ISSN 1007-130X
CN 43-1258/TP

计算机工程与科学
Computer Engineering & Science

第 48 卷第 10 期 2026 年 10 月
Vol. 48, No. 10, Oct. 2026

文章编号:1007-130X(2026)10-1854-19

面向成本效益的超大规模训练集群建设*

焦方正[1],张梦豪[1],王士诚[3],贾彦民[2],张　琰[2],张铭俊[2],胡效赫[2],胡春明[1]

(1.北京航空航天大学软件学院,北京 100191;2.上海基流科技有限公司,上海 200235;
3.清华大学网络科学与网络空间研究院,北京 100084)

摘　要:大语言模型已在众多应用领域展现出卓越的能力,但模型规模的持续增长,使训练成本成为其发展过程中的关键挑战。围绕大规模生产级大模型训练集群方案 Galaxy,介绍了其在设计、优化和运维等方面的先进性。首先建立了一个训练成本分析模型,并厘清低成本高效训练的核心因素,以此为基础,从全栈角度改善训练成本效益。在基础设施层面,将设备选型与网络架构跨层设计进行优化,在控制成本的同时最大化计算与通信效率。在软件层面,联合集合通信库和训练框架进行跨层深度优化,缓解性能瓶颈并提升训练吞吐量和稳定性,提出智能容错与运行稳定性保障技术。在运维方面构建细粒度全栈监控系统与标准化诊断流程,用于快速定位性能异常。同时还结合超大规模大模型训练集群的实践,总结出若干经验,为学术界和工业界提供参考。

关键词:成本效益;网络架构;集合通信;集群建设;大语言模型训练;分布式系统;

中图分类号:TP18　　**文献标志码**:A

doi:10.3969/j.issn.1007-130X.2026.10.015

Cost-effective large-scale LLM training cluster construction

JIAO Fangzheng[1], ZHANG Menghao[1], WANG Shicheng[3], JIA Yanmin[2],
ZHANG Yan[2], ZHANG Mingjun[2], HU Xiaohe[2], HU Chunming[1]

(1. School of Software, Beihang University, Beijing 100191;
2. Infrawaves Inc., Shanghai 200235;
3. Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China)

Abstract: Large language models have proven highly capable across a wide range of applications. However, as these models continue to scale up, the training-cost them has become one of the main obstacles to their further development. This paper describes Galaxy scheme, a production-grade cluster we built for large-scale LLM training, and covers how it was designed and optimized and how it is operated. We first build a training-cost model and uses it to identify the factors that really determine whether training remains low-cost and efficient. Guided by this model, we improve cost efficiency from a full-stack perspective. At the infrastructure level, device selection and the network architecture are co-designed across layers so that computation and communication efficiency are maximized without driving up cost. At the software level, we jointly optimize the collective communication library and the training framework across layers, alleviating performance bottlenecks and improving both training throughput and stability, and we introduce techniques for intelligent fault tolerance and stable operation. On the operations side, a fine-grained full-stack monitoring system and a standardized diagnosis procedure allow us to pinpoint performance anomalies quickly. Finally, we draw on our experience with clusters of this

* **收稿日期**:2026-01-13;**修回日期**:2026-03-17
基金项目:国家重点研发计划(2024YFB4505604);国家自然科学基金(62402025)
通信作者:张梦豪(zhangmenghao@buaa.edu.cn)
引用格式:焦方正,张梦豪,王士诚,等.面向成本效益的超大规模训练集群建设[J].计算机工程与科学,2026,48(10):1854-1872.
Citation: JIAO Fangzheng, ZHANG Menghao, WANG Shicheng, et al. Cost-effective large-scale LLM training cluster construction[J]. Computer Engineering & Science, 2026, 48(10): 1854-1872.

scale to distill a number of practical lessons that we hope will be useful to both the research community and industry.

Key words: cost-effectiveness; network architecture; collective communication; cluster construction; large language model training; distributed systems

1 引言

近年来,随着模型规模和训练数据量的持续增长,缩放定律(Scaling Law)[1]的有效性不断得到验证。从2020年发布的含1 750亿个参数、3 000亿个词元训练数据的GPT-3,到2023年发布的含1.8万亿个参数、使用13万亿个词元训练数据的GPT-4,再到2025年发布的含1.7万亿参数、使用85万亿词元训练数据的MoE(Mixture-of-Experts)模型Grok 4,模型能力显著增强,计算需求也随之爆炸式增长。为了满足如此巨大的计算需求,业界逐渐开始进行数千卡甚至万卡的超大规模分布式训练集群的规划与建设[2,3]。然而,在搭建和运维这样一个超大规模智算集群时,成本效益成为最关键的考量因素。

实现超大规模训练集群的成本效益优化还存在3大挑战。

第1个挑战来自高昂的硬件成本:训练集群依赖高端GPU和网络硬件,这些硬件不仅价格昂贵,而且供应紧张。建设初期本文调研了市场报价,NVIDIA H100的价格约2.5万美元[4],而Broadcom Tomahawk 5交换机芯片的价格约4万美元[5]。照此测算,建设一个包含数千个GPU和数百台交换机的算力集群总成本可能会超过3亿美元。另一部分成本来自于网络:为了实现高效的分布式训练,必须设计高带宽、无阻塞的网络架构。一旦采用不恰当的协议设计或流量调度方案,通信便会成为系统瓶颈,昂贵的GPU资源随之被闲置浪费。

第2个挑战在于训练框架的资源利用率仍有较大的提升空间。现阶段,模型浮点计算利用率MFU(Model FLOPs Utilization)很低,通常为20%~45%[6-8],这意味着大量的资源浪费,训练成本也因此抬高。造成这种低效情况的原因是多方面的:初始化过程时间长,流水线并行中气泡占比高,以及集合通信库和训练框架协同不足,等。这些效率瓶颈直接拉升了训练任务的运行成本。

第3个挑战是系统可靠性,以及故障能否被快速诊断和恢复。集群组件种类繁多、规模庞大,训练又通常持续数月,故障的发生频率和总次数都明显上升。以Meta[8]为例,其用1.6万卡训练Llama3的54天周期里,出现了419次意料之外的中断,平均无故障时间只有3 h左右。本文千卡环境的生产数据也呈现类似趋势:平均无故障时间为20~70 h,每月发生的各类故障可达10~20次,如图1所示。故障的发现与定位常常要耗费大量时间,整体资源利用率也因此被拉得更低。这也说明,一套高效的监控体系配合快速恢复机制,是保障训练稳定运行的必要条件。

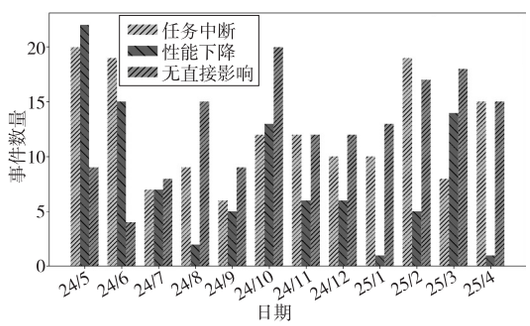


图1 生产环境中每月的硬件故障统计

Figure 1 Monthly statistics of hardware failures in production environment

尽管已有一些研究工作[2,9-13]针对特定环节提出优化方案,但它们大多只关注局部的组成部分,而不是整体的成本效益。Gemini 2.5[9,10]采用全光交换机进行组网规划,同时优化了弹性伸缩的最小粒度应对训练中出现的硬件故障,缩短重启时间,但没有对新架构的成本进行说明。MegaScale[2]为更高的MFU优化了训练算法和框架,但没有优先考虑降低每个词元的成本。Meta[11]和AlibabaHPN[12]更侧重于网络优化,如网络拓扑设计、拥塞控制协议和路由。Fire-flyer[13]提出了一个软硬件协同设计框架,在PCIe服务器上运行以降低成本。这些方案虽然都能带来一定程度的优化,但尚未形成完整的、以成本效益为核心的大规模智算集群全栈优化的解决方案。

本文提出了Galaxy,一个旨在降低大规模集群的成本和提高效益的LLM集群方案。基于多年来为众多客户成功建设和运维大型智算集群和AI云平台的经验,本文详细介绍了LLM训练集群的设计、优化和运维实践。具体将详细介绍以下内容:

1)提出了一个成本效益模型,该模型能够系统地描述建设和运维大规模智算集群的成本收益。

通过对影响每词元训练成本的因素进行定量建模，确定了整个基础设施、软件栈和运维的成本效率优化原则。

2)在成本模型的指导下，对基础设施、软件栈和监控运维进行架构设计和系统优化。在基础设施方面，对 GPU 服务器、交换机和网络架构分别进行了特殊设计，以在最大效率发挥硬件性能的前提下降低资金成本。在训练的软件栈方面，跨层联合优化了集合通信库和训练框架，从而在大规模集群上最大化训练效率。在集群可靠性和故障定位方面，部署了全栈监控系统、容错和恢复机制，以及制定故障定位标准操作流程 SOP(Standard Operating Procedure)，以实现高效的故障诊断和恢复，以及高稳定性。相应的评估都证明了在成本效益模型指导下实现的成本节约或性能提高。

3)最后分享了在部署、调优和监控 LLM 训练集群方面的见解和经验，为人工智能基础设施社区提供实践指导。

综合来看，Galaxy 已经部署和运行了一年多，在 1 万多张 Hopper 架构 GPU 上实现了 42%以上的 MFU，同时降低了 19%的成本。

2 模型训练成本模型

2.1 大模型训练架构

大语言模型的参数规模庞大，且需在海量数据集上完成训练，一次预训练或微调往往要动用成千上万块 GPU 协同运算。为支撑这种规模下的高效训练，Megatron[14] 和 DeepSpeed[15] 等主流的训练框架通常组合多种混合并行策略来统一调度所有 GPU 资源。

1)张量并行 TP(Tensor Parallelism)。将矩阵按照张量维度划分并分布至多个 GPU，由此降低单卡的内存需求、提高运算效率。但是，TP 需要在多个 GPU 间交换中间的计算结果，单次传输的数据量和传输次数都明显高于大多数并行策略，整体传输量高出其他并行策略 30 倍以上[16]，通常只在支持高速互联(如 NVLink)的节点内部使用，以提升带宽利用率和传输的效率。

2)流水线并行 PP(Pipeline Parallelism)。通过将模型层划分为若干顺序阶段，并将各阶段分配给不同的 GPU，降低了对单卡 GPU 显存的需求，使得更大规模的模型训练成为可能。但由于阶段之间采用流水线依次执行的方式，非稳定计算阶段存在流水线空泡，大大降低了计算效率，因此需要精心设计计算与通信的重叠策略，以最大化硬件资源利用率。

3)数据并行 DP(Data Parallelism)。将训练数据分布到多个 GPU 上，每个 GPU 保留一份完整的模型副本，并通过 AllReduce 操作同步梯度。尽管 DP 的实现较为简单，但由于模型副本的存在，通信开销较大，并受到 GPU 显存容量的限制。

4)序列并行 SP(Sequence Parallelism)。将输入序列切分到多个 GPU 上，并需在启用 TP 的前提下开启。SP 的思路是缩短部分计算步骤的序列长度，进而压低激活值的存储占用，使长序列能够在较低的单卡内存开销下并行处理。

