、模型推理	√		√		√							
	参数压缩				√			√						
通信延迟	分层通信	通用，特别适合小批量数据通信			√	√		√		√			√	
	冗余消除	模型推理或通过张量并行训练、推理	√					√						
	其他*	通用，Janus 除外			√	√	√			√				
	拓扑敏感路由	模型训练，特别是网络拓扑复杂的集群												√
计算效率	动态容量	通用，特别是专家数量多且偏好差异大的模型							√	√				
	专家负载	负载均衡损失用于训练，路由均衡门控均可												
	设备负载	专家负载差异明显、变化显著，设备数量多			√						√	√		√
	激活预测	模型推理，但通常需要训练过程的统计数据		√							√	√		
	静态流水	非极致性能需求										√		√
	自适应流水	模型规模大、执行开销大、极致性能需求					√			√				
	内核计算	通用						√	√	√			√	
并行扩展	静态并行	同一模型频繁使用，训练、推理开销不大	√					√	√					
	自适应并行	极致性能需求、模型频繁变化、大规模训练推理								√	√			

注：“其他*”代表路由无关通信迟优化中的其他优化.

MoE架构模型在通信延迟、计算效率、流水并行等方面的不足，是一个有价值且具有挑战的研究方向.

2)路由改进. 在传统MoE模型中，由于路由机制的离散性和稀疏性常常导致路由的不稳定、样本丢弃以及负载不均衡等问题. 虽然近来的改进路由设计在提升稀疏操作效率、减少通信延迟、改善负载均衡等方面取得一定进展，但目前尚未有一个路由方法能够同时兼顾这些优势. 例如，旨在优化通信延迟或实现负载均衡的路由策略可能会对模型精度产生负面影响. 同样，面向稀疏操作优化的路由设计通常不能统筹负载均衡. 软路由机制[121]有效解决了传统路由的低效稀疏操作和负载不均衡难题，为改进路由方法提供了新思路. 沿着这个方向，进一步探索能够融合稀疏和稠密优势以及对通信拓扑敏感且负载均衡的高效路由设计是一个有趣的研究方向.

3)高效计算. 专家间的相关性对于精准预测激活专家至关重要. 分析相同样本在不同MoE层的专家激活模式、相同样本在同一MoE层中多个激活专家的选择倾向，以及不同样本激活相同专家的一致性等特性有助于精准预测专家激活模式，进而优化专家参数卸载中的专家精确预取和样本通信、专家计算的并行执行. 这不仅能够显著降低模型的通信延迟，还能提升计算效率，是一个值得深入探索的研究方向. 可能的研究方向还包括设计高效的可变尺寸批量计算内核. 数据形状不规整的高效批量通信和计算一直是一个困扰机器学习系统的难题，现有的解决方法通常采用变通的方式将其转变为形状规整的通信和计算，但这种方法无论是在适用性还是性能方面仍存在诸多限制. 因此，对可变尺寸批量计算内核的探索仍将继续，但从现有的研究成果来看，这需要硬件设计层面的创新支持，才有希望获得突破. 此外，探索覆盖范围更广、决策效率更高、转换代价更低以及代价模型辅助的高效自适应并行策略也是一个有价值的研究方向.

作者贡献声明：史宏志调研文献，制定了论文架构，指导论写作并修改；赵健收集和梳理相关文献，负责论文主体内容撰写；赵雅倩和李茹杨指导了论文整体框架和内容规划；魏辉补充完善论文；胡克坤、温东超以及金良对论文提出修改意见.

参 考 文 献

[1] Kaplan J, McCandlish S, Henighan T, et al. Scaling laws for neural language models[J]. arXiv preprint, arXiv: 2001.08361, 2020

[2] Raffel C, Shazeer N, Roberts A, et al. Exploring the limits of transfer learning with a unified text-to-text transformer[J]. Journal of Machine Learning Research, 2020, 21(140): 1−67

[3] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners[C] //Proc of the 34th Int Conf on Neural Information Processing Systems. New York: Curran Associates, 2020: 1877−1901

[4] Rae J W, Borgeaud S, Cai T, et al. Scaling language models: Methods, analysis & insights from training Gopher[J]. arXiv preprint, arXiv: 2112.11446, 2021

[5] Chowdhery A, Narang S, Devlin J, et al. PaLM: Scaling language modeling with pathways[J]. arXiv preprint, arXiv:2204.02311, 2022

[6] Jacobs R A, Jordan M I, Nowlan S J, et al. Adaptive mixtures of local experts[J]. Neural Computation, 1991, 3(1): 79−87

[7] Riquelme C, Puigcerver J, Mustafa B, et al. Scaling vision with sparse mixture of experts[C] //Proc of the 35th Int Conf on Neural Information Processing Systems. New York: Curran Associates, 2021: 8583−8595

