计算机研究与发展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(11): 2806−2825, 2025

面向智算中心的细粒度流量控制技术综述

许 晶[1,2] 王 展[1] 元国军[1] 马振龙[2] 杨 帆[1] 孙凝晖[1]
[1](中国科学院计算技术研究所 北京 100190)
[2](中国科学院大学 北京 100190)
(xujing@ncic.ac.cn)

Survey on Fine-Grained Flow Control for Artificial Intelligence Data Centers

Xu Jing[1,2], Wang Zhan[1], Yuan Guojun[1], Ma Zhenlong[2], Yang Fan[1], and Sun Ninghui[1]
[1](*Institute of Computing Technology*, *Chinese Academy of Sciences*, *Beijing* 100190)
[2](*University of Chinese Academy of Sciences*, *Beijing* 100190)

Abstract With the rapid development of artificial intelligence generated content (AIGC) technologies and the widespread deployment of large language models (LLMs), networks in artificial intelligence data centers are encountering significant challenges. Flow control is a crucial approach for optimizing network performance, providing extremely high bandwidth and ultra-low latency. This paper reviews key issues and solutions in the field of fine-grained flow control, focusing on advances in three key areas: adaptive load balancing mechanisms that dynamically distribute traffic making full use of network resources to avoid congestion, proactive congestion control strategies designed to predict and alleviate potential congestion, and out-of-order packet reordering techniques that ensure data integrity despite non-sequential arrivals. We summarize the mainstream implementation solutions and provide a detailed comparison. Building on this, we discuss the key technical solutions currently adopted by leading artificial intelligence data centers, along with the network devices that support fine-grained flow control. We also identify unresolved challenges in this field, propose potential solutions, and explore future development trends, especially as AI technologies continue to evolve and demand more sophisticated network infrastructures. This review offers valuable insights for researchers and practitioners to optimize network performance in AI-driven applications and highlights important directions for future research in fine-grained flow control.

Key words load balancing; congestion control; packet spray; out-of-order reordering; artificial intelligence data centers

摘 要 随着人工智能生成内容技术的迅猛发展和大语言模型(large language models, LLMs)广泛应用,智算中心的网络面临着严峻挑战,流量控制是优化网络性能的重要方法.综述细粒度流量控制领域的关键问题及解决方案,重点梳理了自适应负载均衡机制、主动式拥塞控制机制和乱序包重排机制3个方面的研究进展.自适应负载均衡能够有效规避网络内部的拥塞,主动式拥塞控制用于预防自适应负载均衡无法避免的最后一跳拥塞问题,而乱序包重排解决了自适应负载均衡过程中可能引发的数据包乱序问题,三者协同作用确保了网络在高负载、高延迟等复杂环境下的稳定性与高效性.在此基础上,阐述了当前主流智算中心采用的关键技术方案以及目前支持细粒度流量控制的网络设备,最后总结了该领域尚未解决的关键问题及可能的解决方案,并对未来发展趋势进行了展望.

收稿日期: 2024-10-10; **修回日期**: 2025-03-21
基金项目: 北京市科技计划项目(Z241100004224026)
This work was supported by the Beijing Municipal Science and Technology Program (Z241100004224026).
通信作者: 王展(wangzhan@ncic.ac.cn)

关键词 负载均衡；拥塞控制；包喷洒；乱序重排；智算中心

中图法分类号 TP393.03；TP303

DOI: 10.7544/issn1000-1239.202440796 **CSTR:** 32373.14.issn1000-1239.202440796

AIGC[1](artificial intelligence generated content)是当前AI领域的一项前沿技术，可基于机器学习模型自动生成文本、图像、音频等多种形式的内容. 其核心在于强大的语言模型，比如大语言模型[2](large language models, LLMs). LLMs通过学习海量互联网文本数据，不仅能够理解语言的复杂结构，还能捕捉到文化、情感等深层次语境，进而生成连贯、有创意且贴近人类表达的内容. 这些模型的出现，极大地推动了AIGC技术的发展，例如OpenAI的ChatGPT在其发布后的3个月内服务了超过1亿活跃用户，是有史以来增长最快的应用程序[3].

随着LLMs参数规模的快速增长，大规模并行训练应用得越来越广泛，网络性能在AIGC技术的发展中扮演着至关重要的角色. 2020年，GPT-3以1 750亿参数量成为规模最大的模型[4]. 2023年，GPT-4模型已经增加到1.76万亿[5]，并且还会持续增长. 按照神经网络缩放定律[6](scaling law)，模型参数规模越大，性能也随之增长. 这些日益增长的大模型需要越来越大的集群进行训练，以GPT4为例，其训练集群的规模达到了数万卡级别. 网络对于大模型的训练和推理至关重要. 例如分布式训练过程中包括大量的计算和通信操作，下一阶段计算的开始时间取决于GPU[7]，NPU[8]，TPU[9]等之间通信阶段的完成时间[10]，集合通信的尾延时(即最后到达的消息)会阻塞所有计算单元的计算过程，进而降低训练性能. 根据中国移动通信研究院发布的《面向AI大模型的智算中心网络演进白皮书》[11]估算，以1 750亿参数规模的GPT-3模型训练为例，从理论估算模型分析，当网络动态时延从10 μs提升至1 000μs时，GPU有效计算时间占比将降低约10%. 因此，通信优化在AIGC中扮演着至关重要的角色，直接影响模型训练的速度和效率，也会降低昂贵的计算资源利用率.

流量控制是提高网络性能的一个重要技术方向. 衡量网络性能的重要指标是流完成时间[12](flow completion time, FCT)，即一个数据流从发送端开始到接收端完全接收完毕所需的总时间. 网络优化的主要目标是不断降低平均FCT和尾部FCT，其中负载均衡[13-14]是流量控制最有效的一个优化方向，负责将网络流量均衡地分配到交换链路中，提高整体网络转发效率，避免部分转发路径过载造成网络拥塞，而其他转发路径空闲的情况，其中的网络拥塞是影响FCT动态变化的关键因素.

当前网络中应用的负载均衡方法未充分考虑LLMs训练的通信需求. LLMs训练的通信负载具有3个典型特征[15]：

1)低熵. 与传统数据中心工作负载相比，AI工作负载的流量数量和多样性更多，流量模式的可重复性和可预测性较强.

2)突发性强. 流量传输通常表现出“开关”特性，即短时瞬发传输和网络闲置状态以毫秒级为粒度交替出现.

3)大象流占比大. 每次突发传输的数据量较大，线速(如400 Gbps)传输几秒至几十秒，更容易造成交换机内部队列堆积，增加排队延迟.

传统数据中心采用的负载均衡策略是等价多路径[16](equal cost multi path, ECMP)，该方法对数据流中携带的五元组信息(源IP，目的IP，源端口号，目的端口号，协议)进行哈希运算，根据哈希结果选择可用路径，从而将流量随机分布到可用路径上. 由于该方法既没有考虑网络状态的动态变化，也没有考虑不同流数据量的差异化特征，因此，当发生哈希冲突时，可能会出现多个突发大象流被调度到相同链路上的情况，引起链路负载极度不均衡，最终造成部分路径过载而产生网络拥塞，一方面降低了网络整体利用率，会浪费超过50%的对分带宽[17]，另一方面造成了排队时延增加，尾部FCT恶化10～100倍.

与传统数据中心承载企业、政府和个人等多种应用不同，智算中心重点承载AI训练/推理应用，因此，研究面向AI训练/推理负载特征的网络优化将是重点方向.

近年来，以数据包/子流(flowlet)为代表的细粒度负载均衡机制逐渐受到广泛的关注. 通过将流进行逐包或者切分成子流进行网络调度，可充分利用互连网络的“多路径”优势，与图1中粗粒度，即以流的粒度进行路由(属于一条流的所有数据包都沿着相同路径转发)相比，图2中细粒度的负载均衡方法(属于一条流的数据包可以在所有等价路径上转发，即“包喷洒”方法)可显著减少网络热点，更适用于具

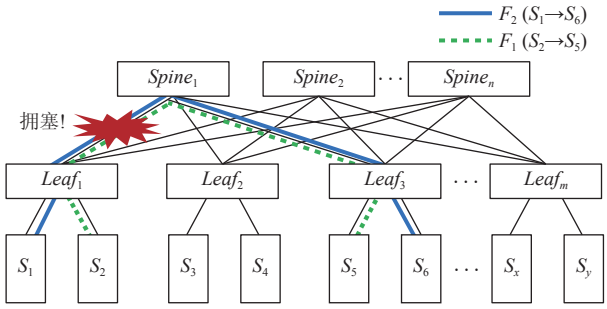


Fig. 1 Coarse-grained load balancing architecture

图 1 粗粒度负载均衡架构

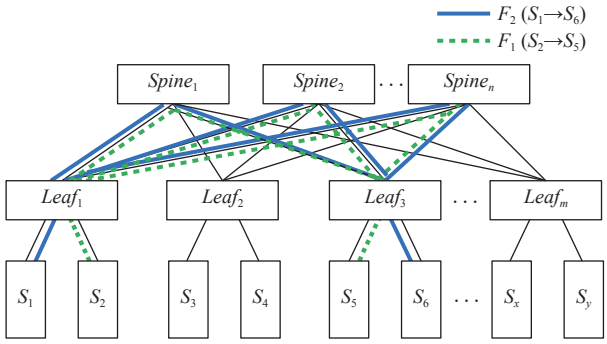


Fig. 2 Fine-grained load balancing architecture

图 2 细粒度负载均衡架构

有“大象流占比大”特征的 LLMs 训练.

2023 年 9 月, 为更好地支持 AI 训练, 超以太网联盟(Ultra Ethernet Consortium, UEC)[18] 应运而生, 汇聚了包括 AMD、Arista、Broadcom、Cisco、Eviden(Atos 旗下)、HPE、Intel、Meta 及 Microsoft 等在内的具有影响力的企业, UEC 的一个重要特色就是引入多路径传输与数据包粒度的负载均衡方案, 也推动“包喷洒”技术从学术研究逐步进入产业界. 在国内, 2024 年由阿里云公司和中国科学院计算技术研究所联合发起成立了高通量以太网联盟[19], 旨在兼容传统以太网协议的同时, 重点解决负载不均、可扩展性不足、性能欠佳等问题, 加速 AIGC 应用.

虽然采用逐包或者子流的细粒度负载均衡机制可显著改善网络性能, 与传统 ECMP 方法相比, 可以降低尾部 FCT 1/4～1/2[20], 但是当前包喷洒机制仍然存在 3 个关键问题亟需解决.