5)上下文并行 CP(Context Parallelism)。将长输入序列直接进行切分，让注意力等操作能够跨多个 GPU 并行执行。不过，注意力机制要综合每个词元的上下文，一旦启用 CP，就得在各切分之间频繁交换注意力权重与隐藏状态，通信负担相当重；加之 Mask 机制的存在，各词元的计算量并不均衡，还需引入均衡策略来改善整体效率。

6)专家并行 EP(Expert Parallelism)。将模型的前馈神经网络部分进行切分，每一部分作为一个“专家”，每个专家负责更新模型参数的一部分。在训练过程中，专家通过门控机制动态选择参与计算的部分，从而提高计算效率和硬件资源的利用率。通过将模型参数分布到多个专家并行计算，EP 大幅降低了模型计算量，选择性地更新参数矩阵的一部分。由于专家对输入词元具有选择性，专家并行的引入使得 GPU 之间需要进行 AlltoAll 的数据交换，因此专家并行存在负载不均和显著“多打一”的拥塞特点。

通过组合上述多种并行策略，可以实现将大数据量和大参数量的 LLM 模型训练任务分布至大规模 GPU 集群中。划分词元以进行批次并行处理，使得大型语言模型能够从海量数据中有效学习。这种混合并行的设计对于实现现代大模型训练的可扩展性与计算效率至关重要。

2.2 为何每词元成本至关重要

对于训练即服务 TaaS(Training as a Service)，本文引入“每词元训练成本”作为衡量训练效率与经济性之间关系的标准化指标。

MFU 常被广泛用于衡量训练效率，它反映了实际达到的浮点计算量与理论峰值浮点计算量的比值，却没有把硬件费用、运维成本这类经济因素纳入考量[7,8,17]。举例来说，在低成本硬件平台上运行一套优化过的软件栈，可能获得与高端系统相当的 MFU 值，但其资本开销却大幅降低，可见仅

凭 MFU,很难全面评估 LLM 训练服务的性价比。

实际上,训练成本可以用多种形式进行描述,例如单位时间成本和每参数成本等。单位时间成本是指将经济开销(如硬件折旧、电力消耗等)按时间摊销得到的资源使用成本,摊销周期一般取 4～5 年。该指标在许多基础设施即服务 IaaS(Infrastructure as a Service)场景中得到广泛应用,但在训练服务场景中缺乏足够的训练任务细节,如训练时长、参数数量和词元数量等,难以用于预测特定任务的成本。

每参数成本刻画了模型复杂度与成本之间的关系。但参数数量常常与模型内部的层数、结构等复杂性紧密耦合,参数更多的模型并不一定带来线性增长的训练成本;这就使得每参数成本既难以收敛成统一的评价标准,也无法切实指导效率改进。

结合本文的研究和生产集群的经验,大多数训练任务通常集中使用少数固定的模型架构,真正差别明显的是训练数据集,它们在来源和规模上往往相差很大。

正是基于上面的分析,本文把"每词元成本"作为核心评估指标。它不仅便于在不同训练数据集之间直接比较,还能真实反映训练方案的端到端综合效率,把底层硬件、训练框架到运行调度的全链路优化程度都囊括了进来。

2.3 成本效益模型构建

本文提出了一个由 N 个 GPU 节点组成的 LLM 训练集群的成本效益模型。模型分析中用到的变量如表 1 所示。

表 1 成本效益模型符号说明

Table 1 Notation of cost-benefit model

符号	描述
S	总浮点运算次数
M	模型参数大小
T	词元数量
P	稳定阶段 MFU
F	每个 GPU 的最大 MFU
t_i	初始化时间
α	性能表现因子
f_c	检查点频率
t_c	检查点持续时间
N	GPU 数量
f	单故障频率
t_d	单故障平均诊断时间
t_r	单故障平均恢复时间

对于指定模型参数大小为 M 和词元数量为 T 的模型训练任务,每个词元的训练成本 $costPerToken$ 可以表示为式(1)所示:

$$costPerToken=\frac{costPerTimeUnit\times t}{T} \tag{1}$$

其中,t 为总训练时间,$costPerTimeUnit$ 为单位时间的成本。

训练成本主要来源于硬件设备成本 $CapEx$(Capital Expenditure)和运维成本 $OpEx$(Operational Expenditure)。在硬件设备中,GPU 服务器通常占总 $CapEx$ 的 70%～80%,网络设备占剩余的 20%～30%。运维成本主要由单位时间内的功耗和研发成本组成。单位时间的成本可由以上成本除以时间得到,如式(2)所示:

$$costPerTimeUnit=\frac{CapEx}{DepreciationTime}+OpEx \tag{2}$$

根据本文的经验,即使按折旧时间 $DepreciationTime$(4～5 年)摊销,设备的单位成本也远高于运维成本。总训练时间 t、总浮点运算次数 S 和底层 GPU 节点提供的最大 MFU F 与训练时间内模型平均浮点计算利用率 $avgMFU$[7]有关。F 可以用每个 GPU 节点的最大 MFU 之和 $F=\sum_{i=1}^{N}F_i$ 表示。由此可以得到训练总时间 t 与 S,F,$avgMFU$ 三者的关系,如式(3)所示:

$$t=\frac{S}{avgMFU\times F} \tag{3}$$

总浮点运算次数 S 可以近似为 k 倍的模型大小 M 与词元数量 T 的乘积[18],通常 $k\approx 6$,如式(4)所示:

$$S=k\times M\times T \tag{4}$$

整个训练时间包含多个不同 MFU 的时间阶段。在初始化时间 t_i 中,由于框架设置和数据加载,MFU 接近于 0。在检查点持续时间,训练一般保持稳定的 MFU,记为 P。由于还要定期执行检查点保存的工作,本文将 GPU 状态保存到远端的共享存储中,用于模型评估或故障恢复。在检查点保存过程中,MFU 下降到一个更低的级别,表示为 $\alpha P(0<\alpha<1)$,其中 α 是性能下降因子。假设检查点频率为 f_c,则浪费的时间为 $(1-\alpha)t_c\times f_c$[19]。此外,大规模集群故障频发,一旦发生故障,训练任务被迫终止直到故障诊断完成,随后加载最新的检查点并重新初始化训练框架。假设有 N 张 GPU 的集群,单个 GPU 的故障频率为 f。

这种恢复过程会导致以下3个因素的时间浪费：

1)用于检测、定位和解决故障的诊断时间。假设单个故障的平均诊断时间为 t_d，则总体诊断时间为 $t\times N\times f\times t_d$。

2)从最后一个检查点开始浪费的GPU时间。假设故障在检查点之间均匀分布[20]，则平均浪费的时间可以表示为 $t\times N\times f/2f_c$。

3)每次故障加载最新检查点的恢复时间为 $N\times f\times t_r$[19]。综上所述，$avgMFU$ 可以表示为不同训练阶段的MFU的加权平均值 $avgMFU$，如式(5)所示：

$$avgMFU=\left[1-\frac{t_i}{t}-(1-\alpha)f_c\times t_c-N\times f\left(t_d+\frac{1}{2f_c}+t_r\right)\right]P \tag{5}$$

根据式(1)—式(5)，可以推导出 $costPerToken$，如式(6)所示：

$$costPerToken=\frac{costPerTimeUnit\left(\frac{k\times M\times T}{P\times F}+t_i\right)}{\left[1-(1-\alpha)f_c\times t_c-N\times f\left(t_d+\frac{1}{2f_c}+t_r\right)\right]T} \tag{6}$$

2.4 优化原则

借助成本模型，本文从全栈角度做了一次逐层剖析，弄清设计、优化和运维方案在不同层次上如何左右每词元的训练成本。最底层是基础设施，由GPU服务器和网络设备(交换机、网卡、光模块和线缆等)构成，为训练框架提供最基本的计算与通信能力。服务器硬件这一侧，单GPU的浮点运算能力、机内通信能力和内存带宽都会作用于MFU，例如NVIDIA GPU服务器上的NVLink就为机内卡间提供了高速带宽。服务器之间的网络，则由拓扑、路由和传输协议的设计决定机间通信能力，并由此影响整体训练效率。

在基础设施之上，训练所使用的软件栈对效率的影响同样明显。集合通信库要尽量把所有硬件的带宽发挥出来，以此压低通信开销。在训练框架层面，并行方案、重叠机制等策略决定了训练阶段能维持多高的稳定MFU，即 P 的取值。软件栈整体的实现质量最终都会体现在 $avgMFU$ 上：更短的初始化时间 t_i、检查点时间 t_c 和恢复时间 t_r 可以最大限度地减少非计算时间并提高 $avgMFU$。

稳定的MFU固然依赖基础设施和训练软件栈的精心设计与优化，但运维监控对于稳住系统、减少故障停机同样不可忽视。细粒度监控能加快故障诊断、帮助定位故障组件，并最小化 t_d，提高集群的使用效率。作为容错的基础，检查点机制应该尽可能异步和并行，使其对 $avgMFU$ 的影响最小化。检查点频率 f_c 和高效的状态保存/加载机制(α，t_c 和 t_r)有助于最小化性能损失。通过定义高效的故障诊断标准操作程序SOP将有助于减少故障平均诊断时间 t_d，更及时地发现和定位分布式集群故障，从而显著提高 $avgMFU$。

基于以上分析，本文提出了以下优化原则：

1)采用多个供应商的标准化商用硬件以降低资金成本实现高性能。

2)高度定制的多层协同设计的训练框架提供高稳态MFU并降低非计算时间开销。

3)细粒度全栈监控，快速检测并排除定位故障节点。

Galaxy的整体架构如图2所示。其基础设施由GPU服务器和网络硬件组成，为整个集群提供计算和通信能力。在基础设施之上，训练软件栈执行LLM训练任务，该任务包括训练框架PyTorch、底层集合通信库NCCL(NVIDIA Collective Communication Library)以及各种CUDA库。除了训练平台外，Galaxy还部署了一个运维操作平台，用于监控和管理全套软硬件环境，以实现全域状态监测、快速故障定位和高效稳定训练。

3 基础设施

3.1 优化指导

3.1.1 服务器

GPU服务器作为LLM训练的基础设施，通常配备双路CPU和8卡GPU，如图3所示。在硬件选型方面，本文使用AMD CPU代替Intel CPU以压低非GPU成本。为提升通信效率，GPU服务器内部通过NVLink和NVSwitch实现卡间的高速互联，同时将每对GPU与网卡部署在同一PCIe Switch下，启用GDR(GPU Direct RDMA)功能实现跨服务器间的高速数据传输。

为了提高服务器内部通信性能，本文启用NVLS(NVLink SHARP)[21]技术来提升机内的集合通信性能，并在上层的集合通信库里进行针对性优化。PCIe参数也会显著影响服务器内部通信性能。在实践中，本文关闭PCIe访问控制服务ACS(Access Control Services)来避免GPU-RNIC