[8] Fan Zhiwen, Sarkar R, Jiang Ziyu, et al. M^{3}ViT: Mixture-of-experts vision transformer for efficient multi-task learning with model-accelerator co-design[C] //Proc of the 36th Int Conf on Neural Information Processing Systems. New York: Curran Associates, 2022: 28441−28457

[9] Li Bo, Shen Yifei, Yang Jingkang, et al. Sparse mixture-of-experts are domain generalizable learners[J]. arXiv preprint, arXiv: 2206.04046, 2023

[10] Xue Fuzhao, Shi Ziji, Wei Futao, et al. Go wider instead of deeper[C] //Proc of the 36th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2022: 8779−8787

[11] Shen Sheng, Yao Zhewei, Li Chunyuan, et al. Scaling vision-language models with sparse mixture of experts[J]. arXiv preprint, arXiv: 2303.07226, 2023

[12] Dai Yong, Tang Duyu, Liu Liangxin, et al. One model, multiple modalities: A sparsely activated approach for text, sound, image, video and code[J]. arXiv preprint, arXiv: 2205.06126, 2022

[13] Mustafa B, Riquelme C, Puigcerver J, et al. Multimodal contrastive learning with LIMoE: The language-image mixture of experts[C] //Proc of the 36th Int Conf on Neural Information Processing Systems. New York: Curran Associates, 2022: 9564−9576

[14] Kumatani K, Gmyr R, Salinas F C, et al. Building a great multi-lingual teacher with sparsely-gated mixture of experts for speech recognition[J]. arXiv preprint, arXiv: 2112.05820, 2022

[15] You Zhao, Feng Shulin, Su Dan, et al. SpeechMoE: Scaling to large acoustic models with dynamic routing mixture of experts[J]. arXiv preprint, arXiv: 2105.03036, 2021

[16] You Zhao, Feng Shulin, Su Dan, et al. Speechmoe2: Mixture-of-experts model with improved routing[C] //Proc of the 2022 IEEE Int Conf on Acoustics, Speech and Signal Processing. Piscataway, NJ: IEEE, 2022: 7217−7221

[17] Li Dingcheng, Li Xu, Wang Jun, et al. Video recommendation with multi-gate mixture of experts soft actor critic[C] //Proc of the 43rd Int ACM SIGIR Conf on Research and Development in Information

Retrieval. New York: ACM, 2020: 1553−1556

[18] Cao Zelin, Xu Jun, Dong Zhenhua, et al. Prediction of the positional propensity scores based on multi task learning[J]. Journal of Computer Research and Development, 2023, 60(1): 85−94 (in Chinese)
(曹泽麟, 徐君, 董振华, 等. 基于多任务学习的位置倾向性得分预测算法[J]. 计算机研究与 发展, 2023, 60(1): 85−94)

[19] Fedus W, Zoph B, Shazeer N. Switch Transformers: Scaling to trillion parameter models with simple and efficient sparsity[J]. arXiv preprint, arXiv: 2101.03961, 2021

[20] Shazeer N, Mirhoseini A, Maziarz K, et al. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer[J]. arXiv preprint, arXiv: 1701.06538, 2017

[21] Zoph B, Bello I, Kumar S, et al. ST-MoE: Designing stable and transferable sparse expert models[J]. arXiv preprint, arXiv: 2202.08906, 2022

[22] Lee-Thorp J, Ainslie J. Sparse mixers: Combining MoE and mixing to build a more efficient bert[J]. arXiv preprint, arXiv: 2205.12399, 2022

[23] Kudugunta S, Huang Y, Bapna A, et al. Beyond distillation: Task-level mixture-of-experts for efficient inference[J]. arXiv preprint, arXiv: 2110.03742, 2021

[24] Du Nan, Huang Yanping, Dai A M, et al. GLaM: Efficient scaling of language models with mixture-of-experts[C] //Proc of the 39th Int Conf on Machine Learning. New York: PMLR, 2022: 5547−5569

[25] Lou Yuxuan, Xue Fuzhao, Zheng Zangwei, et al. Cross-token modeling with conditional computation[J]. arXiv preprint, arXiv: 2109.02008, 2022

[26] Lin Junyang, Men Rui, Yang An, et al. M6: A Chinese multimodal pretrainer[J]. arXiv preprint, arXiv: 2103.00823, 2021

[27] Lin Junyang, Yang An, Bai Jinze, et al. M6-10T: A sharing-delinking paradigm for efficient multi-trillion parameter pretraining[J]. arXiv preprint, arXiv: 2110.03888, 2021

[28] Ren Xiaozhe, Zhou Pingyi, Meng Xinfan, et al. PANGU-Σ: Towards trillion parameter language model with sparse heterogeneous computing[J]. arXiv preprint, arXiv: 2303.10845, 2023

[29] Jiang A Q, Sablayrolles A, Roux A, et al. Mixtral of experts[J]. arXiv preprint, arXiv: 2401.04088, 2024

[30] Nguyen H D, Chamroukhi F. Practical and theoretical aspects of mixture - of - experts modeling: An overview[J]. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery, 2018, 8(4): e1246