问题 1. 当网络发生链路/设备故障时, 网络拓扑会变得不对称, 若仅采用随机均匀喷洒的方案, 仍然会造成与故障相关的路径拥塞[21], 如图 3 所示, 流 F_1 和流 F_2 同时向 S_{10} 发送数据时, 由于 $Leaf_1$ 和 $Spine_1$ 链路故障, F_1 仅剩 2 条路径传输, 而 F_2 有 4 条路径, 均匀喷洒会导致在 $Core_3$—$Spine_6$ 和 $Core_4$—$Spine_6$ 链路上流量大于 $Core_1$—$Spine_5$ 和 $Core_2$—$Spine_5$ 链路上流量, 发生拥塞.

问题 2. 采用“包喷洒”等技术不可避免最后 1 跳拥塞[22]. 虽然显著避免交换网络上行链路(如图 2 中的 Leaf 到 Spine 链路)的拥塞, 但是当网络有多到 1 的流量时(如磁盘故障恢复[23]、LLMs 训练中的规约操作[24] 等), 最后 1 跳拥塞是不可避免的, 如图 3 中所示, 假设 F_1 和 F_2 都以 400 Gbps 速率向 S_{10} 发送数据, 由于 S_{10} 的接收带宽为 400 Gbps, 将难以避免 $Leaf_5$—S_{10} 的链路拥堵.

问题 3. 多路径传输会造成包乱序问题. 同一条流的不同数据包被调度到不同链路上转发, 由于不同路径的延时存在差异, 会导致到达接收端的数据包出现乱序, 如图 4 所示. 然而现有通信协议(RDMA, TCP 等)并不支持数据包的乱序提交, 会认为网络发生了丢包而进行降速.

针对上述细粒度负载均衡面临的关键问题, 本文第 1~3 节分别探讨自适应细粒度负载均衡方法、主动式拥塞控制以及乱序包重排序等关键技术, 这 3 项技术相互补充, 形成一个完整的细粒度负载均衡分析框架及解决方案. 具体而言, 自适应负载均衡方

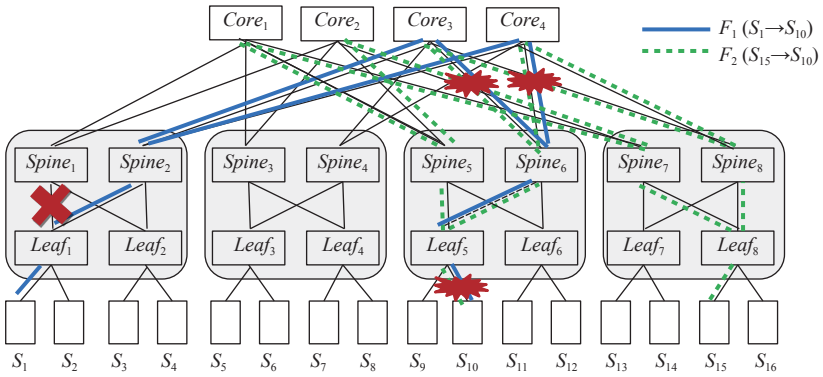


Fig. 3 Congestion issues caused by packet spraying in heterogeneous topologies mechanism and the many-to-one flow.

图 3 不对称拓扑包喷洒机制和多到 1 流量引起的拥塞问题

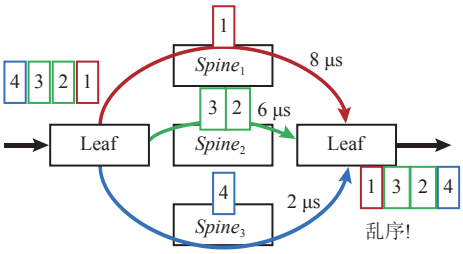


Fig. 4 Packets arrive out of order at the receiver due to varying delays on different paths

图 4 不同路径延时不同导致的数据包到达接收端时乱序

法能够有效规避网络内部的拥塞，主动式拥塞控制则用于预防自适应负载均衡方法无法避免的最后 1 跳拥塞问题，而乱序包重排序机制则解决了自适应负载均衡过程中可能引发的数据包乱序问题. 三者的协同作用确保了网络在高负载、高延迟等复杂环境下的稳定性与高效性. 本文同时总结了当前智算中心的部署方案和支持细粒度负载均衡技术的商业网络设备(见第 4 节)，并展望了未来网络优化的趋势(见第 5 节).

1 自适应细粒度负载均衡

传统数据中心通常采用基于流粒度的负载均衡策略，如 ECMP，但该策略容易导致哈希冲突，从而引起链路负载极度不均衡，进而引发网络拥塞. 而细粒度负载均衡方法能够充分利用互连网络的“多路径”优势，将同一流的不同数据包分散到多个路径上，有效减少网络中的热点并优化流量分布. 然而，当网络发生链路或设备故障时，网络拓扑会变得不对称，导致与故障相关的路径上仍然可能出现拥塞问题. 因此，如何避免拥塞、实时感知拥塞并进行自适应路由，是本节的研究重点.

本节将介绍主流的自适应细粒度(包括 flowlet[25] 粒度和包粒度)负载均衡机制，其主要分为基于交换侧的负载均衡机制和基于主机侧的负载均衡机制 2 类.

1.1 交换侧的负载均衡

交换侧的负载均衡是由交换网络中每一跳交换机按照数据包/子流的粒度进行路径选择，可以充分利用网络链路资源，尽可能避免在网络中造成拥塞.

2012 年 DeTail[26] 提出了数据包分发(packet scatter)的思想，证明了当有多个最短路径可用时，基于包粒度的随机转发效果显著优于基于传统的流粒度方法，将第 99.9 百分位的 FCT 缩短了一半.

但是当在拓扑不对称的情况下，包分发性能可能明显差于 ECMP 方法. 以一种常见的故障类型举例，即当核心交换机和汇聚交换机之间的 1 Gbps 链路降级到 100 Mbps 时，包分发的性能比 ECMP 差大约 12%. DeTail 提出基于拥塞情况进行负载均衡决策，当数据包到达交换机时，交换机选择队列中字节计数最小的输出端口进行转发，从而将数据包转发到最不拥塞的最短路径上. 考虑到交换机有大量的转发端口以及计算每个输出端口队列的计数器成本过高，DeTail 采用基于阈值的信号，每当端口优先级队列的字节计数器的值低于预定义的阈值时，交换机就会使能相关信号，将此队列加入可输出队列列表. 当有多个端口的队列列表符合时，DeTail 随机选择其中一个端口.

属于一条流的数据包可能在具有不同延迟的路径上传输，后面发送的数据包可能比之前发送的先被接收，造成到达接收端乱序的情况. 由于 TCP 协议不能区分因多路径导致的数据包乱序和数据包丢失的情况，它将触发拥塞避免机制、缩减拥塞窗口大小，导致传输带宽下降，性能变差. 为避免此类情况，DeTail 禁用传输层对乱序数据包的监控和反馈机制，以免被传输层协议栈误认为发生了丢包，在无损网络下，网络会开启优先级流控(priority flow control, PFC)机制[27]，丢包情况非常罕见，因此在传输层直接禁用丢包检测不会引起可靠性问题.

尽管使用包喷洒能显著提升网络性能，而且许多商品交换机(如 Cisco[28])已经实现了数据包喷洒(packet spray)技术，但是由于重新排序及其对 TCP 传输层协议的影响，网络研究人员以及从业者通常对在数据中心应用数据包喷洒技术持有谨慎态度. RPS[21](random packet spraying)于 2013 年被提出，RPS 与 DeTail 提出的数据包分发(packet scatter)含义相同，RPS 通过实验证明了包喷洒不太可能受到数据包重排序的影响. 首先，基于多根树形拓扑的网络往往是对称的，如果所有网络中的流都使用 RPS，那么每个等价路径中的所有链路将承载相同的负载. 所以，这些路径之间的延迟差异以及重排序的数据包数量预计会非常小.

当网络拓扑对称性被破坏时(如发生链路故障)，RPS 采用主动式队列管理[29]方案随机早期丢弃[30](random early discard, RED)造成不同路径上队列长度差距的数据包，使不同路径上的延时保持在较低水平，这样不同路径上转发的数据包在到达目标端后，乱序程度依旧会较低. 为了避免数据包乱序引起网

络性能下降问题, RPS 设置发送方重复 ACK(duplicate acknowledgment, DupACK)阈值(通常是 3), 当重复 ACK 数目不超过阈值时, 不触发拥塞控制, 发送窗口不会改变, 不降低发送速率.

RPS 方案在以下方面仍存在不足.它对较低网络负载情况进行了实验论证, 各路径的延时差异较小, 但是未对网络高负载情况下各路径延时差异、包乱序程度进行讨论.

为了减少包喷洒可能引入的严重乱序问题, Cisco 公司于 2014 年提出 CONGA[31], 它采用了子流的粒度进行路由, 这一概念最早由 Kandula 等人[32] 提出. 子流属于同一条流, 如果 2 个数据包之间的空闲间隔大于路径间的最大延迟差, 则第 2 个及之后的数据包可以沿着与第 1 个不同的路径发送, 形成子流, 子流间隔足够大时, 不会导致数据包乱序. 因此, 子流为负载均衡提供了一种更高粒度的替代方案, 避免了重排序问题. CONGA 是一个感知全局拥塞和源 ToR (top of rack)交换机到目的 ToR 交换机之间以及子流的负载均衡方案. 架构图如图 5 所示.

CONGA 利用转发的数据包来收集全局拥塞信息. 数据包 VXLAN[33] 协议中的字段携带以下信息: *LBTag*(4 b)(标识前向转发路径序列号); *CE*(3 b)(拥塞程度, 与链路利用率正相关); *FB_LBTag*(4 b)(指示返回转发路径序列号, 此域由目标 ToR 交换机使用, 负责将拥塞信息回传给源 ToR 交换机); *FB_Metric*(3 b)(表明路径相应的拥塞程度).

CONGA 路由取决于每个上行链路的转发路径的拥塞程度, 在源 ToR 交换机进行路由决策时, 会对每个 flowlet 第 1 个包进行路由, 选择负载最轻的路径, 属于同一个 flowlet 的数据包将按照第 1 个包的路径转发. 由于 CONGA 针对 flowlet 进行路由, flowlet 之间空闲间隔大, 在接收端数据包之间不会乱序. 而且收集全局负载信息, 可以有效应对链路故障/降级等引起的网络不对称问题. 在测试床实验中, 在单个链路故障的情况下, CONGA 的 FCT 比应用 ECMP 快 5 倍.

CONGA 方案在以下方面仍存在不足.源 ToR 交换机到目的 ToR 交换机之间收集链路负载信息的控制环路长, 而且当流出现长的空闲间隔, 即 flowlet 出现时, 才能重路由, 虽然有助于最小化数据包重新排序, 但它不够灵活, 不能及时应对网络突发流量. 而且 CONGA 不能识别到交换机故障, 它依靠利用率来判断链路空闲, 但是当遇到交换机发生黑洞[20] 现象时, 可能会误认为链路空闲, 导致更多的数据包丢失.