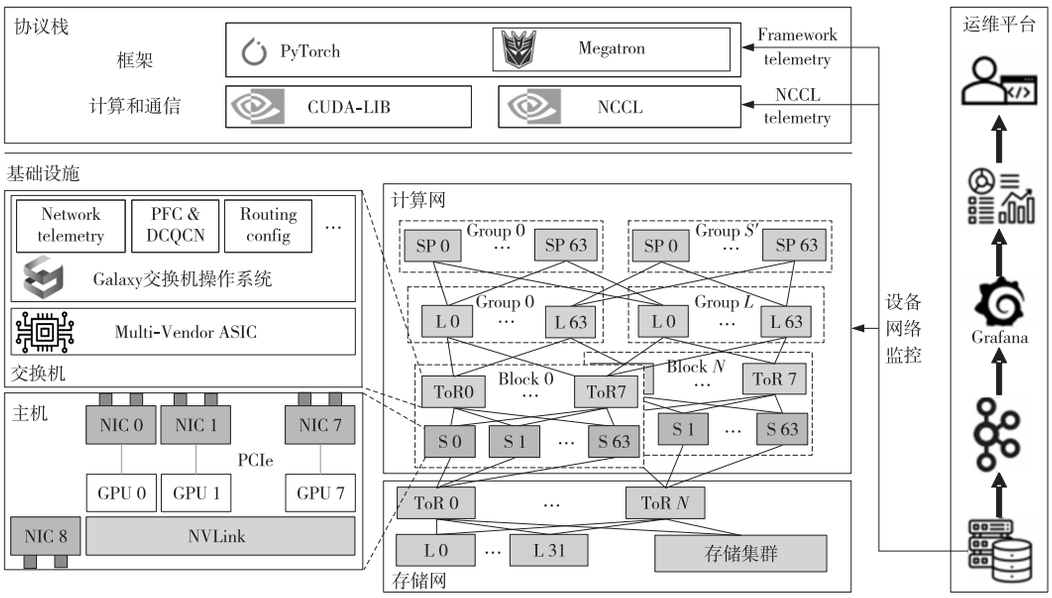


图 2 整体架构 Galaxy

Figure 2 Overview architecture of Galaxy

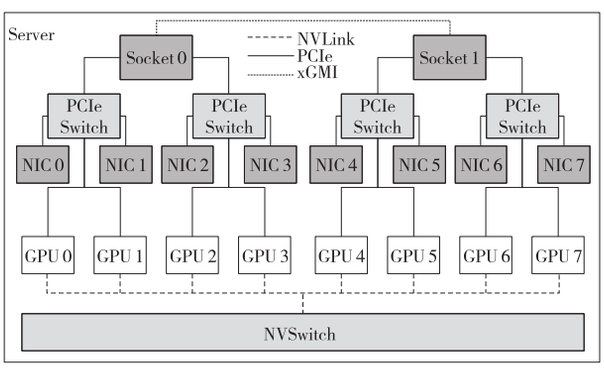


图 3 机内拓扑

Figure 3 Intra-node topology

通信的数据通过 CPU 的 PCIe 根节点。同时关闭 PCIe 地址转换服务 ATS(Address Translation Services)来避免虚拟化带来的性能开销。在 LLM 训练场景中,GPU 服务器通常只被单个 LLM 训练任务占用,虚拟化功能可以完全关闭。此外,本文还启用了 PCIe 乱序功能,允许 PCIe 上的传输无序,并适当设置 *MaxReadReq* 提高 PCIe 上的最大字节数。本文还设置并启用了 *ADVANCED_PCI_SETTINGS*(PCIe 高级功能),并设置 *MAX_ACC_OUT_READ*(最大累计未完成读请求字节),以便在发出更多读请求的同时缩短链路空闲时间。通过对服务器内部硬件的细致调校,本文以较低的成本获得了最先进的通信性能。

3.1.2 网络交换机

以 NVIDIA[22] 和 ARISTA[23] 为代表的传统商用交换机,网络方案成熟,技术支持也完善,但可采购成本偏高,用在大规模场景中就成了一处明显短板。这类黑盒设备走的是闭源软硬件协议栈,后期想升级网络功能会遇到很高的技术壁垒;同时不同厂商的交换机难以统一管理,互操作性也不好。针对这些弊端,本文提出"开放硬件+定制软件"的交换机设计范式,既压低了成本,又保留了操作上的灵活度。

Galaxy 采用多供应商白盒交换芯片,配上基于 SONiC(Software for Open Networking in the Cloud)[24] 深度定制的网络操作系统实现硬件抽象统一,并据此自研了 Mercury 白盒交换机。相比原生 SONiC,Mercury 交换机操作系统提供了统一的软件定义接口,并在多供应商交换芯片上实现了功能灵活性,屏蔽了底层硬件的复杂性。如图 4 所示,该交换机把硬件、驱动、SDK(Software Development Kit)、协议以及各类网络应用集成到一起,并在架构上让硬件与软件解耦,借由交换机抽象接口 SAI(Switch Abstraction Interface)和驱动程序开发框架 DDF(Driver Development Framework)对外提供软件定义网络。

通过以上操作,Mercury 交换机比普通交换机的成本降低了 20%~40%,并且交货周期短、产能充足。Mercury 的标准软硬件接口屏蔽了底层硬件的复杂性,实现了对不同硬件的统一管理。因此,Mercury 支持灵活的软件定义网络操作,能够快速迭代网络功能。在商品交换机上支持一项新功能通常需要半年多的时间,而 Mercury 可以在 2~3 周内完成新功能的开发、测试与部署。

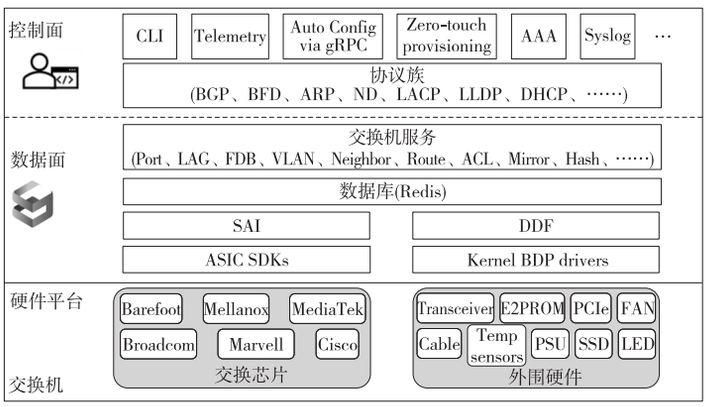


图 4 白盒交换机架构

Figure 4 Architecture of white-box switch

3.1.3 光模块

模块成本与技术的成熟度、产能息息相关。400 Gbit/s 光模块在智算中心建设的初期,产能低且需求旺盛,价格居高不下。如果采用单端口 400 Gbit/s 网卡,不仅光模块成本高,而且交付周期长,集群建设成本效益下降。为在实现高带宽和快速交付的同时降低集群在网络设备方面的成本,本文采用 200 Gbit/s 的光模块与双口网卡作为网卡和模块的选型,以此来提供更低的成本和更短的交付周期。

3.2 网络拓扑

3.2.1 组网方案

从成本效益和功能性出发,本文基于 200 Gbit/s 的模块和双端口网卡提出了一个双口网络架构,以实现低成本、高性能和高稳定性的集群建设方案。一种直接的解决方案是采用端口聚合的方式将 2 个 200 Gbit/s 端口连接到 ToR(Top of Rack)交换机[12,25-27]上的 LAG 端口,其中 2 个端口之间启用自动负载平衡,以获得相当于 400 Gbit/s 的聚合性能[28]。然而,这种依赖网卡端口哈希策略实现端口流量均衡的方案存在随机性,同时由于 GPU 通信之间的低熵流量模式,不均衡的概率会更高,一旦不平衡就会导致聚合吞吐下降。因此,Galaxy 直接采用 2 个非绑定(non-bond)端口,通过一分二线缆连接到交换机端口,再配合增强的集合通信库,实现经济且高效的 400 Gbit/s 网络架构。

首先,Galaxy 采用训练流量和存储等流量的分离解耦方案,如图 5 所示。计算网构建了一个双端口非绑定网络架构专门用于训练通信流量。存储网负责管理、存储等其他业务,支持 TCP 和 RDMA 混合流量。一般情况下,同步与建链等信息采用 TCP,对性能要求高的存储流量采用 RDMA 通信,并设置高优先级队列,尽可能避免流量冲突。

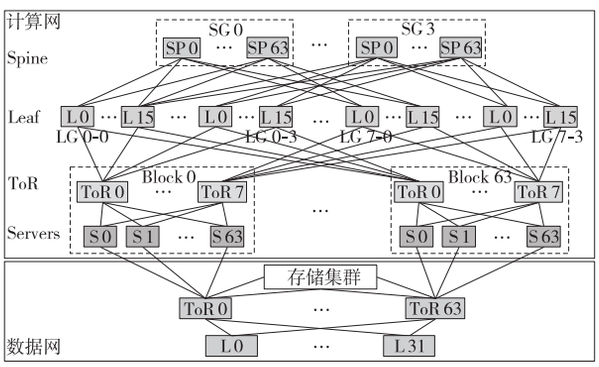


图 5 网络架构设计

Figure 5 Design of network architecture

计算网采用的 RDMA 网络为 3 层多轨道拓扑[29]。不同主机上的相同位置网卡属于同一轨道,连接到同一个 ToR 交换机。多轨拓扑可以保证具有相同索引的 GPU 之间的通信仅为 1 跳或 3 跳。为了实现任意 2 个 GPU 之间的无阻塞通信,每层交换机的下行和上行带宽都采用无阻塞设计,保证 1∶1 收敛比。将每台 ToR 交换机、同一层的叶(Leaf)交换机、同一层的脊(Spine)交换机配置为一个 BGP 自治域,并通过 BGP 声明建立彼此之间的路由[30]。

GPU 服务器中网卡数量通常与 GPU 的数量相对应,一个组网架构支持的网卡数量由交换机的端口密度和网络架构决定。交换机的端口密度越高,支持的网卡数量越多;网络层级越高,支持的网卡越多,但转发跳数和延迟也会增加,路由冲突的可能性也会增加。

假设每个 GPU 服务器有 m 个网卡,每个交换机同质且有 p 个接口。如图 5 所示,本文将所有服务器部署在一个大平面中,并通过分组叶交换机进行连接。对于每个叶群,第 i 个 p/m 交换机都

连接到同一脊交换机组,因此只需要 $m/2$ 组。因此,只需要 $m \times p/2$ 个 ToR 交换机、$m \times p/2$ 个叶交换机和 $m \times p/4$ 个脊交换机提供 $m \times p^2/4$ 个端口。特别地,如图 5 所示,考虑到单机 8 卡 GPU 服务器和最新的 51.2 Tbit/s(128×400 Gbit/s)交换机(例如,博通 Tomahawk 5[31] 和 Marvell Teralynx[32]),该拓扑可以轻松支持 32 768 个网卡接口和 1 280 个交换机。

3.2.2 负载均衡优化

因为难以适应低熵的流量模式,传统基于五元组的 ECMP(Equal-Cost Multi-Path)负载均衡策略在 LLM 训练里会遇到性能瓶颈。LLM 训练的流量特征和传统数据中心明显不同:流的数量级大幅下降,主导地位落在少量大象流(Elephant Flow)上。无状态的 ECMP 感知不到流的大小,分配时就容易出现流量分布不均,从而引发网络拥塞。又因为训练期间的流量模式一直很稳定,这种不均衡会长期存在,训练效率也就持续受到影响。

目前有许多负载均衡改进解决方案,包括基于主机[33-35]和基于交换机[36-38]的解决方案。然而,这 2 种方案都需要专有的网卡或交换机提供支持,从而引入了巨大的成本代价。因此,Galaxy 基于通用商用 RDMA 网卡、交换机与训练框架进行跨层协同优化,实现了对商用硬件的高效率和独立性。

在优化方案中,最重要的一点是减少跨 ToR 的流量。如图 6 所示,采用轨道优化拓扑不同服务器上具有相同索引的 GPU 在一个 ToR 交换机中通信,这种通信只需要 1 跳就能实现主机间的通信。本文在集合通信中还实现了一种拓扑感知的通信建链机制,以减少 ToR 间流量。