[31] Masoudnia S, Ebrahimpour R. Mixture of experts: A literature survey[J]. Artificial Intelligence Review, 2014, 42(2): 275−293

[32] Yuksel S E, Wilson J N, Gader P D. Twenty years of mixture of experts[J]. IEEE Transactions on Neural Networks and Learning Systems, 2012, 23(8): 1177−1193

[33] Fedus W, Dean J, Zoph B. A review of sparse expert models in deep learning[J]. arXiv preprint, arXiv: 2209.01667, 2022

[34] Liu Tianlin, Blondel M, Riquelme C, et al. Routers in vision mixture of experts: An empirical study[J]. arXiv preprint, arXiv: 2401.15969, 2024

[35] Cai Weilin, Jiang Juyong, Wang Fan, et al. A survey on mixture of experts[J]. arXiv preprint, arXiv: 2407.06204, 2024

[36] Lepikhin D, Lee H, Xu Yuandong, et al. GShard: Scaling giant models with conditional computation and automatic sharding [J]. arXiv preprint, arXiv: 2006.16668, 2020

[37] Zhang Zhengyan, Lin Yankai, Liu Zhiyuan, et al. MoEfication: Transformer feed-forward layers are mixtures of experts[J]. arXiv preprint, arXiv: 2110.01786, 2021

[38] Zuo Simiao, Zhang Qingru, Liang Chen, et al. MoEBERT: From BERT to mixture-of-experts via importance-guided adaptation[J]. arXiv preprint, arXiv: 2204.07675, 2022

[39] Zhu Tong, Qu Xiaoye, Dong Daize, et al. LLaMA-MoE: Building mixture-of-experts from LLaMA with continual pre-training[J]. arXiv preprint, arXiv: 2406.16554, 2024

[40] Dai Damai, Deng Chenqi, Zhao Chenggang, et al. DeepSeekMoE: Towards ultimate expert specialization in mixture-of-experts language models[J]. arXiv preprint, arXiv: 2401.06066, 2024

[41] Xue Fuzhao, Zheng Zian, Fu Yao, et al. OpenMoE: An early effort on open mixture-of-experts language models[J]. arXiv preprint, arXiv: 2402.01739, 2024

[42] xAI. Open release of Grok-1[EB/OL]. [2024-08-02]. https://x.ai/blog/ grok-os

[43] Reid M, Savinov N, Teplyashin D, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context[J]. arXiv preprint, arXiv: 2403.05530, 2024

[44] Snowflake AI Research Team. Snowflake arctic: The best LLM for enterprise AI — Efficiently intelligent, truly open[EB/OL]. [2024-08-02]. https://www.snowflake.com/en/blog/arctic-open-efficient-foundation-language-models-snowflake/

[45] The Mosaic Research Team. Introducin DBRX: A new state-of-the-art open LLM[EB/OL]. 2024 [2024-08-02]. https://www.databricks.com/blog/introducing-dbrx-new-state-art-open-llm

[46] Choquette J, Gandhi W, Giroux O, et al. Nvidia A100 tensor core GPU: Performance and innovation[J]. IEEE Micro, 2021, 41(2): 29−35

[47] Choquette J. Nvidia Hopper H100 GPU: Scaling performance[J]. IEEE Micro, 2023, 43(3): 9−17

[48] Ren Jie, Rajbhandari S, Aminabadi R Y, et al. ZeRO-Offload: Democratizing billion-scale model training[C] //Proc of the 2021 USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2021: 551−564

[49] Chen Xiaoming, Chen D Z, Hu X S. MoDNN: Memory optimal DNN training on GPUs[C] //Proc of the 21st Conf & Exhibition on Design, Automation & Test in Europe. Piscataway, NJ: IEEE, 2018: 13−18

[50] Shriram S B, Garg A, Kulkarni P. Dynamic memory management for GPU-based training of deep neural networks[C] //Proc of the 33rd IEEE Int Parallel and Distributed Processing Symp. Piscataway, NJ: IEEE, 2019: 200−209

[51] Ren Jie, Luo Jiaolin, Wu Kai, et al. Sentinel: Efficient tensor migration and allocation on heterogeneous memory systems for deep learning[C] //Proc of the 27th IEEE Transactions Symp on High Performance Computer Architecture. Piscataway, NJ: IEEE, 2021: 598−611

[52] Huang C C, Jin Gu, Li Jinyang. SwapAdvisor: Pushing deep learning beyond the GPU memory limit via smart swapping[C] //Proc of the 25th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2020: 1341−1355

[53] Huang Haiyang, Ardalani N, Sun Anna, et al. Towards MoE deployment: Mitigating inefficiencies in mixture-of-expert (MoE) inference[J]. arXiv preprint, arXiv: 2303.06182, 2023

[54] Eliseev A, Mazur D. Fast inference of mixture-of-experts language models with offloading[J]. arXiv preprint, arXiv: 2312.17238, 2023