为了及时应对网络流量的微突发, 2017 年 Ghorbani 等人[22] 提出了 DRILL, 指出数据包丢失的情况大多数由微突发造成, 而且微突发仅持续数微秒. 而基于全局拥塞信息收集和反应的系统如 CONGA 通常具有几个往返时延(round trip time, RTT)的控制回路, 远远慢于数据中心中大多数拥塞事件的持续时间, 因此 DRILL[22] 提出基于本地交换机信息的路由方案, 将路由决策放到每个单独的交换节点上, 实现微秒级的负载均衡.

DRILL 是一个基于分布式、本地信息的负载均衡方案. 受 power of two choices [34] 模式启发, DRILL[22] 提出 DRILL(d,m) 算法, 在每个数据包到达时, 数据包的转发引擎从所有可能的输出端口中随机选择 d 个路径, 再加上前几个时间段最轻的 m 个负载路径, DRILL(d,m) 指从这($d+m$)个样本中找到当前队列占用最小的端口, 将数据包路由到该端口. 该算法的时

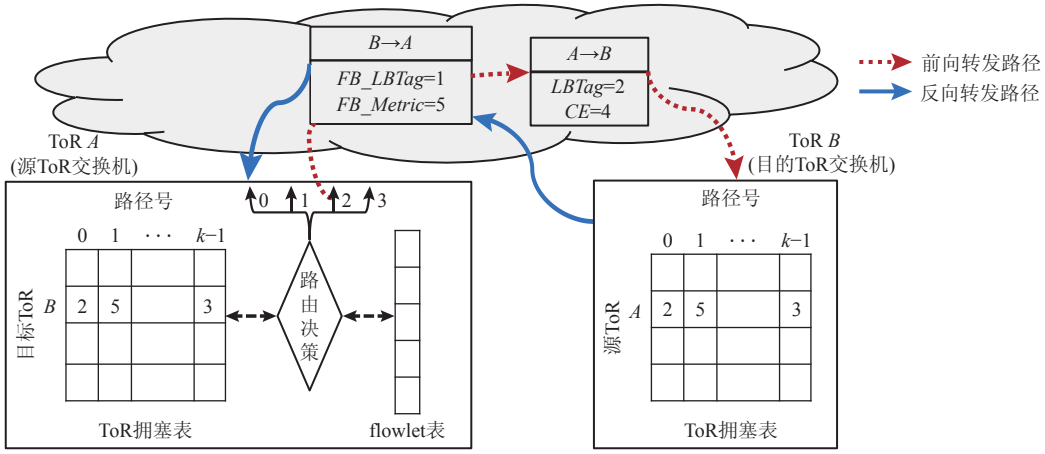


Fig. 5 CONGA architecture

图 5 CONGA 架构

间复杂度为 $O(d+m)$. 考虑到存储和算法时间复杂度，以及通过实验论证，DRILL 指出 DRILL(2,1) 优于或接近 d 和 m 为更大的值的方案. 实验证明在超额订购比为 1∶1 的胖树拓扑以及流量在 80% 负载下，DRILL 与 CONGA 相比，平均 FCT 降低了 40%.

在不对称网络拓扑情况下，DRILL 将不同的等价路径划分为多个对称组，组内各路径负载相似，在进行路由时，对称组间用哈希进行随机选择，再进一步选择组内的路径时，采用随机路由的机制.

为了应对乱序包，DRILL 通过设置 DupACK 门限值来避免 TCP 传输层误认定位发生了丢包. 实验证明，重新排序程度很少达到 TCP 重传阈值，即使在重负载 80% 下，只有 0.02% 的流超过了典型的阈值为 3 的 DupACK.

DRILL 方案在以下方面仍存在不足. 在不对称网络拓扑情况下，交换机需要为每个网络设备计算负载的权重来构建对称组，这往往需要通过本地控制软件或者中央控制器完成，构建对称组时延会很高，并不能及时应对微突发导致的丢包或者拥塞.

以上的细粒度负载均衡方案，均针对 TCP 协议，然而，由于 RDMA 高性能互连协议具有低时延的特性，目前也广泛应用在数据中心. 在 RDMA 网络中采用针对 TCP 协议提出的负载均衡方案会造成严重的性能下降，这是由于目前支持 RDMA 的 NVIDIA ConnectX 系列[35] 等主流商业网卡无法区分数据包乱序和网络拥塞引起的丢包，当网络采用多路径路由方案引起乱序时，网卡会误认为由于网络拥塞发生了丢包从而开始降低发送速率，导致吞吐下降. 为了解决此类问题，新加坡国立大学在 2023 年提出了 Conweave[20]，提出谨慎的路由（1 条流最多同时在 2 个路径上传输）和重排序机制，架构图如图 6 所示.

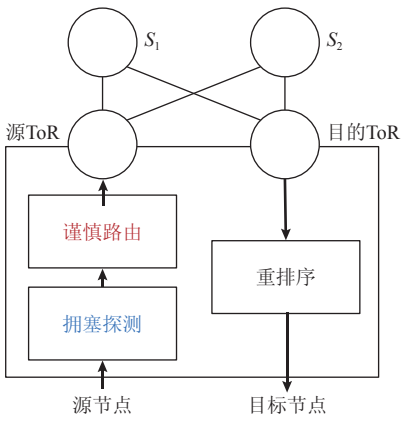


Fig. 6 Conweave architecture

图 6 Conweave 架构

Conweave 在目标 ToR 交换机上实现了重排序功能，对端侧网卡/主机屏蔽了乱序问题. 由于现代数据中心网络交换机通常仅具有一定数量的队列和缓冲区，因此 Conweave 提出属于 1 条流的数据包最多同时在 2 个路径上传输，这样仅需要 2 个队列即可完成重排序.

Conweave 会对全局网络进行拥塞探测并进行谨慎的路由. 源 ToR 交换机会周期性地向目标 ToR 交换机发送 RTT 请求，根据回复时间来判断网络拥塞情况. 如果 RTT 请求超时或者源 ToR 交换机收到目标 ToR 交换机的拥塞指示，则触发谨慎的路由机制. Conweave 根据 3 个条件执行重路由：1）现有路径拥塞. 2）存在一个未拥塞的可行路径. 3）由先前重路由引起的任何无序数据包都已经被目的 ToR 接收. 条件 1）和 2）是为了确保需要重路由并且有一个更优的替代路径可用. 条件 3）是为了产生可预测的到达顺序，在任何时候任何流都只能在最多 2 路径上有飞行数据包. Conweave 在平均 FCT 和第 99 百分位 FCT 上分别实现了最高 42.3% 和 66.8% 的性能提升.

Conweave 方案在以下方面仍存在不足. 任意时刻，每个流最多只能在 2 路径上有在途数据包，限制了路径多样性. 此外，此工作未对不对称拓扑下的性能效果进行评估.

1.2 主机侧的负载均衡

除了在交换机侧进行负载均衡方案，主机侧可以通过修改数据包头中的五元组（源 IP，目的 IP，源端口号，目的端口号，协议），也可以改变数据包在交换网络中的传输路径以实现负载均衡.

在主机侧，针对 TCP 协议，Raiciu 等人[36] 最早在 2010 年提出了多路径 TCP（MPTCP）. 在数据中心中采用 MPTCP 替代传统 TCP，能有效利用数据中心中可用带宽来提升吞吐量，并在多种拓扑结构下实现更好的公平性. MPTCP 使单个 TCP 连接可以通过多条网络路径传输数据，在连接建立时，主机间会协商确定 MPTCP 的支持情况，也会确认是否具备额外的 IP 地址. 如果支持额外 IP，子流使用不同的 IP 地址进行区分；如果不支持，子流使用不同端口号来进行区分. 在交换网络侧选择转发路径时，默认使用 ECMP 路由将子流散列到不同路径上.

应用程序无需感知 MPTCP 的使用，且无需修改便可在支持 MPTCP 的 Linux 内核上直接运行. 每个 MPTCP 子流拥有独立的序列号空间，并维护自己的拥塞窗口，以适应不同路径的网络条件. MPTCP 通过在亚马逊 EC2 云平台上的 Linux 环境中进行验证，发

现 2/3 的流量可以使用多路径, 使用 MPTCP 的吞吐量约为常规 TCP 的 3 倍.

MPTCP 虽然可以显著改善网络利用率, 但是部署起来具有挑战性, 因为网络运营商通常无法控制终端主机栈(如在公共云中), 即便能够控制, 有些高性能应用(如低延迟存储系统)也会绕过内核, 实施自己的传输协议.

MPTCP 是基于子流的负载均衡方案, 主机侧也可以实现数据包粒度的方案, 2017 年 Zhang 等人[37]提出 Hermes, 支持灵活应对网络突发流量、检测和处理黑洞和随机数据包丢失等交换机故障[38]、谨慎的重路由方案, 架构图如图 7 所示.

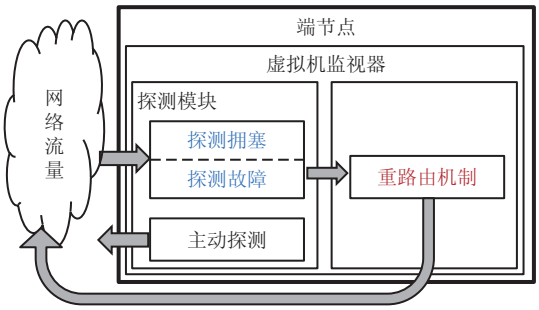


Fig. 7 Hermes architecture

图 7 Hermes 架构

Hermes 同时利用 RTT 和 ECN[39](explicit congestion notification)探测拥塞, 将路径划分为拥塞、良好和灰色状态. 当 ECN 被标记频繁且 RTT 高的时候, 路径状态为拥塞; 当 ECN 被标记少且 RTT 低的时候, 路径状态为良好; 其他组合情况, 如 ECN 被标记频繁且 RTT 低、ECN 被标记少且 RTT 高或者适中, 则路径状态为灰色的状态. Hermes 检测黑洞和随机数据包丢失等常见故障[29,40], 避免路由转发至此路径. 黑洞故障指的是交换机丢弃某个特定源-目的 IP 地址对的数据包. 检测数据流的超时时间, 一旦一条路径上发生超时超过 3 次, 而且没有收到过 ACK 确认包, 则认定发生了黑洞故障. 它通过检测丢包重传机制来识别, 当路径上的数据包重传次数超过一定比例(如 DCTCP 协议时为 1%), 且当前路径是不拥塞路径时, 则认定为发生了随机包丢失故障.