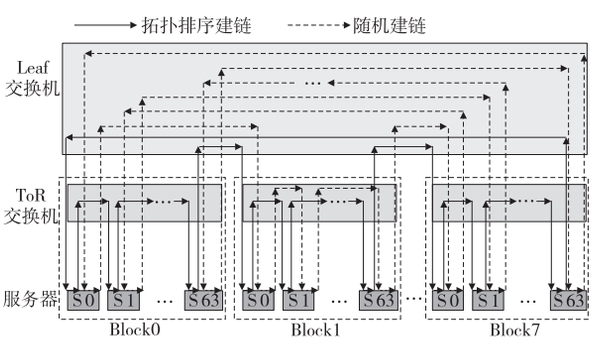


图 6 拓扑排序的流量编排

Figure 6 Traffic orchestration based on topological sorting

此外,在使用 MPI 或 TorchRun 等方式启动训练任务之前,先对主机列表进行拓扑排序,以确保相同 ToR 交换机下的 GPU 服务器相邻放置,而不是随机分布在整个主机列表中。该操作能够显著减少 ToR 交换机间流量,如图 6 所示,实线构成的通信环路明显短于不做排序的虚线部分。除了减少 ToR 交换机间流量外,本文还通过调优集合通信参数,如 *NCCL_IB_QPS_PER_CONNECTION* 来增加网络中流的数量,将大象流分解成更多的子流,从而增加网络中流量的熵。通过这些优化,根据本文的生产统计,AllReduce、ReduceScatter 和 AllGather 等集合通信提高了 27%~45%,带宽接近 400 Gbit/s 线速率。

3.2.3 拥塞控制优化

在 LLM 训练中,张量并行和序列并行由于传输数据量更大,一般都会被限制在服务器内网络,数据并行和流水线并行的流量通过 RDMA 网络实现机间互通。对于流水线并行来说,通信发生在 Send 和 Recv 原语之间,是一种点对点通信。数据并行由 AllReduce、ReduceScatter 或 AllGather 原语执行,通常使用基于树或环的算法实现。因此,网络中几乎没有“多打一”的流量,出现大规模拥塞的可能性很小[11]。

专家并行通常采用 AlltoAll 来进行数据交换,“多打一”的情况相比其他并行要明显得多。但是,专家并行组的通信域一般只在专家组机器之间,通信域很小。在此基础上,本文使用默认的 RDMA 拥塞控制算法,即 DCQCN(DataCenter Quantized Congestion Notification)[39]实现小规模拥塞控制参数调整。在训练带有专家并行的 MoE 模型时,借助 DeepSeek 实现的高效专家并行通信库,充分利用机内的 NVLink 带宽,减少机间的流量,让多打一的情况限定在单一轨道面,再配合调度平台进行亲和性调度策略,让专家并行组的机器都集中在同一个 ToR 下,实现专家并行流量不跨轨,进一步降低拥塞范围。

3.3 基础设施成本分析

硬件选型和技术栈的选择决定了集群搭建的 $CapEx$,本文在不同的硬件之间,配合软件的调试,在相同性能下实现更低的成本。与原始 IB(InfiniBand)集群解决方案相比,Galaxy 实现了相似的训练效率,成本降低了约 19%,节省了 1 亿美元,如表 2 所示。首先,网络成本显著降低。通过选择 2×200 Gbit/s 双端口网卡和白盒交换机,大大降低了网络设备(网卡、交换机、线缆和光模块)的成本,也缩短了交货周期。此外,网络拓扑支持超过 1 万个 GPU 的容量,交换机数量相对较少,

从而减少了所需的整体设备数量。与 IB 相比,本文的集群方案将网络成本降低了 70.63%,与传统的 RoCEv2(RDMA over Converged Ethernet v2)方案相比降低了 26.93%。与传统设计[40]相比,网络成本占数据中心总成本的 20%~30%,本文的目的是将网络设备成本降低到不高于 10%。其次,通过在服务器上部署时采用 AMD CPU,也降低了 0.32%的服务器成本,为集群建设节省了数十万美元。

表 2 Galaxy 方案与 IB、传统 RoCEv2 解决方案成本对比

Table 2 Cost comparison: Galaxy vs. InfiniBand vs. traditional RoCEv2 %

类目	方案		
	IB	RoCEv2	Galaxy
GPU 服务器	73.42	73.42	73.10
网卡	2.94	2.94	2.06
交换机	4.15	3.35	1.56
光模块	18.75	4.09	3.90
线缆	0.58	0.24	0.24
施工	0.16	0.16	0.16
合计	100.00	84.20	81.02

4 软件栈

4.1 通信库优化

4.1.1 集合通信库性能优化

集合通信库为上层分布式训练框架提供高吞吐量和低延迟的通信能力。本文在通信库层面实现了一系列跨层协同优化来获得更高的通信传输效率。基于服务器内部 NVSwitch 提供的 SHARP 技术,NVLSTree 算法通过将 AllReduce 操作卸载至 NVSwitch 以避免服务器内部 GPU 间的多次数据传输,显著提升了 AllReduce 原语的性能。然而,原生 NCCL 实现中的 NVLSTree 算法无法兼容双端口非绑定的 RDMA 网卡。而采用双端口绑定又会因低熵流量的哈希冲突导致端口间负载失衡。

为了解决这个问题,本文深入分析 NCCL 代码,发现原生 NVLSTree 的图搜索算法受限于 *NCCL_MAX_NVLS_ARITY* 参数,最多只能建立 8 个通道。在非绑定模式下,双端口 RNIC 的每个端口都会为同一 GPU 建立通道,导致只能覆盖 8 个 GPU 中的 4 个设备创建双向通道,造成 GPU 覆盖不全。

基于上述发现,本文对图搜索算法进行了重构:避开已经建立通道的 GPU,保证所有 GPU 和 RNIC 均被 NVLSTree 通道覆盖。为进一步优化通信的负载均衡,Galaxy 为未绑定 GPU 的 RNIC 设计了智能选卡机制。基于通道索引与 GPU 索引将创建的队列均匀分配至网卡的 2 个端口,使流量在双端口上实现均衡分布。

如图 7 所示,在不同集群规模下,Galaxy 非绑定算法配合 NVLSTree 展现出更高且更稳定的总线带宽。非绑定模式协同 NVLSTree 的通信性能显著超越传统环式 Ring、树形 Tree 结构算法。综上,Galaxy 非绑定方案通过充分利用双端口资源,在实现近似单端口网卡性能的同时能大幅降低成本,并与 NVLSTree 高效协同带来显著的性能提升。

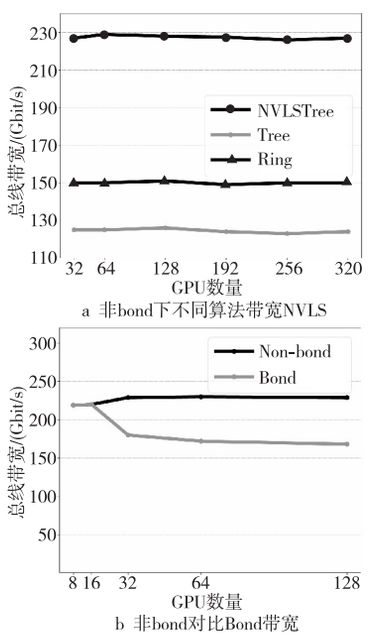


图 7 非 bond 对比 bond 下的带宽优势

Figure 7 Bandwidth advantage of non-bonding mode over bonding mode

原生 NCCL 在建立环形拓扑结构时,不同服务器会使用相同的 GPU 索引号排列作为环形拓扑的节点。该策略导致服务器间通信(尤其在数据并行场景中)产生显著的跨 ToR 交换机流量。如图 8 所示,所有 GPU 服务器均按照相同策略选择 GPU 0 作为入口节点、GPU 1 作为出口节点。虚线路径中,从 Server 0 的 GPU 1 发往 Server 1 的 GPU 0 的流量必须穿越 ToR 交换机和叶交换机,这不仅增加负载不均匀风险,还显著增加传输延迟。为解决这一问题,本文提出基于拓扑感知的通信建链算法以降低跨轨道流量。通过在半数 GPU 服务器上反转 GPU 索引号排列,确保环形链路中的流量始终在同索引 GPU 间传输。如图 8 中实线路径表明:Server 0 的 GPU 1 至 Server 1 的 GPU 1 的流量

仅需单跳传输(即经由 ToR 1 交换机)。

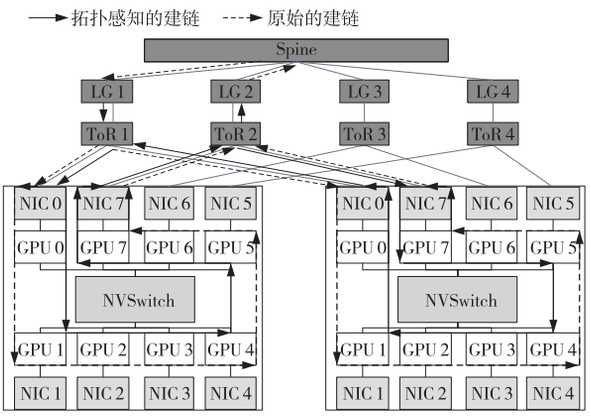


图 8 Galaxy 集合通信建链方式对比

Figure 8 Comparison of collective communication link establishment methods in Galaxy

4.1.2 集合通信库稳定性优化

服务器间 GPU 通信依赖于不同服务器 RNIC 之间的 RDMA 队列对 QP(Queue Pair)。然而,由于光模块数量庞大且易受施工过程中灰尘的影响等原因,使得网络端口宕机等故障频繁发生,中断上层应用的 QP 连接,导致训练进程持续等待通信返回,进而触发超时机制退出训练进程。在 2 000 卡集群可以观测到,由于网络问题导致任务波动或失败的比例能够占到 40%,仅次于由于 GPU 故障导致训练失败的比例。当出现网络故障训练退出后,只能通过排除故障机器,并从最近检查点的方式重新启动训练。这期间无论是等待时间还是重启回退都会造成严重的 GPU 时间浪费。为此,本文开发了 RNIC 的端口级容错机制,以避免端口故障时训练中断,同时借助备份链路继续训练提高 GPU 利用率。

在端口初始化 QP 时,系统会在另一端口同步创建备用 QP,从而使 RNIC 双端口互为备份。通信过程中,接收端定期从完成队列 WC(Work Complete)轮询状态,一旦检测到 QP 故障,接收端会向发送端发送通知消息,携带当前确认接收的数据信息,从而将双方的通信自动切换至另一备份端口的备用 QP,并按照交换的信息从断点位置接力发送。为了保证主 QP 恢复连通后,训练能够顺利回切,收发双方记录进行集合通信的次数 CC_{CNT},当满足阈值 Δ 次后,双方同时尝试切回主 QP;若此时链路仍未恢复正常,则继续使用备份链路进行数据传输,直到下一次达到阈值并重复上述过程。