[55] Kong Rui, Li Yuanchun, Feng Qingtian, et al. Serving MoE models on resource-constrained edge devices via dynamic expert swapping[J]. arXiv preprint, arXiv: 2308.15030, 2023

[56] Hwang R, Wei Jianyu, Cao Shijie, et al. Pre-gated MoE: An algorithm-system co-design for fast and scalable mixture-of-expert inference[J]. arXiv preprint, arXiv: 2308.12066, 2023

[57] Shen Liang, Wu Zhihua, Gong Weibao, et al. SE-MoE: A scalable and efficient mixture-of-experts distributed training and inference system[J]. arXiv preprint, arXiv: 2205.10034, 2023

[58] Liu Juncai, Wang J H, Jiang Yimin. Janus: A unified distributed training framework for sparse mixture-of-experts models[C] //Proc of the 37th ACM SIGCOMM 2023 Conf. New York: ACM, 2023: 486−498

[59] Kim Y, Lim H, Han D. Scaling beyond the GPU memory limit for large mixture-of-experts model training[C] //Proc of the 41st Int Conf on Machine Learning. New York: PMLR, 2024: 24342−24353

[60] Yi Rongjie, Guo Liwei, Wei Shiyun, et al. EdgeMoE: Fast on-device inference of MoE-based large language models[J]. arXiv preprint, arXiv: 2308.14352, 2023

[61] Xue Leyang, Fu Yao, Lu Zhan, et al. MoE-Infinity: Activation-aware expert offloading for efficient MoE serving[J]. arXiv preprint, arXiv: 2401.14361, 2024

[62] Kamahori K, Gu Yile, Zhu Kan, et al. Fiddler: CPU-GPU orchestration for fast inference of mixture-of-experts models[J]. arXiv preprint, arXiv: 2402.07033, 2024

[63] Zhang Zheng, Xia Yaqi, Wang Hulin, et al. MPipeMoE: Memory efficient MoE for pre-trained models with adaptive pipeline parallelism[C] //Proc of the 37th IEEE Parallel and Distributed Processing Symp. Piscataway, NJ: IEEE, 2023: 167−177

[64] Jain P, Jain A, Nrusimha A, et al. Checkmate: Breaking the memory wall with optimal tensor rematerialization[C/OL] //Proc of the 3rd Machine Learning and Systems. 2020 [2024-08-02]. https://proceedings.mlsys.org/paper_files/paper/2020/file/0b816ae8f06f8dd-3543dc3d9ef196cab-Paper.pdf

[65] Chen Tianqi, Xu Bing, Zhang Chiyuan, et al. Training deep nets with sublinear memory cost[J]. arXiv preprint, arXiv: 1604.06174, 2016

[66] Peng Xuan, Shi Xuanhua, Dai Hulin, et al. Capuchin: Tensor-based GPU memory management for deep learning[C] //Proc of the 25th Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2020: 891−905

[67] Wang Linnan, Ye Jinmian, Zhao Yiyang, et al. SuperNeurons: Dynamic GPU memory management for training deep neural networks[J]. ACM SIGPLAN Notices, 2018, 53(1): 41−53

[68] Kim Y J, Awan A A, Muzio A, et al. Scalable and efficient MoE training for multitask multilingual models[J]. arXiv preprint, arXiv: 2109.10465, 2021

[69] Singh S, Ruwase O, Awan A A, et al. A hybrid tensor-expert-data parallelism approach to optimize mixture-of-experts training[C] //Proc of the 37th Int Conf on Supercomputing. New York: ACM, 2023: 203−214

[70] Heo T, Rashidi S, Man C, et al. Exploring memory expansion designs for training mixture-of-experts models[C/OL] //Proc of the 1st Workshop on Hot Topics in System Infrastructure. 2024 [2024-08-02]. https:// hotinfra23.github.io/papers/hotinfra23-paper4.pdf

[71] Hinton G, Vinyals O, Dean J. Distilling the knowledge in a neural network[J]. arXiv preprint, arXiv: 1503.02531, 2015

[72] Molchanov P, Tyree S, Karras T, et al. Pruning convolutional neural networks for resource efficient inference[J]. arXiv preprint, arXiv: 1611.06440, 2016

[73] Han Song, Mao Huizi, Dally W J. Deep compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding[J]. arXiv preprint, arXiv: 1510.00149, 2016

[74] Dong Zhen, Yao Zhewei, Arfeen D, et al. HAWQ-V2: Hessian aware trace-weighted quantization of neural networks[C] //Proc of the 34th Int Conf on Neural Information Processing Systems. New York: Curran Associates, 2020: 18518−18529

[75] Micikevicius P, Narang S, Alben J, et al. Mixed precision training[J]. arXiv preprint, arXiv: 1710.03740, 2018

[76] Dabre R, Fujita A. Recurrent stacking of layers in neural networks: An application to neural machine translation[J]. arXiv preprint, arXiv: 2106.10002, 2021