Hermes 采用及时但谨慎的重路由机制, 采用以数据包为粒度可以及时应对可能发生的故障或拥塞. 但是由于拥塞不匹配问题以及可能造成严重的乱序问题, 它会综合考虑路径条件以及流的完成状态. 只有当收益大于代价时, 才会进行路由切换. 具体来说: 第一, 如果新路径的流发送速率比原路径的速率高得并不明显, 则收益很小, 只有当速率高得很明显时, 才会切换; 第二, 如果发现要重路由的数据流即将发送完毕, 则收益也很小, 不切换路径; 第三, 如果原路径的发送速率已经很高, 那么切换新路径的收益会很小, 也不切换路径.

Hermes 可以有效地避免发生拥塞不匹配和乱序的问题, 但是由于采用端到端的反馈机制, 反应会慢于在交换机侧进行负载均衡的方法, 而且其重路由机制过于保守, 容易导致链路利用率低等问题.

尽管已有多种支持多路径路由的方法可应用于 TCP 协议, 但这些方案并未考虑 RDMA[41] 协议. 与 TCP 不同, RDMA 的协议处理逻辑完全在网卡(network interface card, NIC)硬件中实现, 计算资源和片上内存非常有限, 多路径 RDMA 传输的关键设计目标是最小化内存占用, 这带来了 3 个挑战: 1)路径拥塞状态跟踪. 多路径传输需要跟踪每条路径的拥塞状态, 然而随着路径数量的增加, 这些状态跟踪的内存需求也线性增长. 即使对一个 RDMA 连接使用有限数量的路径, 也会引发显著的内存开销. 2)乱序数据包处理. 多路径会导致数据包在接收端乱序到达, 接收端需要额外的元数据来跟踪哪些数据包已经到达. 如果路径条件差异较大, 所需的元数据会显著增加. 3) 内存缓冲限制. 接收端的 NIC 内存不足以缓冲乱序的数据包, 因此必须在数据包到达时直接写入主机内存. 这会带来同步问题, 例如第 2 个 WRITE 操作先于第 1 个 WRITE 操作到达, 导致数据更新乱序.

为应对上述挑战, 2018 年 Lu 等人[42] 提出 MP-RDMA, 主要解决如何高效利用有限的硬件资源, 同时避免由于多路径传输带来的乱序和同步问题.

MP-RDMA 不维护每条路径状态, 对所有路径使用 1 个拥塞窗口, 从而最小化片上内存的使用. 如图 8 所示, 拥塞控制算法基于 ECN, 在 IP 头中标记 *CE* (congestion experience), 与传统计算平均 *CE* 比率来估计拥塞水平的方法不同, MP-RDMA 直接对每一个 ACK 包的 *CE* 作出反应. 由于在 RDMA 网络中数据包很少被丢弃, 对每个 ACK 作出反应是精确且可靠的.

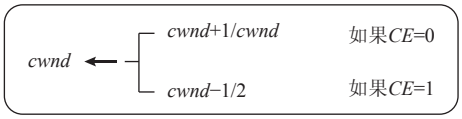


Fig. 8 MP-RDMA congestion window adjustment equation

图 8 MP-RDMA 拥塞窗口调节方程

MP-RDMA 在接收方使用位置图数据结构来跟踪到达的数据包. 每个位置包含 2b 数据包. 根据消息类型, 一个插槽可能是下面 4 种状态之一: 1)空, 相

应的数据包尚未接收. 2)已接收，相应的数据包已接收，但不是消息的尾部(最后一个)数据包. 3)尾部，接收到的数据包是消息的尾部数据包. 4)尾部且需要完成通知. 接收到的数据包是消息的尾部数据包，需要完成通知. 例如, SEND 或 READ 响应. MP-RDMA 仅使用有限的插槽 L 在跟踪位图中，例如，$L=64$, 以减少 NIC 硬件中的内存占用.

MP-RDMA 通过路径选择算法控制乱序数据包的乱序程度(out of order degree, OOD)，核心思想是主动“修剪”掉慢速路径，并只选择快速路径. 此外, MP-RDMA 添加新变量 *snd_ooh* 和 *snd_ool* 来识别慢速路径，*snd_ooh* 记录了发送端接收到的 ACK 的最高序列号，$snd_ool = snd_ooh - \Delta$, 其中 $\Delta \leqslant L$ 是一个可调参数，用于确定 MP-RDMA 的乱序程度，如果发送方检测到确认包序列号低于 *snd_ool*, 则识别到慢速路径，需要将数据包切换到快速路径中.

综上，与单路径 RDMA 相比，MP-RDMA 仅在每个连接状态上增加了 66 B, 此外，实验表明，MP-RDMA 可以在故障情况下显著提高鲁棒性(在 0.5%～10% 的链路丢包率下，吞吐量提高 2～4 倍)，并提高整体网络利用率高达 47%.

1.3 小 结

我们总结了基于交换机和基于主机的负载均衡机制的代表性方法，且对比了 2 类方法的优缺点，如表 1 所示.

2 主动式拥塞控制

数据中心中传统拥塞控制方法，例如 DCTCP[43], DCQCN[44], HPCC[45], TIMELY[46], Swift[47] 等，是“反应式”的方法，这类方法通常只有在丢包或队列长度超过预设阈值时，才会触发流量速率调整，存在滞后性，属于被动式拥塞控制. 对于 LLM 训练中的网络突发流量，传统拥塞控制无法及时响应，数据包已在队列中积累，排队延迟增加. 这种传统方法不适用于频繁进行参数同步的 LLM 训练场景，因为单个流的长尾延时会显著拖慢整个训练过程的效率.

为了满足日益增长的性能要求，近年来很多工作提出“主动式”拥塞控制，这类方法是一种预防性的方法，在拥塞发生之前预测和防止. 主动式拥塞控制可以与第 2 节所介绍的细粒度的负载均衡机制相配合，解决细粒度的负载均衡机制不能避免的最后 1 跳拥塞问题. 主动式的拥塞控制的关键思想是采用“请求和分配”的模式.

2.1 主机侧的拥塞控制

主机侧的拥塞控制完全在主机侧实现，发送端发起通信“请求”，接收端回复“分配”信息，无需交换机参与.

推测性保留协议[48](Speculative Reservation Protocol, SRP)就是基于主机侧实现的“请求和分配”模式的拥塞控制算法. SRP 要求发送端和接收端之间的通信之前需要进行请求授权握手，以避免接收端过度预订. 同时为了减少握手引起的延迟开销，发送者可以在请求授权返回之前开始推测性地传输数据包. 如果网络中发生拥塞，网络可以丢弃推测性数据包. 如图 9 SRP 原理图所示，显示了 SRP 正常操作的时间线. 请求包中包含发送大小 n, 指示发送端打算传输的数据包数量. 发出请求后，发送端开始向接收端推测性地发送数据包 P_1 和 P_2. 这些推测性数据包在低优先级虚通道上传输，并具有有限的等待时间(time to wait, TTW)，如果数据包在网络内部的总累积排队时间超过其 TTW, 路由器会丢弃推测性数据

Table 1 Comparison of Different Load Balancing Solutions

表 1 不同负载均衡方案对比

分类	方法	粒度	创新点	协议	优点	缺点
交换侧	DeTail[26]	packet	选择本地交换机字节计数最小的输出端口转发	TCP	可以实时感知到网络状态变化，及时进行重路由； 与现有商业主机端网卡兼容.	需要修改现有交换机芯片设计，芯片迭代慢，难以立即应用.
	RPS[21]	packet	随机喷洒数据包，当网络不对称时，随机早期丢弃数据包	TCP		
	CONGA[31]	flowlet	探测全局拥塞程度，对每个 flowlet 第一个包进行路由决策	TCP		
	DRILL[22]	packet	根据本地交换机信息路由，从 $d+m$ 个路径中选择	TCP		
	Conweave[20]	packet	探测全局拥塞，任何流都只能在最多 2 条路径上传输	RDMA		
主机侧	MPTCP[36]	flowlet	将 1 条流分割成多个子流	TCP	通过修改主机端协议栈实现，可实际部署； 与现有交换机兼容.	探测网络状态需要至少 1 个 RTT，重路由机制不及时； 所有主机同时修改协议栈，对于数据中心多租户场景来讲，部署困难.
	Hermes[37]	packet	同时利用 RTT 和 ECN 来检测路径状态，收益大于代价时，才进行路由切换	TCP		
	MP-RDMA[42]	packet	识别慢速路径，主动删除慢速路径	RDMA		

包. 一旦请求包到达接收端, 接收端就返回一个带有起始时间t_s的小型授权包G(t_s), 此外, 接收端更新其请求时间表, 以便任何下一个发送端请求包请求时间不早于$t_s+n(1+\epsilon)\tau_p$. 其中τ_p是在接收端接收单个数据包的时间, 参数ϵ考虑了控制包的带宽开销. 当发送端接收到授权包时, 它停止向接收端推测性地传输. 在达到时间t_s后, 发送端以非推测性模式恢复向接收端传输, 非推测性数据包不能被丢弃, 也不需要确认应答.

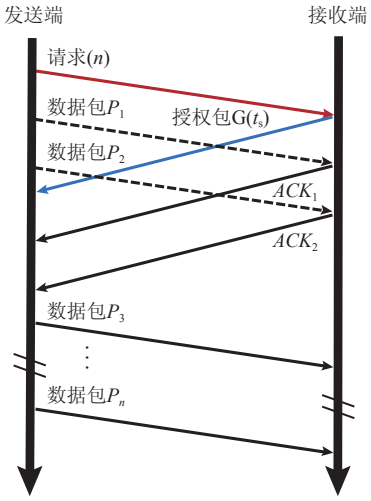


Fig. 9 SRP mechanism diagram

图 9 SRP 机制图

实验表明, 与传统“反应式”拥塞控制方法 ECN 相比, SRP 几乎可以立即对刚开始的拥塞做出反应, 对于突发性流量模式, SRP 通过有效地解决突发流量造成的瞬态热点, 实现了比 ECN 更高的饱和吞吐量.

除了基于分配时间片段的方案, 也可以采用分配令牌的方法, 不严格限制发送开始时间, 如 2015 年 pHost[49] 的提出是受到了无线协议(例如, 802.11)[50] 和 capability-based DDoS[51] 机制的启发. pHost 原理图如图 10 所示.