4.1.3 下一代通信库

随着后训练和推理的不断发展,集合通信库在延迟敏感的训推场景中暴露出性能瓶颈。NCCL 需要 GPU 的内核线程与 CPU 的代理线程同步数据传输状态,控制路径一旦拉长,就会造成额外的数据传输延迟。DeepSeek 在训练中采用 IBGDA 通信,由 GPU 执行的内核函数直接发起通信过程,绕过 CPU 直接与 RNIC 交互,从而降低了通信的延迟。对训练效率和性能的分析还表明,MoE 训练不同阶段对算力的需求特征不同。Attention 阶段的计算核心是动态的、与序列长度平方相关的矩阵乘法,需要频繁读写大量键值(KV Cache),高度依赖高速内存带宽,属于内存访问密集型(memory-bound);前馈神经网络 FFN(Feed Forward Network)层则由静态、参数密集的矩阵乘构成,计算量远超通信量,属于计算密集型(compute-bound)。正因为二者属性不同,才可以分配到不同的 GPU 卡上分离训练。推理侧也有类似情形:预填充阶段(prefill)要处理完整的输入提示,做一次密集的并行注意力计算,性能主要卡在芯片的计算吞吐上(compute-bound);解码阶段(decode)则以自回归方式逐个生成词元,每次只算一个词元的注意力,计算量很小,性能却极度依赖读取权重的内存带宽(memory-bound)。这种情形下同样可以走 GPU 异构分离的路线,在提升推理性能的同时把成本降下来。不过这类分离传输的数据包小、频率高,对通信库的延迟要求更为苛刻。这也意味着,在现有高带宽之上,下一代通信库应兼顾实现更低延迟、小包友好的通信模式,并借助零拷贝技术把延迟压到极低,以此改善传输效率与场景适配能力。

4.2 训练框架优化

4.2.1 训练框架性能优化

在原版 Megatron 实现中,数据并行通信操作被置于每轮迭代的末端。本文观察到 DP 通信与反向计算不存在依赖关系,这使上述通信与计算操作具备重叠的优化潜力。

为提升训练效率,如图 9 所示,本文将数据并行通信与反向计算过程实施重叠优化。然而,该重叠机制并不是简单地将计算和数据并行的通信阶段进行组合就能达到预期的效果,在重叠机制上还存在 3 方面的问题:

1)不管是计算还是通信,都需要发起 GPU 内核函数并占用 SM 资源。重叠计算和通信首先要解决的就是 SM 数量争抢的问题。针对这种情况,通过分析可以发现,SM 的占用与通信时使用的通

道数量成正相关。因此,需区分重叠时的通信和非重叠时的通信,以此来控制通信对计算资源的占用,同时还能保证在重叠时间内完成发送任务。

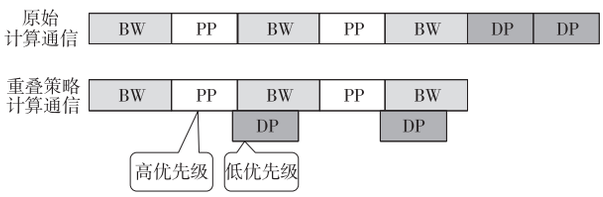


图 9 将数据并行通信与反向计算重叠

Figure 9 Overlapping data-parallel communication with backward computation

2)DP 通信耗时远长于 PP 通信,两者争抢带宽会拉大相邻 2 次计算之间的空泡。对此,本文在重叠阶段调节每次 DP 通信的数据量,尽量避开与 PP 通信的时间重合,并把数据并行的传输尽量塞进流水线里原本就存在的空泡中。

3)网络本身存在波动,DP 通信与 PP 通信仍有可能产生资源争用,降低流水线并行通信效率。由于该通信位于迭代的关键路径上,任何延迟都将阻碍后续的反向计算执行,造成昂贵的计算资源浪费。为了缓解这个问题,本文在网络层启用多优先级机制,并为流水线并行通信流量分配更高优先级。具体方案为:通过 PyTorch 框架对通信组设定优先级参数,该优先级参数随后传递至 NCCL 通信库,在网卡队列及网络交换机差分服务代码点 DSCP(Differentiated Services Code Point)生效。通过该机制确保数据并行通信不会干扰流水线并行通信,实现迭代时间的稳定降低。如图 10 所示,面向不同的大语言模型,这类重叠优化把单次迭代时间降低了 3%~7%。

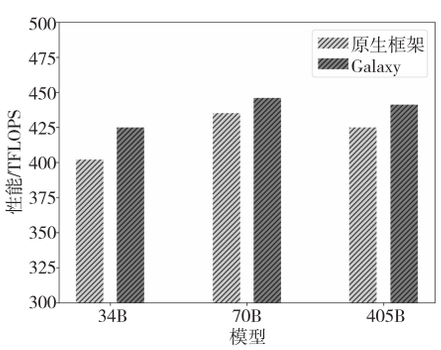


图 10 Galaxy 与原生 Megatron 在不同模型下性能对比

Figure 10 Performance of Galaxy vs. megatron on different models

4.2.2 检查点技术

Galaxy 使用检查点技术[41],按固定周期把模型状态存入持久化存储,故障之后再加载上一个状态来接着训练。但原生检查点走的是同步保存,开销明显:保存状态期间训练必须整体暂停,直到写完为止。为压缩这部分开销,本文设计了两阶段检查点,并依托 DLRover[42] 实现检查点与训练异步解耦。

两阶段检查点的具体流程:参数状态先在极短时间内同步写入共享内存,再由各主机上的独立进程异步地持久化到远程存储,GPU 的暂停时间由此大幅缩短。生产环境的实测表明,175B 模型用时不到 6 s,单次迭代存在约 5%的浮点运算量下降。配合通用并行文件系统 GPFS(General Parallel File System)等分布式文件系统,远程存储持久化过程能够进一步得到加速。即使遇到中断训练的故障,检查点数据也能完整落盘,确保了数据的高可靠性。实践中 70%以上的故障可以靠重启训练进程解决,重启后直接从内存加载检查点即可绕开 I/O 开销,而异步检查点机制会确保内存中的数据不会丢失。

4.3 跨层联合优化

本文在生产环境的数据中发现,大规模集群启动 LLM 训练任务时,初始化流程耗时十分显著。实测表明:基于 1 万张 Hopper 架构 GPU 运行 GPT-3.5 同规模预训练(采用 Megatron 框架),首轮迭代启动耗时约 10 min,根据本文的成本模型,过长的启动时间会严重制约 $avgMFU$ 。通过深入分析启动阶段的时间消耗,将性能瓶颈定位在 Megatron 初始化阶段中 Gloo 通信组创建过程。如图 11 所示,模型并行初始化函数耗时随服务器规模呈平方级增长,其根源在于 Gloo 通信组需要进行计算节点之间全互联的初始化操作。

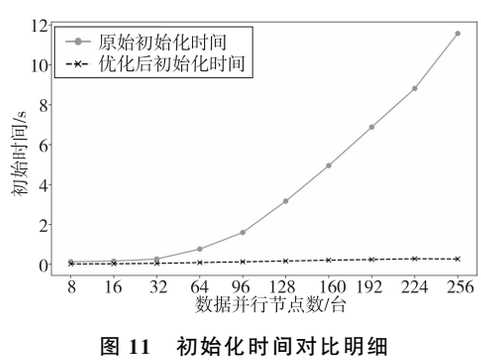


图 11 初始化时间对比明细

Figure 11 Comparison of initialization time comparison

进一步分析 Gloo 功能发现,Gloo 通信组主要用于检查点的存储加载操作,由 CPU 端单控制节点执行 Gather/Scatter 的集合通信操作。NCCL 在 CPU 侧也会建立类似的初始化网络,因此完全可以将该集合通信的功能迁移到 NCCL 中,将检

查点集合通信操作卸载至 NCCL 的初始化网络,以此来规避高开销的 Gloo 通信组构建,实现整体初始化时间的优化。

Galaxy 方案利用了初始化 NCCL 通信组时,在 CPU 端建立的 Bootstrap 网络,基于该网络在 CPU 端提供 Scatter 与 Gather 功能,从而实现 Gloo 在检查点操作中 CPU 侧通信原语的完整替代。可知经改进后,Megatron 初始化耗时保持线性关系,在服务器规模持续扩展时增长缓慢,与 Gloo 通信组初始化平方级的时间开销增长差异明显。

4.4 成本效益分析

软件的升级优化主要侧重在 2 方面,首先是尽可能地提高训练的峰值 MFU 性能。通过对 NCCL 通信的优化和更多的计算通信重叠,使得训练的峰值 MFU 提升 3%~5%。其次,则是对训练系统稳定性的优化,从而增大 α,降低故障对训练性能的损失。在引入备份 QP 之后实现网络的容错,能够在网络端口出现 down 时,以 99% MFU 继续训练而不会出现训练挂死,以及所有 GPU 闲置空等的状态。在这期间,可以同步安排现场运维更换模块或线缆的操作,实现不停机修复网络故障,在网络 down 口恢复后,容错能够重新切回原链路恢复到 100% MFU。进一步地,对网络 down 口状态的容错还能够防止由于超时错误导致的训练退出,从而节约重新启动和重新加载检查点的时间,提高集群训练效率。

5 运维与故障监控

5.1 Venus 监控与管理平台

学术界已在设备性能[43,44]、网络性能[45-47]及训练框架[19,20]的细粒度监测与诊断方面开展了大量研究。然而,这些研究往往只关注单一层面(如网络层[45])或特定问题(如 PFC[46]),无法提供覆盖整个集群和技术栈的综合可观测性方案。直接将各项独立研究方案进行简单组合也会带来显著的开销。为此,本文在生产环境中部署了一套监控管理系统,命名为 Venus,覆盖设备层的配置与性能、网络层性能以及框架层性能的全栈监测体系。

5.1.1 VenusCheck——设备级配置巡检

由于多服务和异构设备带来的复杂性,LLM 训练集群很容易因配置错误、机器故障、重装或重启导致配置不一致,从而产生训练启动失败、性能不佳的问题。本文提出了一套设备配置检查系统 VenusCheck,用于对主机端和交换机端进行有效的配置管理。其架构如图 12 所示。简言之,该系统在每个设备上设置了一个守护进程,定期检查配置并在检测到不一致时发出警报。目前,它监控服务器上的配置如 GPU、PCIe、PFC、DCQCN 和路由等指标,以及交换机上的 PFC、ECN、QoS 和路由等配置,检查间隔为 10 s。

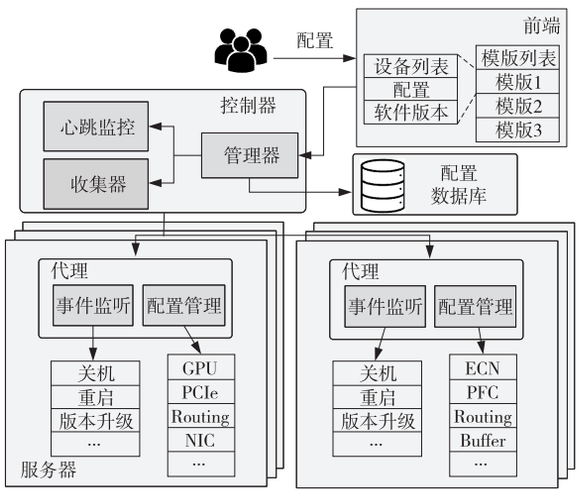


图 12 配置检测系统

Figure 12 Configuration verification system

运维人员通过 Web 前端为训练任务设置具有特定配置的集群,包括设备列表、软件版本和详细参数等。经管理员批准后,控制器生成具体配置:先将配置写入 SQL 数据库进行维护,再通过代理中间件安装到主机和交换机上。控制器还会定期向设备发送心跳包以维护其活跃状态,并对宕机设备发出警报。检查代理同时监听每台设备上可能影响配置一致性的各种事件,例如设备关机或重启导致的某些易失性配置失效。借助这套设备层配置检查系统,能够及时发现和定位配置不一致问题,并在生产环境中解决了大量配置一致性问题。