[77] Lan Zhenzhong, Chen Mingda, Goodman S, et al. ALBERT: A lite BERT for self-supervised learning of language representations[J]. arXiv preprint, arXiv: 1909.11942, 2020

[78] Rajbhandari S, Li Conglong, Yao Zhewei, et al. DeepSpeed-MoE: Advancing mixture-of-experts inference and training to power next-generation AI scale[C] //Proc of the 39th Int Conf on Machine Learning. New York: PMLR, 2022: 18332−18346

[79] Koishekenov Y, Berard A, Nikoulina V. Memory-efficient NLLB−200: Language-specific expert pruning of a massively multilingual machine translation model[J]. arXiv preprint, arXiv: 2212.09811, 2022

[80] Lu Xudong, Liu Qi, Xu Yuhui, et al. Not all experts are equal: Efficient expert pruning and skipping for mixture-of-experts large language models[J]. arXiv preprint, arXiv: 2402.14800, 2024

[81] Muzio A, Sun A, He C. SEER-MoE: Sparse expert efficiency through regularization for mixture-of-experts[J]. arXiv preprint, arXiv: 2404.05089, 2024

[82] Chowdhury M N R, Wang Meng, Maghraoui K E, et al. A provably effective method for pruning experts in fine-tuned sparse mixture-of-experts[J]. arXiv preprint, arXiv: 2405.16646, 2024

[83] Liu Enshu, Zhu Junyi, Lin Zinan, et al. Efficient expert pruning for sparse mixture-of-experts language models: Enhancing performance and reducing inference costs[J]. arXiv preprint, arXiv: 2407.00945, 2024

[84] Kim Y J, Fahim R, Awadalla H H. Mixture of quantized experts (MoQE): Complementary effect of low-bit quantization and

robustness[J]. arXiv preprint, arXiv: 2310.02410, 2023

[85] Frantar E, Alistarh D. QMoE: Practical sub−1-bit compression of trillion-parameter models[J]. arXiv preprint, arXiv: 2310.16795, 2023

[86] Kim Y J, Henry R, Fahim R, et al. Who says elephants can't run: Bringing large scale MoE models into cloud scale production[J]. arXiv preprint, arXiv: 2211.10017, 2022

[87] Kim Y J, Henry R, Fahim R, et al. FineQuant: Unlocking efficiency with fine-grained weight-only quantization for LLMs[J]. arXiv preprint, arXiv: 2308.09723, 2023

[88] Imani H R, Amirany A, El-Ghazawi T. Mixture of experts with mixture of precisions for tuning quality of service[J]. arXiv preprint, arXiv: 2407.14417, 2024

[89] Gao Zefeng, Liu Peiyu, Zhao Xin, et al. Parameter-efficient mixture-of-experts architecture for pre-trained language models[J]. arXiv preprint, arXiv: 2203.01104, 2022

[90] He S, Fan R Z, Ding Liang, et al. Merging experts into one: Improving computational efficiency of mixture of experts[J]. arXiv preprint, arXiv: 2310.09832, 2023

[91] Zhang Rongyu, Luo Yulin, Liu Jiaming, et al. Efficient deweahter mixture-of-experts with uncertainty-aware feature-wise linear modulation[C] //Proc of the 38th AAAI Conf on Artificial Intelligence. Palo Alto, CA: AAAI, 2024: 16812−16820

[92] Chen Tianyu, Huang Shaohan, Xie Yuan, et al. Task-specific expert pruning for sparse mixture-of-experts[J]. arXiv preprint, arXiv: 2206.00277, 2022

[93] Xue Fuzhao, He Xiaoxin, Ren Xiaozhe, et al. One student knows all experts know: From sparse to dense[J]. arXiv preprint, arXiv: 2201.10890, 2022

[94] Li Jiamin, Jiang Yimin, Zhu Yibo, et al. Accelerating distributed MoE training and inference with Lina[C] //Proc of the 2023 USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2023: 945−959

[95] Nie Xiaonan, Zhao Pinxue, Miao Xupeng, et al. HetuMoE: An efficient trillion-scale mixture-of-expert distributed training system[J]. arXiv preprint, arXiv: 2203.14685, 2022

[96] Hwang C, Cui Wei, Xiong Yifan, et al. Tutel: Adaptive mixture-of-experts at scale[C/OL] //Proc of the 6th Machine Learning and Systems. 2023 [2024-08-02]. https://proceedings.mlsys.org/paper_files/paper/2023/file/5616d34cf8ff73942cfd5aa922842556-Paper-mlsys2023.pdf

[97] He Chaoyang, Zheng Shuai, Zhang A, et al. SMILE: Scaling mixture-of-experts with efficient bi-level routing[J]. arXiv preprint, arXiv: 2212.05191, 2022

[98] Shoeybi M, Patwary M, Puri R, et al. Megatron-LM: Training multi-billion parameter language models using model parallelism[J]. arXiv preprint, arXiv: 1909.08053, 2020