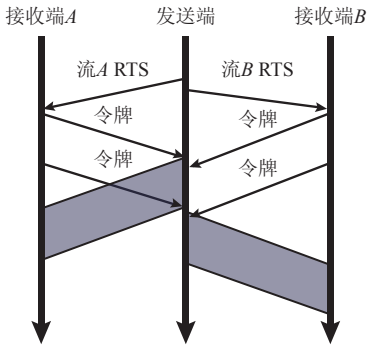


Fig. 10 pHost mechanism diagram

图 10 pHost 机制图

pHost 的设计主要分为 5 个步骤: 1)每个源端主机在流到达时, 向流的目的地发送一个请求发送(requests to send, RTS)数据包、流 A RTS 和流 B RTS. RTS 包含与调度决策相关的信息(如流的大小、租户编号等). 2)目标主机会查看待处理的 RTS 集合, 并向一个源端发送“令牌”. 该令牌允许源端主机发送一个数据包, 并可能指定其优先级. 3)如果源端在收到令牌后的短时间内未使用令牌, 则令牌将过期, 过期时间默认为 1.5 倍最大传输单元(maximum transmission unit, MTU)的数据包传输时间. 为了避免第 1 个 RTT 的带宽浪费, 每个源端主机还可以为每个流分配一些“自由令牌”, 这些令牌不需要从目的地主机获取. 4)源端主机从其未过期的令牌中选择 1 个, 并发送相应的数据包. 5)目标主机接收到流的所有数据包后, 会向源端主机发送 1 个 ACK 数据包. 在整个流程中, 所有控制数据包(如 RTS、令牌、ACK 包)都以最高优先级发送.

分配令牌的方式可以按照多种调度策略来进行, 如最短剩余处理时间(shortest remaining processing time, SRPT)优先策略、截止日期受限[52](deadline-constrained)优先或跨多租户的公平性调度策略等, 但由于 pHost 只使用 2 个优先级, 影响了其近似最短剩余处理时间优先的能力, 导致小消息的排队延时增加. 它静态分配一个高优先级用于所有第一个 RTTbytes 的数据包传输, 以及一个较低的优先级用于所有已调度(即已分配令牌)的数据包传输. 对于大于 RTTbytes 的小消息, 不能高效地抢占大消息链路资源.

为了进一步优化小消息的延时, Homa[53] 于 2018 年被提出, 它同样是基于请求和分配信用的机制, Homa 指出现有的数据中心小消息占比大, 已有研究并没有充分利用网络内部的优先级, 不能在高网络负载下为小消息提供硬件级最低延迟. Homa 提出在接收方分配信用的同时, 充分利用现代网络交换机提供的优先级队列, 按照剩余最短处理时间优先原则分配高优先级, 最大程度地利用交换机中的优先级队列. 由于小消息的剩余处理时间非常短, 因此会被分配较高优先级, 在交换机进行队列调度时, 会被最先服务, 从而保证了小消息的低延时.

为了避免请求和授权过程引入的延迟开销, Homa 允许在第一个 RTT 内发送 RTTbytes(1 个 RTT 内线速发送的数据量)的数据包, 此部分数据包的优先级会被分配为较高优先级. 如图 11 所示, 其中 Pri_7 是最高优先级队列, Pri_0 是最低优先级队列, 发送端 2 的消息 m_2 的第 1 个 RTTbytes 部分被放到目标 ToR

交换机出端口队列中更高的优先级队列，而发送端的消息 m_1 的已授权部分放到低优先级队列，高优先级队列会被优先调度出队，最先到达接收端.

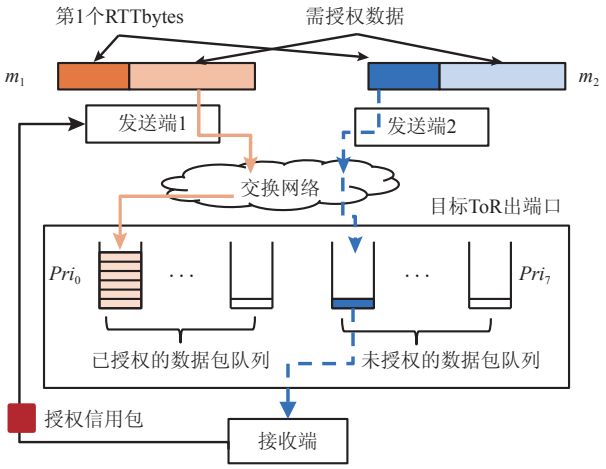


Fig. 11 Homa mechanism diagram

图 11 Homa 机制图

如果 1 个发送端要向多个接收端发送数据，接收端会相继授权，但是发送端不能以线速同时向所有接收端发送数据，否则会造成部分接收端的带宽浪费，因此接收端在授权时，Homa 使用“过度授权信用”. 多授权一些，可以有效地利用网络带宽. Homa 在 10 Gbps 网络 80% 负载下，降低了短消息第 99 百分位的往返时延到 15 μs 以下，是现有研究的延迟的 1/100.

尽管主动式的拥塞控制可以显著改善链路利用率和降低延迟，但有一个关键挑战是申请授权至少需要 1 个 RTT 获取信用，这个 RTT 阶段被称为“前信用(pre-credit)”阶段，为了不浪费带宽，ExpressPass 和 Homa 会允许发送端以线速发送数据包，此时，网络链路上可能承载大量流量，导致网络拥塞，现有的主动解决方案都没有适当处理它. 基于此，Aeolus[54] 于 2020 年被提出，它专注于“前信用”数据包传输的解决方案. Aeolus 按照已授权数据包优先(scheduled packet first, SPF)原则，降低了第 1 个 RTT 传输的数据包优先级，而不是像 pHost 和 Homa 分配最高优先级. Aeolus 旨在同时实现 3 个设计目标：1) 新流量充分利用备用带宽，避免较长的 FCT；2)保护已授权的数据包以保持主动传输的确定性；3)使其易于在生产数据中心部署，即 Aeolus 必须能够使用商品交换机实现.

Aeolus 在终端主机上允许流量在前信用阶段以线速开始，然后再根据收到的信用调整其发送速率. 为了保护已授权数据包，在网络中采用 SPF 原则，同时在交换机中引入了选择性丢弃机制，当端口拥塞时，选择性地丢弃非授权数据包. 通过这种方案，Aeolus 可以有效地利用剩余带宽进行已授权数据包的传输，同时不会破坏主动控制期望性能. Aeolus 将 ExpressPass 的平均 FCT 提高了 56%，将 Homa 的尾部 FCT 减少了 95%.

2.2 主机侧和交换侧同时参与的拥塞控制

虽然“请求和分配”的模式可以显著改善接收端的拥塞问题，但是此类方法不能避免中间的共享交换网络链路的拥塞问题. 基于此，ExpressPass[55] 提出交换机也参与信用包的速率限制，以确保发送的数据包数量不超过链路容量. 假设有 2 个流通过交换机的同一个链路发送，当 2 个不同流的信用包都以线速发送时，交换机会将 2 条流的一半的信用包丢弃掉，从而达到控制流量的目的.

如图 12 所示，以 2 个流的场景进行说明，图 12 中所有链路的带宽相等. 假设在一个时间窗口内，链路上只能传输 2 个数据包. 接收方 R_A 和 R_B 将分别以最大信用速率生成信用 (A_1, A_2) 和 (B_1, B_2). 所有 4 个信用包到达相同的输出端口，在输出端口通过对信用包进行速率限制以匹配反向链路的容量. 因此，一半的信用(如 A_2, B_2)在输出端口被丢弃. 最终每个发送者得到 1 个信用并发送 1 个数据包回到接收方. 在 2 个流的 RTT 相同的理想情况下，数据包不会经历任何排队，因为它们在瓶颈链路的带宽被交换机进行了理想的分配.

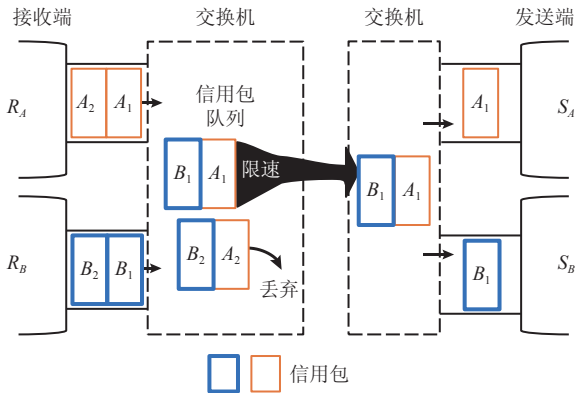


Fig. 12 ExpressPass mechanism diagram

图 12 ExpressPass 机制图

ExpressPass 方案在以下方面仍存在不足. 在未授权的第 1 个 RTT 时间窗口内，不允许发送数据包会导致带宽的浪费. 其次，一个信用包代表发送方允许发送一个数据包，当流是大象流时，需要非常多的信用包占用网络带宽.

2.3 小 结

我们对目前典型的主动式拥塞控制方案进行了

Table 2 Comparison of Different Active Congestion Control Schemes

表 2 不同主动式拥塞控制方案对比

方法	年份	创新点	优点	缺点
SRP[48]	2012	第 1 个 RTT 可以发送推测性数据包，接收方收到请求后分配发送端发送时隙	有效避免最后 1 跳拥塞	当发送端向多个接收端申请授权时，会导致其中一个接收端的时隙浪费； 未考虑中间交换网络的拥塞.
pHost[49]	2015	目标端授权源端令牌后，源端才能发送数据，第 1 个 RTT 阶段可以发送数据包，且授予高优先级	实现机制简单且无需定制专用的交换设备，与现有交换机兼容	pHost 只使用有限的优先级，对于大于 RTTbytes 的小消息，不能立即抢占大消息，导致小消息的排队延时增加； 1 个令牌代表发送方允许发送 1 个数据包，当流是大象流时，需要非常多的授权包占用网络带宽.
ExpressPass[55]	2016	拥塞控制机制与 pHost 的“请求和授权”机制相同，不同的是交换机也会参与对信用包的速率限制，避免中间交换机的拥塞问题	充分考虑了中间交换节点的拥塞问题	未授权的第 1 个 RTT 时间窗口内，不允许发送数据包，会导致带宽的浪费； 1 个信用包代表发送方允许发送 1 个数据包，当流是大象流时，需要非常多的信用包占用网络带宽； 限制了授权包的传输路径，要求与数据包传输路径相同.
Homa[53]	2018	第 1 个 RTT 发送的数据包具有高优先级，且为避免带宽浪费，提出“过度授权信用”	充分优化小消息的流完成时间	在信用分配之前就过度发送数据，可能会不可预测地增加网络负载并打破接收方精细的信用分配策略，影响预期的性能.
Aeolus[54]	2020	降低第 1 个 RTT 发送的数据包优先级，且当队列拥塞时，随机早期丢弃此类包	Aeolus 可以有效地利用剩余带宽进行已授权数据包的传输，同时不会破坏主动拥塞控制期望性能	对于可以在 1 个 RTT 内完成的短流，优先级低，很可能受到其他流的影响而增加 FCT.