5.1.2 VenusMonitor——设备级性能监控

如图 13 所示,Galaxy 设计了一套主动性能监控系统 VenusMonitor,持续监测 GPU 服务器(GPU、CPU、内存和磁盘)及网络设备(含交换机、RNIC 和光模块)的运行时指标。

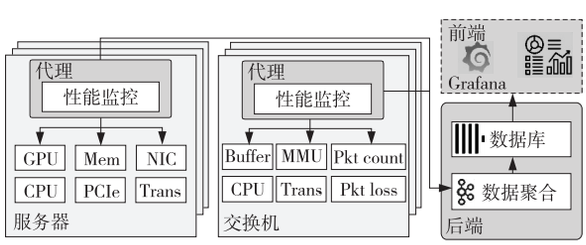


图 13 性能监控系统

Figure 13 Performance monitoring system

主机端代理监控 GPU、CPU、内存、磁盘、RNIC 及光模块的状态与性能,交换机端代理监测端口/队列层级的流量速率、控制报文计数(如 PFC、ECN、CNP 和 NACK)、丢包情况以及总内存使用量。网卡监控指标主要包括:各优先级流量大小、PFC、ECN、CNP 和 NACK 等控制消息计数,以及乱序包、错序包、超时包等错误日志。光模块监控指标涵盖电压、温度、偏置电流、接收光功率与发射光功率。

图 13 中,遥测数据通过 gRPC 从各代理汇聚至监控控制器,随后同步至 Kafka 消息队列[48]。数据库通过消费 Kafka 队列持续更新数据,基于 Grafana[49]的可视化系统实时渲染监测结果。完整监控项如表 3 所示。

表 3 监控项列表

Table 3 List of monitoring items

设备	种类	指标项
服务器	CPU 信息	CPU ID、Idle、IO Wait、IRQ、Nice、System、User
	GPU 信息	GPU ID、DecUtil、EncUtil、GPUUtil、MemUtil
	功率	GPUPower
	温度	GPUTemp
	PCIe	PciRetry、PciRxBytes、PciTxBytes
	内存使用	Total、Available、Used、Cached、Swap
	网卡计数	PFC、Recv/Trans、Control Packet、Buffer Usage、Out-of-order、Out-of-buffer、CNP、R/W Request
	接口计数	Status、Traffic、Error
	光模块	Serial Number、Type、Power、Temperature、Bias Current、Voltage
交换机	基本信息	IP、Name、Manufacturer、CPU Usage
	缓存	Egress/Ingress Buffer Usage
	丢包	Dropped Bytes/Packets
	接口计数	Status、Traffic、Error、LSM
	PFC/ECN	ECN Marked、WRED Dropped、PFC Recv/Send Num
	光模块	Serial Number、Type、Power、Temperature、Bias Current、Voltage

Galaxy 性能监控系统实现了网络全域设备的系统性可观测,使性能劣化能够被自动检测并快速启动诊断流程。

5.1.3 VenusToolset——检测工具集

设备级配置检查系统与性能监控系统提供了全面的单设备指标覆盖,但它们只关注单一设备的性能指标,对整体网络行为的监测能力有限。部分状态信息,如协议交互行为、ECMP 端口选择信息等在运行期间会动态变化,这些因素使得网络级性能监控成为必要。为此,本文引入了多项网络级监控工具,搭建了一整套工具集来应对不同问题。

受 TCP PingMesh[50]启发,Galaxy 支持类似 RDMA PingMesh[45]这样的工具来检测 RDMA 网络连通性,辅助定位网络相关问题。

本文还部署了带内网络遥测 INT(In-band Network Telemetry)技术,能实时看到数据包级的 ECMP 路径和交换机队列深度,把运行时的网络状态采集完全放在数据平面里完成。

本文采用 ERSPAN(Encapsulated Remote Switch Port ANalyzer)[51]跟踪协议行为变化,该技术利用交换芯片的用户自定义字段 UDF(User Defined Fields)特性镜像特定数据包副本,并将其发送到目标服务器进行分析。ERSPAN 支持设置采样率与报文截取长度,可有效降低目标服务器负载。使用该技术,能进行数据包粒度的会话分析、观察协议具体交互情况并获取更细粒度的信息。这些网络级监控系统为日常运维、故障排查与性能优化提供了丰富的数据支撑。

5.1.4 VenusTrace——训练性能遥测

在应用层,本文为 Galaxy 提出了训练遥测系统 VenusTrace,用于监控训练性能并分析计算与通信之间的依赖,进而识别制约性能的关键路径。VenusTrace 实现了以毫秒级精度监测 RDMA 通信性能以检测训练异常。传统轮询脚本只能提供秒级甚至分钟级的速率监控,无法捕捉毫秒级的 NCCL 通信操作。本文通过监控 WR(Work Request)与 WC 来估算通信性能。实现上采用一个默认覆盖 50 条数据的滑动窗口,把 WR 提交与数据真正开始传输之间的间隙压到最小。这样窗口内的传输时延处在数十微秒量级,足以对 RNIC 吞吐量做高精度测量。

如图 14 所示,在 80 张 Hopper 架构 GPU 上执行 ReduceScatter 操作时,当最小通道数设置为 16 时,RtS(Ready to Send)数据大小能准确反映对应集合通信原语的带宽,展示毫秒级精度。

其次,VenusTrace 还能追踪 NCCL API 调用时间以监控计算性能。通过记录每次 API 调用的时间戳,本文能够分析训练阶段的等待依赖关系,并基于批次索引、阶段信息与耗时等元数据定位关键路径上的慢速服务器。如图 15 所示,通过分析 AllReduce 调用间的时间间隔 $time_$ 可近似估算前向或反向传播的耗时,而通信调用的位置分析能获取训练过程中异常的详细节点和阶段。

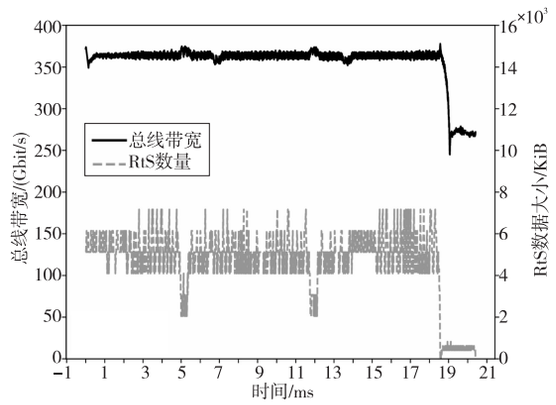


图 14 网卡中性能带宽与 RtS 数量关系

Figure 14 NIC bandwidth vs. number of RtS

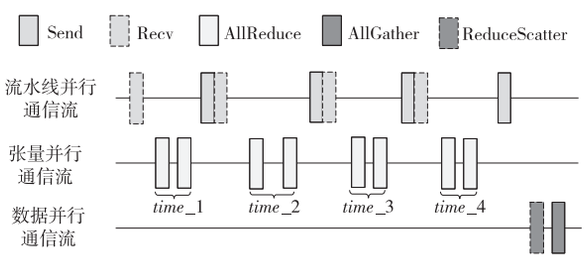


图 15 通过 API 调用间隔反映计算时间

Figure 15 Inferring computation time from API call intervals

5.2 问题定位标准流程

在大规模 GPU 集群中,任何硬件或软件故障都会导致显著的性能下降甚至训练中断。经过长期实践,本文沉淀出一套标准化的性能异常诊断流程,硬件、网络、通信库到框架各层的定位都被纳入其中。流程首先依托 Galaxy 的综合监控系统提供细粒度可观测性,MFU 下降、网络带宽回落和错误告警等异常都能被及时捕捉。接着借助对监控数据的分析,很快就能定位到出故障的组件,具体指向某台服务器节点。多数故障(例如网卡宕机、GPU 报错等硬件问题,或路由表配置错误)都能由监控系统直接定位;故障设备经 K8S 驱逐,由健康节点顶替后,训练任务再加载最新检查点即可快速恢复。

对于监控系统覆盖不到的未知异常,如性能缓慢降低或挂起这类静默故障,传统二分搜索法需要执行 $O(\log \widetilde{N})$ 次子集群测试。为加速诊断,本文设计双配对测试法:将 $\widetilde{N}$ 台服务器以 2 种不同方式配对为 $\widetilde{N}/2$ 组,确保任意服务器在不同排列中不会重复配对,只需 2 组配对测试就能精确定位故障节点。

接下来的排查按层展开:先用基于 Megatron 的 GEMM 计算测试查单节点的算力问题;再依照 Megatron 的通信组配置,调用 AllReduce、AlltoAll 等不同 NCCL-tests 原语做组间通信测试,若这一步性能达标,就排除通信因素,把问题归到框架层;最后用 MPI 运行 Perftest 搭出全连接拓扑,测点对点的 RDMA 通信性能,如果网络也正常,故障基本就落在单节点内部,例如 GPU、NVLink 或 PCIe。

每次故障分析后,本文将新发现的监控指标纳入系统。例如:训练过程中 GPU 瞬时降频问题被从启动检查调整为实时监控,实现更细粒度的状态追踪。当前监控系统已覆盖绝大多数常见指标,99% 以上的故障可在 5 分钟内完成诊断定位与恢复。

6 经验总结

6.1 故障定位

1)训练挂起。在执行一个 3 840 卡 GPU 的训练任务时,出现了随机训练卡死现象,训练进行到几小时后出现挂起。通过分析 VenusTrace 采集的 NCCL 调用,发现卡死发生在 Node 188 与 Node 107 之间的 Send/Recv 操作,Node 107 完成前向计算并调用 Send 通信后,Node 188 的 GPU 2 未按预期执行对应 Recv 操作。为确认问题源自 Node 188,本文再次携带该节点重启训练。在第 2 轮数据采集中发现,同一批次的反向传播阶段,Node 188 的 GPU 2 与 GPU 3 未能完成数据同步。2 次分析都表明 Node 188 的 GPU 2 存在问题,本文立刻隔离该节点并重启训练继续观察。此后训练持续正常运行 3 天,最终因 GPU ECC 错误终止,期间没有再出现卡死引发的重启事件。与此同时,使用 DCGM(NVIDIA Data Center GPU Manager)工具对 Node 188 进行测试全部通过,但 GPUBurn 工具显示 GPU 2 内存使用率只能到 20%。在单 GPU 小规模训练测试中,经过数小时后在 GPU 2 上复现该问题。测试中还检测到 XID 109 错误,这在 3 840 卡规模训练故障时未出现,由此确认 Node 188 存在硬件故障。

2)性能异常。在训练过程中,同样遭遇到很多训练性能异常的情况。除了前文提到的瞬时的 Slowdown 状态,本文还遇到训练性能的抖动和掉速。经过分析,训练抖动原因来自于新版本的采集代理占用了超过 400% 的 CPU 核,这种大量占用 CPU 的情况会影响到 GPU 核调用的速度,从而影响到训练的性能。为此,本文将非训练进程的 CPU 占用情况纳入到监控之中,一旦有超过