[99] Yao Jinghan, Anthony Q, Shafi A, et al. Exploiting inter-layer expert affinity for accelerating mixture-of-experts model inference[C] //Proc of the 38th IEEE Int Parallel and Distributed Processing Symp. Piscataway, NJ: IEEE, 2024: 915−925

[100] Liu Rui, Kim Y J, Muzio A, et al. Gating Dropout: Communication-efficient regularization for sparsely activated transformers[C] //Proc of the 39th Int Conf on Machine Learning. New York: PMLR, 2022: 13782−13792

[101] He Jiaao, Zhai Jidong, Antunes T, et al. FasterMoE: Modeling and optimizing training of large-scale dynamic pre-trained models[C] //Proc of the 27th ACM SIGPLAN Symp on Principles and Practice of Parallel Programming. New York: ACM, 2022: 120−134

[102] Chen Chang, Li Min, Wu Zhihua, et al. TA-MoE: Topology-aware large scale mixture-of-expert training[C] //Proc of the 36th Int Conf on Neural Information Processing Systems. New York: Curran Associates, 2022: 22173−22186

[103] Zeng Zhiyuan, Xiong Deyi. SCoMoE: Efficient mixtures of experts with structured communication[C] //Proc of the 11th Int Conf on Learning Representations. Amherst, MA: OpenReview. net, 2023: 1−23

[104] Kossmann F, Jia Zhihao, Aiken A. Optimizing mixture of experts using dynamic recompilations[J]. arXiv preprint, arXiv: 2205.01848, 2022

[105] Zheng Bojian, Jiang Ziheng, Yu C H, et al. DietCode: Automatic optimization for dynamic tensor programs[C/OL] //Proc of the 5th Machine Learning and Systems. 2022 [2024-08-02]. https://proceedings.mlsys.org/paper_files/paper/2022/file/f89b79c9a28d4cae22ef9e557d9fa191-Paper.pdf

[106] Zheng Zhen, Pan Zaifeng, Wang Dalin, et al. BladeDISC: Optimizing dynamic shape machine learning workloads via compiler approach[J]. Proceedings of ACM on Management of Data, 2023, 1(3): 1−29

[107] Chen Simin, Wei Shiyi, Liu Cong, et al. DyCL: Dynamic neural network compilation via program rewriting and graph optimization[C] //Proc of the 32nd ACM SIGSOFT Int Symp on Software Testing and Analysis. New York: ACM, 2023: 614−626

[108] Yu Feng, Li Guanglin, Zhao Jiacheng, et al. Optimizing dynamic-shape neural networks on accelerators via on-the-fly micro-kernel polymerization[C] //Proc of the 29th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2024: 797−812

[109] He Jiaao, Qiu Jiezhong, Zeng Aohan, et al. FastMoE: A fast mixture-of-expert training system[J]. arXiv preprint, arXiv: 2103.13262, 2021

[110] Gale T, Narayanan D, Young C, et al. MegaBlocks: Efficient sparse training with mixture-of-experts[C/OL] //Proc of the 6th Machine Learning and Systems. 2023 [2024-08-02]. https://proceedings.mlsys.org/paper_files/paper/2023/file/5a54f79333768effe7e8927bcccffe40-Paper-mlsys2023.pdf

[111] Zheng Ningxin, Jiang Huiqiang, Zhang Quanlu, et al. PIT: Optimization of dynamic sparse deep learning models via permutation invariant transformation[C] //Proc of the 29th Symp on Operating Systems Principles. New York: ACM, 2023: 331−347

[112] Tan S, Shen Yikang, Panda R, et al. Scattered mixture-of-experts implementation[J]. arXiv preprint, arXiv: 2403.08245, 2024

[113] Nie Xiaonan, Miao Xupeng, Cao Shijie, et al. EvoMoE: An evolutional mixture-of-experts training framework via dense-to-

sparse gate[J]. arXiv preprint, arXiv: 2112.14397, 2022

[114] Zhou Yanqi, Lei Tao, Liu Hanxiao, et al. Mixture-of-experts with expert choice routing[C] //Proc of the 36th Int Conf on Neural Information Processing Systems. New York: Curran Associates, 2022: 7103−7114

[115] Zeng Zhiyuan, Guo Qipeng, Fei Zhaoye, et al. Turn waste into worth: Rectifying top-k router of MoE[J]. arXiv preprint, arXiv: 2402.12399, 2024

[116] Lewis M, Bhosale S, Dettmers T, et al. Base layers: Simplifying training of large, sparse models[C] //Proc of the 38th Int Conf on Machine Learning. New York: PMLR, 2021: 6265−6274

[117] Clark A, Casas D D L, Guy A, et al. Unified scaling laws for routed language models[C] //Proc of the 39th Int Conf on Machine Learning. New York: PMLR, 2022: 4057−4086