总结，并深入分析了每个方法的优缺点，如表 2 所示.

3 乱序包重排序

细粒度负载均衡机制将同一流的数据包分散到不同路径上进行传输可显著改善网络性能，然而，由于各路径的延迟不同，这可能导致数据包在到达接收端时发生乱序. 现有的通信协议（如 RDMA, TCP 等）并不支持乱序包的处理，当接收端检测到乱序包时，通常会认为网络发生了严重的拥塞导致了丢包，从而降低流发送速率[20]. 因此，为了不降低网络性能，需要高效的乱序包重排序机制.

本节将对目前已有的重排序机制：传统 TCP[56] 协议实现、推入先出队列 (push in first out, PIFO)[57] 实现重排、传统先入先出队列（first in first out, FIFO）实现重排[20]、SRNIC[58] 利用主机内存实现重排等进行深入介绍和对比.

3.1 主机侧的乱序包重排序

在主机侧，如果采用 TCP 传输协议，则可以直接借助主机内存实现乱序重排序. 传统 TCP 协议[56] 对于不按照顺序到达的数据包应该如何处理，并无明确规定. 如果接收方把不按照顺序到达的数据包一律丢弃，接收窗口的管理会比较简单，但是由于发送方会重复发送较多的数据，会降低网络资源利用率. 因此 TCP 通常将不按序到达的数据临时存放在接收窗口中，用位图来标记达/未到达的数据包序列号，等未按序到达的数据包收到后，再按序交付上层的应用进程，如图 13 所示.

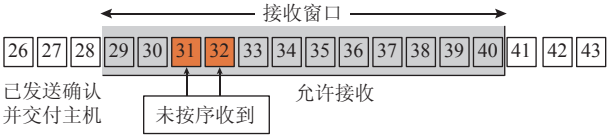


Fig. 13 An example of TCP protocol receiver bitmap

图 13 TCP 协议接收方位图示例

为避免乱序包被 TCP 可靠性重传等机制误认为丢包，可以通过设置 DupACK（累计重复确认门限值）. 如图 14 所示，假设设置 DupACK（累计重复确认门限值）为 3，那么当包 P_2 晚于 P_3 到达时，不会触发 TCP 拥塞控制或者丢包重传，因为累计确认次数不超过 3. 而当 P_5 丢失，连续触发了 3 次重复确认 P_4，则

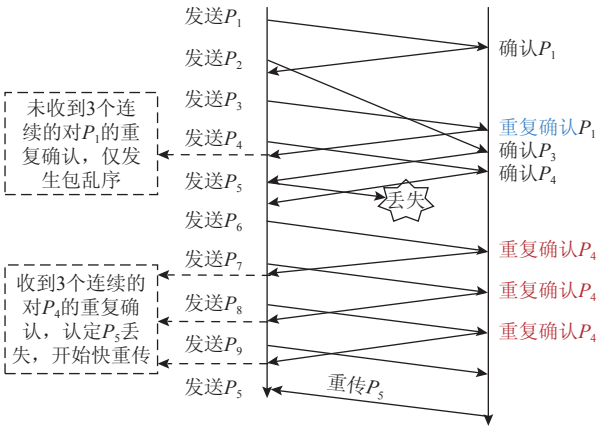


Fig. 14 Example of out of order packets and lost packets of TCP protocol detecting

图 14 TCP 协议检测乱序包和丢失包示例

认定 P_5 发生丢包，开始重传. 目前许多基于 TCP 协议进行的包喷洒或者流粒度的喷洒的负载均衡方案，如 RPS[21] 等都是通过设定（累计重复确认门限值）来支持乱序接收且避免对可靠性重传、拥塞控制产生影响.

针对 RDMA 协议，2023 年提出 SRNIC[58]，此项工作探讨了无损网络的危害，提出了有损 RDMA 网络，其中对丢失包的重排序功能是实现高性能无损网络重要的支撑技术. 为了实现重排序，需要维护 2 种数据结构. 一是位图（bitmap），用于跟踪数据包接收或丢失的状态. 每个 QP（queue pair）需要 5 个带宽-延迟乘积（BDP）大小的位图（每个位图 500 个插槽，以适应带宽 100 Gbps 和 RTT 为 40 μs 的网络的 BDPS 上限），如果是 10 000 个 QP 总共需要 3 MB 的存储空间. 二是重排序缓冲区，用于重新排列乱序数据包，并确保按顺序交付到主机内存中的数据缓冲区. 位图和重排序缓冲区所需的存储空间（仅位图就需要 3 MB）超过 RNIC 片上内存的大小（如 Mellanox RNIC 的 2 MB），因此，SRNIC 提出将所有位图放置到主机内存中，且利用主机内存中的用户数据缓冲区作为重排序缓冲区，如图 15 所示.

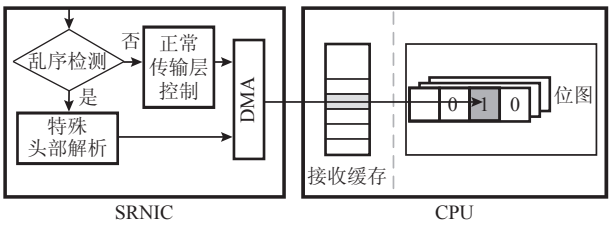


Fig. 15 Schematic diagram of SRNIC reordering

图 15 SRNIC 重排序示意图

SRNIC 将所有接收的数据包直接放置在用户缓冲区的正确地址下. 它进行了 RDMA RoCEv2（RDMA over converged ethernet version 2）头部扩展，以便 RNIC 可以通过解析其头部来派生每个数据包的地址：1）所有 SEND 语义数据包携带发送消息序列号（SSN）和数据包偏移量（PSN_OFFSET），接收端 NIC 用 SSN 和 PSN_OFFSET 来定位相应数据包的接收描述符及其接收缓冲区中的偏移量. 2）所有 WRITE 语义数据包携带它们的远程目标地址. 通过这些修改，顺序和乱序数据包都可以被直接放置到用户缓冲区的正确地址，从而实现原地重排序（in-place reordering）.

3.2 交换侧的乱序包重排序

TCP 乱序重排协议和 SRNIC 是在主机侧实现乱序重排的方案，乱序包重排序也可以在交换机侧实现端侧无感. PIFO[57] 于 2016 年被提出，旨在支持可编程数据包调度，不同的调度算法会为数据包分配相应的优先级，PIFO 根据这些优先级动态调整数据包在队列中的插入位置. 与传统的先入先出队列不同，PIFO 允许数据包依据优先级“插队”，而不是固定插入队尾. 尽管插入顺序不同，PIFO 和 FIFO 在读取时都是从队首开始，优先处理队首数据包.

尽管 PIFO 是为可编程调度器设计的，其乱序插入（根据优先级顺序插入）的原理同样适用于乱序包重排机制. 在该机制中，优先级可以映射为数据包中的{流标识，包序列号}，从而实现对乱序数据包的有效重排.

PIFO 存储格式示意图如图 16 所示. 如果一个优先级为 50 的数据包插入，那么 PIFO 的控制模块会移位已存的低优先级的数据包，图 17 是移动后的 PIFO 内容.

索引	0	1	2	3	4	5	···
优先级	237	234	37	23	22	10	···
数据	--	--	--	--	--	--	···

Fig. 16 Schematic diagram of PIFO storage

图 16 PIFO 存储示意图

索引	0	1	2	3	4	5	6	···
优先级	237	234	50	37	23	22	10	···
数据	--	--	--	--	--	--	--	···

新元素

Fig. 17 Schematic diagram of PIFO storage after inserting a new element

图 17 插入新元素后的 PIFO 存储示意图

在执行 PIFO 读操作时，与 FIFO 相同，会优先弹出队列中索引为 0 的元素，也就是最高优先级的数据. PIFO 的优势在于能够实现数据包的高速插入，但这种机制也带来了一些缺点，如增加芯片的占用面积和功耗，每次数据插入都可能需要移动大量数据.

在交换机处利用 FIFO 实现乱序重排序方案的还有 Conweave[20]，具体的重排机制如下：Conweave 跟踪下一个预期的数据包序列号以确保有序地将数据包发送至目标网卡. 例如，如果数据包预期的序列号是 1，但是交换机收到了一个序列号为 2 的数据包，就必须将其保留，直到数据包 1 到达. 图 18 中列举了重排序的具体实例. 首先分配一个队列 Q_0 专门用于转发有序数据包. 1）当数据包 4 到达时，将数据包序列号与状态下一个序列号（Next SEQ）进行比较，并发现它是无序的，则将数据包 4 保留在空队列 Q_1 中. 2）当

数据包 2 到达并且仍然是无序包的时候，将它保留在 Q_2 中. 3)数据包 1 到达并且是有序包，立即通过 Q_0 转发它. 此时，数据包 2 是下一个顺序包，将从 Q_2 中释放. Next SEQ 相应地增加到 3.4)数据包 3 到达，是顺序包，直接转发，然后刷新 Q_1 转发数据包 4，增加 Next SEQ 到 5.

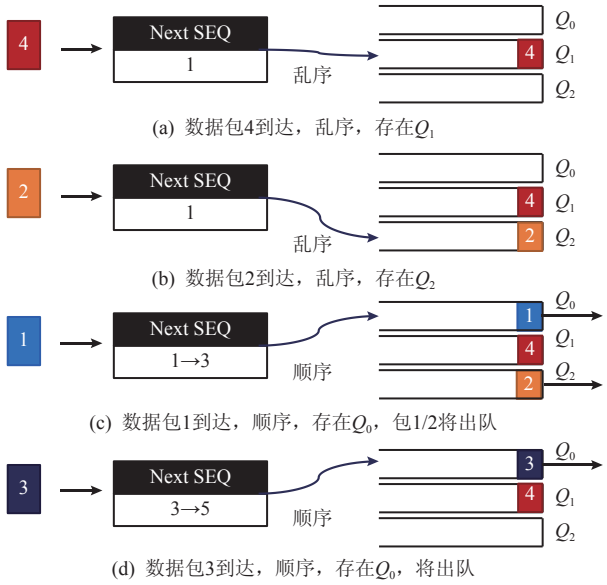


Fig. 18 Schematic diagram of Conweave reordering

图 18 Conweave 重排序示意图

Conweave 方法的优点是实现简单，效率高且延时低. 缺点是需要的队列数目多，假设有 N 个数据包都是以完全相反的顺序到达，则需要 N 个队列，然而目前交换机中的队列数目是有限的. 因此这种方法不适用于任意无序到达的情况. Conweave 最终通过设计“谨慎的”路由方案，将属于一条流的数据包最多同时在 2 个路径上传输，这样将需要的队列降低到了 2 个.