200%占用的程序就会发出告警信息。

3)基础环境问题。除了训练过程出现的性能问题,本文还遇到了很多基础环境和用户操作的问题。包括 K8S 组件版本的原因导致 IPTables 持续被写入更新导致 NCCL 建链失败、K8S 业务端口与训练建链端口冲突以及用户使用镜像版本不一致导致启动失败等问题。为此本文不断完善交付和部署 SOP 形成体系化的交付流程,保持对快速更新的训练组件持续测试验证的同时,用确定性的交付版本进行交付。完善 VenusCheck,对用户行为、关键环境变量进行检查,快速发现并解决问题。

6.2 系统优化

1)跨层协同优化。传统系统设计讲究用分层架构来屏蔽复杂性,但层与层之间的透明化同样会拖累效率。一个表现是,训练任务不了解底层基础设施,应用便无从挑选最优的优化策略,白白增加大量额外开销,甚至造成资源浪费。举例来说,学术界有不少研究试图用各种复杂技术去优化网络负载均衡[33-38]。如果上层框架能够意识到底层的网络拓扑结构,基于轨道优化的拓扑结构和拓扑感知的 GPU 建链可以是简单而有效的。另一方面,不同的模型与场景对带宽和时延的敏感性是不一样的。微软曾在 NCCL 上进行了大量优化[52],显著减少了小消息的延迟。然而,当本文将新的集合通信库应用到训练框架时,并没有看到训练吞吐量的改善。对于稠密模型的训练任务来说,更注重吞吐量而不是延迟,而对于推理场景来说,能够降低用户使用延迟的优化更能体现价值。总的来说,任何没有考虑框架特性而进行的优化改进对最终端到端的训练提升是很小甚至不必要的。

从上面的例子中可以看出,LLM 训练的跨层协同优化对性能提升十分重要。当框架可以直接调用底层接口、底层栈又能够贴合上层框架的真实需求时,原本跨越多层才能解决的性能瓶颈就变得容易处理,端到端性能也随之明显改善。

2)全栈监控的重要性。跨层协同优化把训练性能同框架效率、CCL 调度、流量模式乃至硬件配置都联系在一起,导致这些组件之间的界线变得模糊,监控与运维的复杂度也随之上升。传统的监控或诊断解决方案[43,44,47]通常只关注某一特定层次,而集成所有层次的监控会引入很高的开销。本文观察到,与其依赖复杂的监控诊断算法,不如用更直接的手段来获得系统可观测性,比如定期的配置检查加上全面的性能监控,反而能更快完成排障。这样就把故障处理串成了一个闭环,端到端的运维管理做到了全面、快速而且可解释。

6.3 功能开发与迭代

尽管 Galaxy 的多层次优化部署带来了显著的性能提升,但在实际生产环境中仍会出现未知技术问题。为此,本文建立了测试驱动的迭代验证流程来确保每个新功能的可靠性。

1)小规模功能验证:搭建由少量 GPU 服务器组成的测试拓扑,用于排除硬件(如光模块)和软件(网络操作系统、RNIC 驱动、NCCL 及监控系统)的初期功能性问题。

2)中规模网络验证:构建数十台服务器组成的测试集群,配置与生产环境一致但不配备 GPU,通过大规模训练流量负载测试验证网络配置的鲁棒性。该阶段可识别绝大多数网络层问题。

3)生产环境性能评估:在配备大量 GPU 的生产环境中部署新功能,重点评估大规模 GPU 通信性能。通过测试后,继续部署完整训练框架以评估端到端训练性能。

4)正式投产:将验证通过的功能集成至生产环境并交付客户使用。

7 相关工作

1)大型语言模型训练的部署经验。HU 等人[53]对 LLM 的训练负载进行了深入分析,并总结出一系列可用于进一步优化的关键洞察。Mega-Scale[2]实现了训练算法与训练框架的协同优化,但对底层硬件与网络调优关注较少。GANGIDI 等人[11]分享了 Meta 在 AI 训练中基于 RDMA 的网络部署经验,重点涵盖了网络结构与传输协议的设计与实践。阿里巴巴提出的 HPN 架构[12]专为 LLM 训练中的网络通信优化而设计,主要聚焦于网络层的调优,较少涉及软件栈的协同优化。Fire-flyer[13]提出了一个面向 PCIe 服务器的软件-硬件协同设计框架,旨在降低训练成本,但该方案未对训练的成本效率进行系统性分析,而这正是本文工作 Galaxy 所重点呈现的内容。

2)监控与诊断工具。Marple[54]、Sonata[55]以及 Flow[56]提出了基于查询驱动的遥测系统,可灵活监测网络异常行为。NetSeer[57]与 PINT[58]通过对遥测数据进行聚合,进一步降低了监控系统的开销。RD-Probe[59]结合随机与确定性方法生成高覆盖率的探测任务集合,提升监控的全面性。在主机内网络性能分析方面,Hostping[44]、Collie[43]、Husky[60]和 Lumina[61]等工具尝试了多种基于探测或测试的方法,用于定位主机内部网络或微架构

层面的性能瓶颈与异常。

区别于上述复杂设计,本文借助业界经验,设计并实现了简洁高效的监控与诊断工具,在保持系统简单性的同时,达到更高的故障定位效率。

3)跨层训练优化。TCCL[62]让集合通信库CCL能够感知主机内部的网络结构,帮助识别高性能的PCIe通信路径。TACCL(Topology Aware Collective Communication Library)[63]则赋予CCL对底层拓扑结构的感知能力,生成适应拓扑结构的高效通信算法。MCCS(Managed Collective Communication as a Service)[64]集成了多租户集群信息,允许云服务提供商根据租户需求配置QoS策略。TopoOpt[65]实现了并行策略与网络拓扑的联合优化,提升分布式训练效率。MLT(Machine Learning Transport for AI-centric networking)[66]根据数据包对模型精度的重要性进行调度优化,包括负载均衡与优先级队列等机制。

在此基础上本文结合实践中的性能剖析结果,在训练框架、NCCL通信库与硬件层面展开协同优化,形成一套面向大模型训练的跨层联合优化方案。

8 结束语

本文围绕Galaxy这一面向生产的大规模LLM训练集群方案,把设计、优化与运维过程中的实践经验做了一次完整梳理。Galaxy建立在面向成本优化的基础设施之上,涵盖硬件调优与网络优化,兼顾高计算能力与低成本投入。在软件栈内部,对各层开展深度协同设计,全面提升训练效率。在运行时层面,Galaxy配备了细粒度的全栈监控体系、高效的容错机制以及标准化的故障排查流程,保障系统在大规模训练任务中的稳定性与可靠性。最后,本文介绍了在生产环境中遇到的实际问题,本着"如无必要,勿增实体"的原则,以白盒可解释的方式使用VenusCheck监控诊断集群异常,配合VenusTrace和VenusToolset实现全方位的问题定位诊断。

参考文献:

[1] KAPLAN J, MCCANDLISH S, HENIGHAN T, et al. Scaling laws for neural language models[J]. arXiv: 2001.08361, 2020.

[2] JIANG Z H, LIN H B, ZHONG Y M, et al. MegaScale: Scaling large language model training to more than 10,000 GPUs[C]// Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation, 2024: 745-760.

[3] CAO J M, GUAN Y, QIAN K, et al. Crux: GPU-efficient communication scheduling for deep learning training[C]// Proceedings of the 38th ACM SIGCOMM Conference, 2024: 1-15.

[4] How much is an NVIDIA H100? [EB/OL]. [2025-10-10]. https://modal.com/blog/nvidia-h100-price-article.

[5] FS. N9600-128QC switch data sheet[EB/OL]. [2025-10-10]. https://resource.fs.com/mall/resource/n9600-128qc-switch-data-sheet.pdf.

[6] GPT-4 architecture, infrastructure, training dataset, costs, vision, MoE [EB/OL]. [2025-10-10]. https://newsletter.semianalysis.com/p/gpt-4-architecture-infrastructure.

[7] CHOWDHERY A, NARANG S, DEVLIN J, et al. PaLM: Scaling language modeling with pathways[J]. Journal of Machine Learning Research, 2023, 24(1): 11324-11436.

[8] GRATTAFIORI A, DUBEY A, JAUHRI A, et al. The Llama 3 herd of models[J]. arXiv: 2407.21783, 2024.

[9] JOUPPI N, KURIAN G, LI S, et al. TPU v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings[C]// Proceedings of the 50th Annual International Symposium on Computer Architecture, 2023: Article No.: 82.

[10] COMANICI G, BIEBER E, SCHAEKERMANN M, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities[J]. arXiv: 2507.06261, 2025.

[11] GANGIDI A, MIAO R, ZHENG S B, et al. RDMA over Ethernet for distributed training at meta scale[C]// Proceedings of the 38th ACM SIGCOMM Conference, 2024: 57-70.

[12] QIAN K, XI Y Q, CAO J M, et al. AlibabaHPN: A data center network for large language model training[C]// Proceedings of the 38th ACM SIGCOMM Conference, 2024: 691-706.

[13] AN W, BI X, CHEN G T, et al. Fire-Flyer AI-HPC: A cost-effective software-hardware co-design for deep learning[C]// Proceedings of the 36th International Conference for High Performance Computing, Networking, Storage, and Analysis, 2024: 1314-1336.

[14] NARAYANAN D, SHOEYBI M, CASPER J, et al. Efficient large-scale language model training on GPU clusters using megatron-LM[C]// Proceedings of the 34th International Conference for High Performance Computing, Networking, Storage and A-

nalysis,2021:Article No.:58.

[15] RASLEY J,RAJBHANDARI S,RUWASE O,et al. DeepSpeed: System optimizations enable training deep learning models with over 100 billion parameters[C]//Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining,2020:3505-3506.

[16] LIAO H,LIU B Y,CHEN X P,et al. UB-Mesh:A hierarchically localized nD-FullMesh data center network architecture[J]. IEEE Micro,2025,45(5):20-29.

[17] KOKOLIS A,KUCHNIK M,HOFFMAN J,et al. Revisiting reliability in large-scale machine learning research clusters[C]//Proceedings of the 31st IEEE International Symposium on High Performance Computer Architecture ,2025:1259-1274.

[18] The FLOPs calculus of language model training [EB/OL]. [2025-10-10]. https://readmedium.com/the-flops-calculus-of-language-model-training-3b19c-1f025e4.

[19] GUPTA T,KRISHNAN S,KUMAR R,et al. Just-in-time checkpointing:Low cost error recovery from deep learning training failures[C]//Proceedings of the 19th European Conference on Computer Systems,2024:1110-1125.

[20] WANG Z, JIA Z, ZHENG S, et al. Gemini: Fast failure recovery in distributed training with in-memory checkpoints[C]//Proceedings of the 29th Symposium on Operating Systems Principles,2023:364-381.

[21] NVIDIA scalable hierarchical aggregation and reduction protocol (SHARP) rev 3.8.0[EB/OL]. [2025-10-10]. https://docs.nvidia.com/nvidia-scalable-hierarchical-aggregation-and-reduction-protocol-sharp-rev-3-8-0.pdf.

[22] NVIDIA Spectrum-X Ethernet network platform architecture: White paper[EB/OL]. [2025-10-10]. https://resources.nvidia.com/en-us-accelerated-networking-resource-library/nvidia-spectrum-x.

[23] ARISTA. AI networking: White paper[EB/OL]. [2025-10-10]. https://www.arista.com/assets/data/pdf/Whitepapers/AI-Network-WP.pdf.