[118] Liu Tianlin, Puigcerver J, Blondel M. Sparsity-constrained optimal transport[J]. arXiv preprint, arXiv: 2209.15466, 2023

[119] Roller S, Sukhbaatar S, Weston J, et al. Hash layers for large sparse models[C] //Proc of the 35th Int Conf on Neural Information Processing Systems. New York: Curran Associates, 2021: 17555−17566

[120] Zuo Simiao, Liu Xiaodong, Jiao Jian, et al. Taming sparsely activated transformer with stochastic experts[J]. arXiv preprint, arXiv: 2110.04260, 2022

[121] Puigcerver J, Riquelme C, Mustafa B, et al. From sparse to soft mixtures of experts[J]. arXiv preprint, arXiv: 2308.00951, 2023

[122] Yu Ping, Artetxe M, Ott M, et al. Efficient language modeling with sparse all-MLP[J]. arXiv preprint, arXiv: 2203.06850, 2022

[123] Muqeeth M, Liu Haokun, Raffel C. Soft merging of experts with adaptive routing[J]. arXiv preprint, arXiv: 2306.03745, 2023

[124] Hazimeh H, Zhao Zhe, Chowdhery A, et al. DSelect-k: Differentiable selection in the mixture of experts with applications to multi-task learning[C] //Proc of the 35th Int Conf on Neural Information Processing Systems. New York: Curran Associates, 2021: 29335−29347

[125] Ibrahim S, Chen W, Hazimeh H, et al. COMET: Learning cardinality constrained mixture of experts with trees and local search[C] //Proc of the 29th ACM SIGKDD Conf on Knowledge Discovery and Data Mining. New York: ACM, 2023: 832−844

[126] Zhai Mingshu, He Jiaao, Ma Zixuan, et al. SmartMoE: Efficiently training sparsely-activated models through combining offline and online parallelization[C] //Proc of the 2023 USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2023: 961−975

[127] Nie Xiaonan, Miao Xupeng, Wang Zilong, et al. FlexMoE: Scaling large-scale sparse pre-trained model training via dynamic device placement[J]. Proceedings of ACM on Management of Data, 2023, 1(1): 1−19

[128] Du Zhixu, Li Shiyu, Wu Yuhao, et al. SiDA: Sparsity-inspired data-aware serving for efficient and scalable large mixture-of-experts models[C/OL] //Proc of the 7th Machine Learning and Systems. 2024 [2024-08-02]. https://proceedings.mlsys.org/paper_files/paper/2024/file/698cfaf72a208aef2e78bcac55b74328-Paper-Conference.pdf

[129] Shazeer N, Cheng Youlong, Parmar N, et al. Mesh-Tensorflow: Deep learning for supercomputers[C] //Proc of the 32nd Int Conf on Neural Information Processing Systems. New York: Curran Associates, 2018: 10435−10444

[130] Rajbhandari S, Rasley J, Ruwase O, et al. ZeRO: Memory optimizations toward training trillion parameter models[C/OL] //Proc of Int Conf for High Performance Computing, Networking, Storage and Analysis. Piscataway, NJ: IEEE, 2020 [2024-09-10]. https://dl.acm.org/ doi/abs/10.5555/3433701.3433727

[131] Kosson A, Chiley V, Venigalla A, et al. Pipelined backpropagation at scale: Training large models without batches[C/OL] //Proc of the 4th Machine Learning and Systems. 2021 [2024-08-02]. https://proceedings.mlsys.org/paper_files/paper/2021/file/0c8abcf158ed12d0dd94480681186fda-Paper.pdf

[132] Huang Yanping, Cheng Youlong, Bapna A, et al. GPipe: Efficient training of giant neural networks using pipeline parallelism[C] //Proc of the 33rd Int Conf on Neural Information Processing Systems. New York: Curran Associates, 2019: 103−112

[133] Narayanan D, Harlap A, Phanishayee A, et al. PipeDream: Generalized pipeline parallelism for dnn training[C/OL] //Proc of the 27th ACM Symp on Operating Systems Principles. New York: ACM, 2019 [2024-09-10]. https://dl.acm.org/doi/abs/10.1145/3341301.3359646

[134] Narayanan D, Phanishayee A, Shi Kaiyu, et al. Memory-efficient pipeline-parallel DNN training[C] //Proc of the 38th Int Conf on Machine Learning. New York: PMLR, 2021: 7937−7947

[135] Fu Yichao, Yuhao Qing, Zhao Shixiong, et al. AMPipe: Accelerating MoE model training with intra-block pipelining[EB/OL]. 2024 [2024-08-02]. https://openreview.net/pdf?id=yLgr02IsXY

[136] Jiang Chenyu, Tian Ye, Jia Zhen, et al. Lancet: Accelerating mixture-of-experts training via whole graph computation-communication overlapping[J]. arXiv preprint, arXiv: 2404.19429, 2024