3.3 小 结

我们对目前典型的乱序包重排方案进行了总结，并深入分析了每个方法的优缺点，如表 3 所示.

4 智算系统部署案例和商业网络设备

4.1 智算系统部署案例

本节将介绍分布式 AI 训练的网络优化案例 MLT[59]，同时介绍目前 Meta[15] 和阿里巴巴建设智算中心[60] 的思路.

香港科技大学于 2024 年提出 MLT[59]，通过优化网络协议来加速分布式 DNN 的训练.DNN 训练需要频繁地传输大量的梯度，典型的 DNN 训练包括数百次迭代，每次迭代都以在数十到数百个工作节点之间传输大量梯度结束，严重的拥塞和尾部延迟会将 DNN 训练减速至“爬行”.

MLT 是一个领域专用的网络传输层协议，如图 19 所示，它利用一系列机器学习训练的特定属性来优化分布式 DNN 训练的平均 FCT 和尾部 FCT.

首先，每个数据包携带部分模型参数，参数之间具有独立性，没有先后顺序依赖，因此 MLT 利用数据包之间的独立性属性，实现了基于数据包“喷洒”的负载平衡机制，最小化网络热点的同时可忽略数据包乱序. 其次，MLT 通过实验证明了模型后面层的参数比前面层的参数更影响训练速度，而且参数的数值越大，对训练速度的影响越大. 因此 MLT 根据数据包中梯度的重要性执行优先级排队及丢弃操作. 最后，MLT 探测了不同模型的丢包边界，模型训练的参数并非需要完全无损，允许一定比例的数据包丢失而不需要重传，对训练效率的影响很小. 因此，如果网络拥塞发生时，可实施有界的数据包丢弃且不需要重传，以避免长尾延迟的影响.

MLT 使用主机控制多路径路由. 源路由通过指定每个数据包的路径以提供更可预测的性能. 设计之初考虑了每个数据包负载平衡的 2 种设计方案：

Table 3 Comparison of Different Out-of-Order Packet Rearrangement Schemes

表 3 不同乱序包重排方案对比

方法	位置	创新点	优点	缺点
TCP[56]	主机	不按序到达的数据会临时存放在接收窗口中，等到未按序到达的数据包收到后，再按序交付上层的应用进程，累计收到一定门限值数量的重复 ACK 则认为丢包	应用广泛，在网络协议栈中已成熟使用	受限于 TCP 协议栈在软件中实现，不能为应用提供极低的延时性能
SRNIC[58]	主机	将所有接收的数据包直接放置在用户缓冲区的正确地址下，使用位图跟踪哪些数据包已被接收或丢失	利用主机内存协助处理乱序包管理，可支持更高程度的乱序场景	软件参与乱序处理，增加了处理延时
PIFO[57]	交换机	可按照数据包的优先级进行插队，最终数据包按照优先级从高到低存储在队列中，按顺序弹出	硬件实现，可以线速实现插队进行排序	增加了芯片面积、功耗，每次插入操作都涉及大量数据的移动
Conweave[20]	交换机	分配一个队列 Q_0 专门用于转发有序数据包. 每当无序数据包到达时，存储在新的队列中，当成为顺序数据包时进行转发	实现简单，硬件实现，处理延时低	不适用于任意无序到达的情况. 假设有 N 个数据包都是以完全相反的顺序到达，则需要 N 个队列，在现有可编程交换机中不可实现

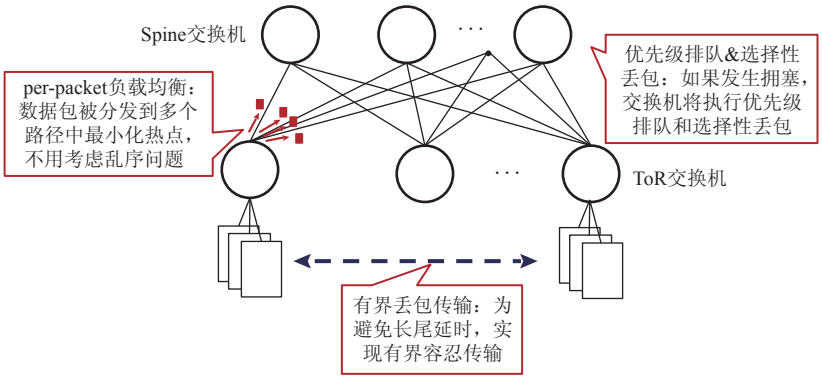


Fig. 19 MLT architecture

图 19 MLT 架构

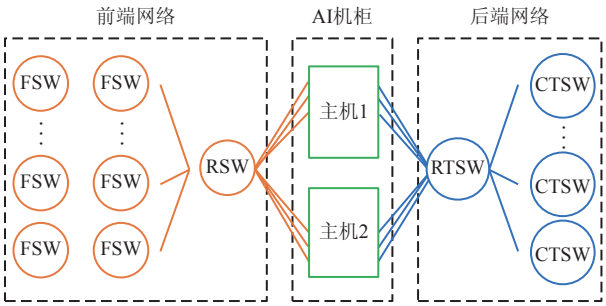
一种是利用交换机端的每个数据包 ECMP. 另一种是让主机端控制多路径路由，可以通过源路由或标签交换完成. 在最终实验时，MLT 选择了主机端实现源址路由的方案，不需要更改交换机配置，但是这种方案不能及时应对网络拥塞进行重路由. 此外，MLT 并未使用主动式拥塞控制的方法，并不能完全避免最后 1 跳的拥塞问题.

Meta[15] 于 2024 年在 SIGCOMM 发表了关于构建 AI 训练集群的经验. 为了满足分布式训练对数据中心网络基础设施的需求，一个典型的生成式 AI 任务需要数千个 GPU 在几周内进行紧密协调. 参与训练的 GPU 需要在每次迭代中多次同步梯度或更新参数，这一过程要求在毫秒级的时间内在 GPU 之间传输大量数据，并在模型收敛之前多次重复，这对网络提出了高带宽、低延迟和可预测延迟的要求. 基于对 AI 工作负载特征的深入理解，Meta 进行了专门的网络设计，涵盖了网络架构、路由、拥塞控制等方面.

在网络架构方面，Meta 为分布式训练构建了一个专用的后端网络，可以独立于数据中心网络，单独进行演进、运营和扩展. 为了支持 LLMs，后端网络扩展到了数据中心规模，如图 20 所示.

在路由算法方面，由于传统 ECMP 路由在部署之后性能不佳，Meta 同时部署了增强 ECMP(E-ECMP)方案和集中式流量工程[61-62](traffic engineer, TE)，以实现针对训练工作负载的最优负载分配.

E-ECMP 通过使用交换机用户定义功能 UDF(user defined field)实现，在路由时，交换机除了对五元组进行哈希以外，还会加上数据包头部中的目的 QP 字段，以减少哈希冲突. Meta 使用了 2 种 QP 扩展策略，第 1 种策略将在单个 QP 上的大消息拆分，改为在多个 QP 上发送，从而产生多个流量，但这导致在网络上的消息变小以及需要多个 ACK 包. 第 2 种方



RSW (rack switches): 机架交换机
FSW (fabric switches): 网络交换机
RTSW (rack training switches): 机架训练交换机
CTSW (cluster training switches): 集群训练交换机

Fig. 20 Meta network architecture

图 20 Meta 网络架构

法是以轮询的方式将不同消息发布到不同的 QP 队列中. 与基本 ECMP 算法相比，E-ECMP 在 AllReduce 集合通信上有高达 40% 的性能提升. 尽管通过 QP 扩展提高了 ECMP 的性能，但使用哈希的方法本质仍然是降低概率，不能完全消除哈希极化问题. 此外，需要根据不同的工作负载类型定制 QP 扩展号和方法，虽然短期内可行，但从长远来看，这增加了操作的复杂性.

TE 可以解决 E-ECMP 仍然存在的问题. TE 控制器基于实时工作负载和拓扑输入动态优化路由. 架构如图 21 所示. 在控制平面，收集器通过使用拓扑生成服务从内部数据仓库中的网络模型生成静态拓扑，并基于 Open/R[63](开源的路由平台，由 Facebook 公司开发，用于构建和维护其网络中的路由协议)提供的链路状态构建动态拓扑，其次，TE 引擎结合流量矩阵和训练作业调度器中的作业放置来产生流量矩阵.TE 引擎运行受限最短路径优先算法来处理实时拓扑和流量矩阵，定期(如每 30 s)生成优化的流量部署. 最后，交换机程序获取流量放置并将其转换为设

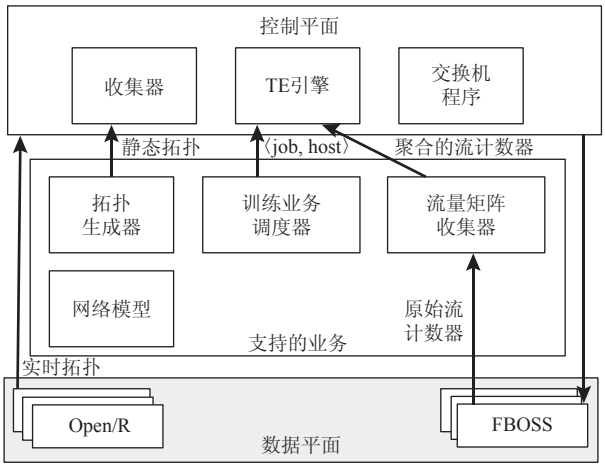


Fig. 21 TE architecture of Meta

图 21 Meta 的 TE 架构

备特定的数据平面原语以执行路由决策. 在数据平面, 按照覆盖默认路由的策略进行路由. 默认路由由 BGP[64](Border Gateway Protocol)协议提供, TE 根据优化的流量部署覆盖这些默认路由决策, 使用源和目的结合的方式提供更细粒度的流量管理.

尽管 Meta 并没有采用细粒度的负载均衡方案进行路由, 但 Gangidi 等人明确指出, 采用细粒度负载均衡性能更佳, 可以达到与 TE 相似的性能, 且代价更小. Meta 未来会考虑应用, 目前正进行更深层次的权衡和评估.

在拥塞算法方面, Meta 通过集合通信库来实现接收方的流量分配. 发送方只有在接收到包含大小和内存信息的 CTS(clear-to-send)数据包后, 才能发送 RDMA 写请求. 本质上讲, 这个方法就是采用了主动式拥塞控制, 与 pHost 和 Homa 等原理相同, 区别在于 Meta 在通信库实现, 而 pHost 和 Homa 在通信协议栈传输层实现.