[24] SONiC[EB/OL]. [2025-10-10]. https://sonicfoundation.dev/.

[25] BAI W,ABDEEN S S,AGRAWAL A,et al. Empowering azure storage with RDMA[C]//Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation,2023:49-67.

[26] POUTIEVSKI L,MASHAYEKHI O,ONG J,et al. Jupiter evolving:Transforming Google's datacenter network via optical circuit switches and software-defined networking[C]//Proceedings of the 36th ACM SIGCOMM Conference,2022:66-85.

[27] NVIDIA. NVIDIA DGX SuperPOD:Next generation scalable infrastructure for AI leadership, reference architecture featuring NVIDIA DGX B200[EB/OL]. (2025-04-24)[2025-10-10]. https://docs.nvidia.com/dgx-superpod/reference-architecture-scalable-infrastructure-b200/latest/index.html.

[28] The Linux kernel. Linux Ethernet bonding driver HOWTO[EB/OL]. [2025-10-10]. https://docs.kernel.org/networking/bonding.html.

[29] NVIDIA. NVIDIA collective communication library (NCCL) documentation: Release notes 2.12[EB/OL]. [2025-10-10]. https://docs.nvidia.com/deeplearning/nccl/release-notes/rel_2-12-7.html.

[30] ABHASHKUMAR A, SUBRAMANIAN K, ANDREYEV A,et al. Running BGP in data centers at scale[C]//Proceedings of the 18th USENIX Symposium on Networked Systems Design and Implementation,2021:65-82.

[31] BROADCOM. Tomahawk 5/BCM78900 series[EB/OL]. [2025-10-10]. https://www.broadcom.com/products/ethernet-connectivity/switching/strataxgs/bcm78900-series.

[32] MARVELL. Marvell Teralynx 10 51.2T Ethernet switch enters volume production for global AI cloud deployments[EB/OL]. (2024-07-25)[2025-10-10]. https://investor.marvell.com/2024-07-25-Marvell-Teralynx-10-51-2T-Ethernet-Switch-Enters-Volume-Production-for-Global-AI-Cloud-Deployments.

[33] DONG J B,WANG S C,FENG F,et al. ACCL:Architecting highly scalable distributed training systems with highly efficient collective communication library[J]. IEEE Micro,2021,41(5):85-92.

[34] QURESHI M A,CHENG Y C,YIN Q W,et al. PLB:Congestion signals are simple and effective for network load balancing [C]//Proceedings of the 36th ACM SIGCOMM Conference,2022:207-218.

[35] SHALEV L, AYOUB H, BSHARA N,et al. A cloud-optimized transport protocol for elastic and scalable HPC[J]. IEEE Micro,2020,40(6):67-73.

[36] BROADCOM. Broadcomunveils industry's highest performance fabric for AI networks [EB/OL]. [2025-10-10]. https://investors.broadcom.com/news-releases/news-release-details/broadcom-unveils-industrys-highest-performance-fabric-ai.

[37] NVIDIA. NVIDIA Spectrum-X Ethernet platform for AI networking[EB/OL]. [2025-10-10]. https://www.nvidia.com/en-us/networking/spectrumx/.

[38] SONG C H,KHOOI X Z,JOSHI R,et al. Network load balancing with in-network reordering support

for RDMA[C]//Proceedings of the 37th ACM SIGCOMM Conference,2023:816-831.

[39] ZHU Y B,ERAN H,FIRESTONE D,et al. Congestion control for large-scale RDMA deployments[J]. ACM SIGCOMM Computer Communication Review,2015,45(4):523-536.

[40] GREENBERG A,HAMILTON J,MALTZ D A,et al. The cost of acloud:Research problems in data center networks[J]. ACM SIGCOMM Computer Communication Review,2008,39(1):68-73.

[41] EISENMAN A,MATAM K K,INGRAM S,et al. Check-N-Run:A checkpointing system for training deep learning recommendation models[C]//Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation,2022:929-943.

[42] WANG Q L,LAN FT,TANG Y H,et al. DLRover-RM:Resource optimization for deep recommendation models training in the cloud[J]. Proceedings of the VLDB Endowment,2024,17(12):4130-4144.

[43] KONG X H,ZHU Y B,ZHOU H P,et al. Collie:Finding performance anomalies in RDMA subsystems[C]//Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation,2022:287-305.

[44] LIU K F,JIANG Z,ZHANG J,et al. Hostping:Diagnosing intra-host network bottlenecks in RDMA servers[C]//Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation,2023:15-30.

[45] LIU K F,JIANG Z,ZHANG J,et al. R-Pingmesh:A service-aware ROCE network monitoring and diagnostic system[C]//Proceedings of the 38th ACM SIGCOMM Conference,2024:554-567.

[46] WU X C, NG T S E. Detecting and re-solving PFC deadlocks with ITSY entirely in the data plane[C]//Proceedings of the 58th IEEE Conference on Computer Communications,2022:1928-1937.

[47] WANG W T,WU X ,TAMMANA P,et al. Closed-loop network performance monitoring and diagnosis with SpiderMon[C]//Proceedings of the 19th USENIX Symposium on Networked Systems Design and Implementation,2022:267-285.

[48] KREPS J, NARKHEDE N,RAO J. Kafka:A distributed messaging system for log processing[C]//Proceedings of the 6th NetDB. 2011:1-7.

[49] Grafana Labs [EB/OL]. [2025-10-10]. https://grafana.com/.

[50] GUO C X,YUAN L H,XIANG D,et al. Pingmesh:A large-scale system for data center network latency measurement and analysis[C] //Proceedings of the 29th ACM Conference on Special Interest Group on Data Communication,2015:139-152.

[51] CISCO. Configuring SPAN and RSPAN:Network management configuration guide (Catalyst 9300 Switches,IOS XE 17. 12. x)[EB/OL]. [2025-10-10]. https://www.cisco.com/c/en/us/td/docs/switches/lan/catalyst9300/software/release/17-12/configuration_guide/nmgmt/b_1712_nmgmt_9300_cg/configuring_span_and_rspan.html.

[52] COWAN M,MALEKI S,MUSUVATHI M,et al. MSCCLang:Microsoft collective communication language[C]//Proceedings of the 37th ACM ASPLOS Conference,2023:502-514.

[53] HU Q H,YE Z S,WANG Z R,et al. Characterization of large language model development in the datacenter[C]//Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation,2024:709-729.

[54] NARAYANA S,SIVARAMAN A,NATHAN V,et al. Language-directed hardware design for network performance monitoring[C]//Proceedings of the 31st Conference of the ACM Special Interest Group on Data Communication,2017:85-98.

[55] GUPTA A,HARRISON R,CANINI M,et al. Sonata:Query-driven streaming network telemetry[C]//Proceedings of the 32nd Conference of the ACM Special Interest Group on Data Communication,2018:357-371.

[56] SONCHACK J,MICHEL O,AVIV A J,et al. Scaling hardware accelerated network monitoring to concurrent and dynamic queries with Flow[C]//Proceedings of the 32nd USENIX Annual Technical Conference,2018:823-835.

[57] ZHOU Y,SUN C,LIU H H,et al. Flow event telemetry on programmable data plane[C]//Proceedings of the 34th Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication, 2020: 76-89.

[58] BASAT R B,RAMANATHAN S,LI Y L,et al. PINT:Probabilistic in-band network telemetry[C]//Proceedings of the 34th Annual Conference of the ACM Special Interest Group on Data Communication,2020:662-680.

[59] DING R, LIU X P,YANG S B,et al. RD-Probe:Scalable monitoring with sufficient coverage in complex datacenter networks[C]//Proceedings of the 38th ACM SIGCOMM Conference,2024:258-273.

[60] KONG X H,CHEN J R,BAI W,et al. Understanding RDMA microarchitecture resources for performance isolation[C]//Proceedings of the 20th USENIX

Symposium on Networked Systems Design and Implementation,2023:31-48.
[61] YU Z L,SU B W,BAI W,et al. Understanding the micro-behaviors of hardware offloaded network stacks with lumina[C]//Proceedings of the 37th ACM SIGCOMM Conference,2023:1074-1087.
[62] KIM H,RYU J,LEE J. TCCL:Discovering better communication paths for PCIe GPU clusters[C]//Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems,2024:999-1015.
[63] SHAH A,CHIDAMBARAM V,COWAN M,et al. TACCL:Guiding collective algorithm synthesis using communication sketches[C]//Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation,2023:593-612.
[64] WU Y J,XU Y C,CHEN J R,et al. MCCS:A service-based approach to collective communication for multi-tenant cloud[C]//Proceedings of the 38th ACM SIGCOMM Conference,2024:679-690.
[65] WANG W Y,KHAZRAEE M,ZHONG Z Z,et al. TopoOpt:Co-optimizing network topology and parallelization strategy for distributed training jobs[C]//Proceedings of the 20th USENIX Symposium on Networked Systems Design and Implementation,2023:739-767.
[66] WANG H,TIAN H,CHEN J R,et al. Towards domain-specific network transport for distributed DNN training[C]//Proceedings of the 21st USENIX Symposium on Networked Systems Design and Implementation,2024:1421-1443.

作者简介:



焦方正(1998—),男,山东济南人,博士生,CCF学生会员(X0206G),研究方向为分布式系统和大模型训练基础设施。**E-mail**:jiaofz@buaa.edu.cn

JIAO Fangzheng, born in 1998, PhD candidate, CCF student member(X0206G), his research interests include distributed system and large model training infrastructure.



张梦豪(1994—),男,陕西西安人,博士,副教授,CCF高级会员(54969S),研究方向为计算机网络、网络系统、深度学习系统和网络安全。**E-mail**:zhangmenghao@buaa.edu.cn

ZHANG Menghao, born in 1994, PhD, associate professor, CCF senior member(54969S), his research interests include computer network, network system, deep learning system, and network security.



王士诚(1997—),男,山东泰安人,博士,研究方向为RDMA网络和网络异常诊断。**E-mail**:wangshicheng1225@163.com

WANG Shicheng, born in 1997, PhD, his research interests include RDMA networks and network anomaly detection.



贾彦民(1976—),男,河南开封人,博士,研究方向为分布式集合通信库和大模型网络架构。**E-mail**:jiayanmin@infrawaves.com

JIA Yanmin, born in 1976, PhD, his research interests include distributed collective communication library and large mode network architecture.



张琰(1983—),男,河北邢台人,研究方向为领域为高性能集合通信库。**E-mail**:zhangyan@infrawaves.com

ZHANG Yan, born in 1983, his research interest includes high-performance collective communication library.



张铭俊(2002—),男,江苏苏州人,硕士生,研究方向为大模型训练的稳定性和低延迟通信库。**E-mail**:zhangmingjun@Infrawaves.com

ZHANG Mingjun, born in 2002, MS candidate, his research interests include stability in large-scale model training and low-latency communication library.



胡效赫(1992—),男,河北沧州人,博士,研究方向为软件定义网络和大模型训练基础设施。**E-mail**:huxiaohe@infrawaves.com

HU Xiaohe, born in 1992, PhD, his research interests include software defined networking and large model training infrastructure.



胡春明(1977—),男,陕西渭南人,博士,教授,CCF杰出会员(08112D),研究方向为分布式计算、云计算和软件工程。**E-mail**:hucm@buaa.edu.cn

HU Chunming, born in 1977, PhD, professor, CCF distinguished member(08112D), his research interests include distributed computing, cloud computing, and software engineering.