[137] Shi Shaohuai, Pan Xinglin, Chu Xiaowen, et al. PipeMoE: Accelerating mixture-of-experts through adaptive pipelining[C/OL] //Proc of the 2023 IEEE Conf on Computer Communications. Piscataway, NJ: IEEE, 2023 [2024-09-10]. https://ieeexplore.ieee.org/abstract/document/10228874

[138] Aminabadi R Y, Rajbhandari S, Awan A A, et al. DeepSpeed-Inference: Enabling efficient inference of transformer models at unprecedented scale[C/OL] //Proc of Int Conf for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2022 [2024-09-10]. https://dl.acm.org/doi/abs/10.5555/3571885.3571946

[139] Valiant L G. A bridging model for parallel computation[J]. Communications of the ACM, 1990, 33(8): 103−111

[140] Narayanan D, Shoeybi M, Casper J, et al. Efficient large-scale language model training on GPU clusters using Megatron-LM [C/OL] //Proc of Int Conf for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2021 [2024-09-

10]. https://dl.acm.org/doi/ abs/10.1145/3458817.3476209

[141] Wang Guanhua, Qin Heyang, Jacobs S A, et al. ZeRO++: Extremely efficient collective communication for giant model training[J]. arXiv preprint, arXiv: 2306.10209, 2023

[142] Rajbhandari S, Ruwase O, Rasley J, et al. ZeRO-infinity: Breaking the GPU memory wall for extreme scale deep learning[C/OL] //Proc of Int Conf for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2021 [2024-09-10]. https://dl.acm.org/doi/ abs/10.1145/3458817.3476205

[143] Jia Zhihao, Lin S, Qi C R, et al. Exploring hidden dimensions in parallelizing convolutional neural networks[C] // Proc of the 35th Int Conf on Machine Learning. New York: PMLR, 2018: 2279−2288

[144] Zheng Lianmin, Li Zhuohan, Zhang Hao, et al. Alpa: Automating inter-and intra-operator parallelism for distributed deep learning[C] //Proc of the 16th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2022: 559−578

[145] Li Zhuohan, Zheng Lianmin, Zhong Yinmin, et al. AlpaServe: Statistical multiplexing with model parallelism for deep learning serving[C] //Proc of the 17th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2023: 663−679

[146] Jia Zhihao, Zaharia M, Aiken A. Beyond data and model parallelism for deep neural networks[C/OL] //Proc of the 2nd Machine Learning and Systems. 2019 [2024-08-02]. https://proceedings.mlsys.org/paper_files/paper/2019/file/b422680f3db0986ddd7f8f126baaf0fa-Paper.pdf

[147] Artetxe M, Bhosale S, Goyal N, et al. Efficient large scale language modeling with mixtures of experts[J]. arXiv preprint, arXiv: 2112.10684, 2022

[148] Zhao Yanli, Gu A, Varma R, et al. Pytorch FSDP: Experiences on scaling fully sharded data parallel[J]. arXiv preprint, arXiv: 2304.11277, 2023



Shi Hongzhi, born in 1988. Master. Member of CCF. His main research interests include computer architecture and deep learning.

史宏志, 1988 年生. 硕士. CCF 会员. 主要研究方向为计算机体系结构、深度学习.



Zhao Jian, born in 1987. Master. His main research interests include computer architecture, and deep learning.

赵　健, 1987 年生. 硕士. 主要研究方向为计算机体系结构、深度学习.



Zhao Yaqian, born in 1981. PhD, senior engineer. Senior member of CCF. Her main research interests include computer architecture and artificial intelligence.

赵雅倩, 1981 年生. 博士, 高级工程师. CCF 高级会员. 主要研究方向为计算机体系结构、人工智能.



Li Ruyang, born in 1990. PhD, senior engineer. Senior member of CCF. Her main research interests include deep reinforcement learning, autonomous driving, and scenario-oriented AI acceleration.

李茹杨, 1990 年生. 博士, 高级工程师. CCF 高级会员. 主要研究方向为深度强化学习、自动驾驶、面向场景的 AI 加速.



Wei Hui, born in 1987. PhD, senior engineer. Member of CCF. His main research interests include virtual reality and 3D vision.

魏　辉, 1987 年生. 博士, 高级工程师. CCF 会员. 主要研究方向为虚拟现实、3 维视觉.



Hu Kekun, born in 1987. PhD, senior engineer. Member of CCF. His main research interests include graph computing and graph deep learning.

胡克坤, 1987 年生. 博士, 高级工程师. CCF 会员. 主要研究方向为图计算、图深度学习.



Wen Dongchao, born in 1979. Master, professor. Member of IEEE and CCF. His main research interests include computer vision and trustworthy deep learning.

温东超, 1979 年生. 硕士, 正高级工程师. IEEE 会员, CCF 会员. 主要研究方向为计算机视觉、可信深度学习.



Jin Liang, born in 1986. Master. His main research interests include computer vision and multimodalities.

金　良, 1986 年生. 硕士. 主要研究方向为计算机视觉、多模态.