阿里巴巴公司于 2024 年提出了自己的高性能网络(high performance network, HPN)构建方案[60], 如图 22 所示. 阿里巴巴公司指出传统的数据中心网络存在

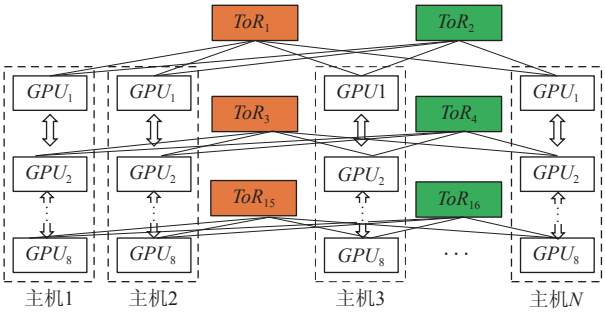


Fig. 22 Double ToR architecture of Alibaba HPN

图 22 阿里巴巴 HPN 的双 ToR 架构

负载不均衡的问题, 并不适合 LLMs 训练, 且 LLMs 训练对单点故障敏感(尤其是 ToR 上的故障). 为了解决这些问题, HPN 提出了一种新的双 ToR 设计, 以替代传统数据中心网络中的单 ToR 设计.

在应对负载均衡问题上, HPN 从 3 个方面减少传统 ECMP 方法的哈希极化问题. 1)HPN 的网络拓扑在一个段(segment)内可互连高达 1 024 个 GPU, 可显著减少了穿过聚合层的流量, 降低了网络中负载不平衡的程度. 如图 22 所示, 在实践中采用优化的网络拓扑以及双 ToR 设计. 单个主机的 3.2 Tbps(每台主机配有 8 个网卡, 每个网卡有 2 个 200 Gbps 端口, 端口 0 和 1)跨多达 16 个 ToR 交换机. 3.2 Tbps 的主机由 2 个 ToR 交换机服务, 每组双 ToR 交换机可以服务 128 个 GPU, 16 个 ToR 交换机共同连接了 1 024 (128×16/2)个 GPU 在 1 个段中. 2)如图 23(b)所示, HPN 在汇聚层实现了双平面转发. 从主机 1 的 NIC 的端口 0 发送的流量通过网络转发, 最终仅由主机 2 的 NIC 的端口 0 接收, 与主机 1 的 NIC 的端口 1 的流量物理隔离. 这种双平面设计避免了在汇聚层(包含 Agg_1 和 Agg_2 交换机)的哈希极化问题, 同时不影响超额订购比为 1∶1 的网络对分带宽, 双平面设计为跨段流量贡献了高达 71.6% 的性能提升. 3)在核心层继续采用双平面设计.

主机侧应对负载均衡问题提出了 2 个核心方案. 1)在集合通信库中高效地获得精确的不相交等价路径, 并在所有路径上平衡负载. 2)在应用层实现负载平衡, 充分利用所有 RDMA 连接, HPN 维护一个计数器, 记录当前活动工作队列描述符(WQE)中的总字节数, 集体通信库将会选择计数器最小的 RDMA 连接发送消息.

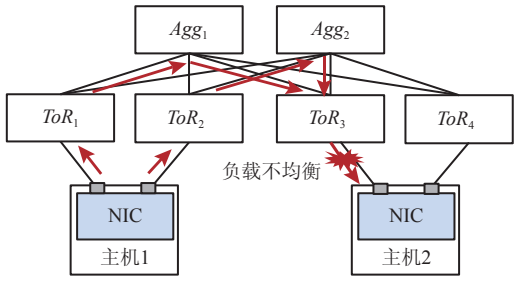
HPN 并未使用交换机侧细粒度的负载均衡方案, 该文作者指出目前的包喷洒的方案尚未经过大规模数据中心中部署验证, 也未使用目前已有的端侧的细粒度负载均衡机制, 因为这些方案都需要修改传输层, 不适用于在硬件实现传输层的 RDMA 网卡.

我们对 AI 应用部署和实际智算中心建设案例中对负载均衡、拥塞控制、乱序处理技术支持的情况进行了总结, 如表 4 所示.

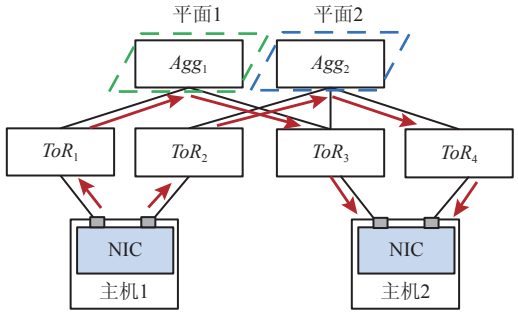
4.2 商业网络设备

本节将详细介绍主流网络厂商在细粒度负载均衡、主动式拥塞控制以及乱序重排方面的技术支持与发展现状.

Cisco 借助 Cisco Silicon One[65] 技术, 能够部署基于标准以太网协议的大规模网络交换设备. 在对性



(a) 典型clos拓扑，会造成负载不均衡



(b) 双平面拓扑，流量隔离，避免负载不均衡

Fig. 23 HPN double plane architecture

图 23 HPN 双平面架构

Table 4 Summary of Current AI Deployment and Intelligent Computing Center Construction Cases

表 4 目前 AI 部署和智算中心建设案例总结

名字	机构	负载均衡	拥塞控制	乱序处理
MLT[59]	HKUST	包喷洒	被动式拥塞控制	应用支持乱序
Meta[15]	Meta	E-ECMP 和集中式 TE 管控，正在探索应用细粒度负载均衡	通信库实现主动式拥塞控制	不乱序
HPN[60]	Alibaba	使用双 ToR 隔离流量，且在汇聚层和核心层构建双平面进行流量隔离	未提及	未提及

能有更高要求的场景下，可以采用全调度(fully-scheduled)网络[65]，Cisco P100 或 Q200L[66] 设备作为顶部机架(ToR)，并利用 Q200L 和 G100 设备构建叶脊(leaf-spine)交换节点. 全调度网络支持在整个网络中均匀“喷洒”数据包，并在出口处对乱序数据包重新排序，有效避免了拥塞的发生.

Broadcom 产品中，早期的 Tomahawk 和 Trident 交换机支持通过动态负载均衡(dynamic load balance, DLB)特性实现自适应路由. DLB 是一种基于本地交换机端口质量的质量感知负载分配方案，它根据本地交换机的端口质量为数据包选择下一跳. 它支持 per-packet“喷洒”和流模式操作，并可以为不同流量类型选择性启用，可以回退到基于传统哈希的 ECMP 方法. DLB 目前已在多个网络中成功部署.

Broadcom 的 Tomahawk 5 交换机[67] 支持全局负载均衡(GLB)，通过使用下游交换机感知的路径质量调节本地路径选择来扩展 DLB. GLB 允许上游交换机避免下游拥塞热点并选择更好的端到端路径. GLB 保留了 DLB 的优势，如当链路断开时受影响流量自动快速故障转移，并且支持不相等代价路径路由.

Broadcom 的 Jericho 和 Qumran/ramon 系列[68] 交换机搭配组建网络实现比数据包更细粒度的负载均衡. Jericho 系列交换机应用为 ToR 交换机，主要实现以“cell”为粒度的动态路由机制和 cell 的乱序重排，cell 是比数据包更小的固定大小的单元，是由 Broadcom 自定义的私有格式，ToR 交换机的入口和出口均兼容以太网协议，实现主机侧透明. Qumran/ramon 系列交换机应用为 Spine 交换机. ToR 交换机负责与终端设备进行连接，而 Spine 交换机则负责连接不同的 ToR 交换机，实现整个网络的互连.

NVIDIA Spectrum-X™网络平台[69-70] 专为 AI应用设计，由 NVIDIA Spectrum-4[71] 以太网交换机和 NVIDIA BlueField-3[72] 超级网络接口卡(SuperNIC)组成. Spectrum-X 平台支持动态路由技术和细粒度负载均衡方案，能够动态优化 RDMA 数据的路径，有效防止网络拥堵，确保最优的负载分配，并提升数据传输效率. Spectrum-4 交换机负责在每个数据包的基础上选择最不拥挤的端口，实现数据传输的均匀喷发，即便同一数据流的不同数据包通过网络的不同路径传输并可能出现无序到达，BlueField-3 DPU 也能在 RoCE 传输层有效处理无序数据包，为应用程序提供有序的数据流，并对应用层透明.

我们对目前商业设备对负载均衡、主动式拥塞控制、乱序处理技术支持情况进行了汇总，如表 5 所示.

5 总结和展望

随着智算中心网络规模和复杂度的不断提升，如何在负载均衡、拥塞控制以及乱序处理等关键技术上取得突破，已成为提升网络性能的核心挑战. 当前的技术方案虽已取得一定进展，但在面对实际部署时仍存在许多亟待解决的问题和优化空间. 接下来，我们围绕目前负载均衡、拥塞控制、乱序处理机制以及系统设计，探讨这些领域未来的发展方向与研究重点.

1)负载均衡机制. 现有机制不能同时统筹本地交换机的负载信息与全局负载信息，仅采用本地交换机信息进行重路由，会造成剩余几跳交换机的拥塞问题，而仅采用全局信息进行重路由，需要 1 个

Table 5 Summary of Current Technical Support for Commercial Devices

表 5 目前商业设备技术支持汇总

设备名称		厂商	设备类型	per-packet 路由	主动式拥塞控制	排序
Silicon One[65]		Cisco	交换机	支持	不支持	支持
Tomahawk 5[67]		Broadcom	交换机	支持	不支持	不支持
Jericho 和 Qumran/ramon[68]		Broadcom	交换机	固定单元大小	支持	支持
Spectrum-X™ Ethernet networking platform[69]	Spectrum-4[71]	NVIDIA	交换机	支持	不支持	不支持
	BlueField-3 SuperNIC[72]	NVIDIA	网卡		不支持	支持

RTT 才能更新网络状态, 不能及时应对网络突发流量导致的瞬时拥塞, 因此综合局部和全局的负载信息进行负载均衡是未来的一个重点研究方向. 此外, 现有进行全局负载信息探测的机制会带来较大开销, 如 CONGA 和 Conweave 重新定义了一套探测协议, 如何与现有交换机支持的拥塞探测机制如 ECN, INT 结合来减少探测开销, 是未来负载均衡的另一个重点研究方向.

2)拥塞控制机制. 现有主动式拥塞控制机制复杂、实现难度大, 很难实际部署, 受 Meta 的智算中心建设启发, 或许将此类拥塞控制的实现迁移至具体应用的调度机制中实现将更灵活, 并更容易在实际生产中使用. 此外, 如何同时控制中间交换节点和最后 1 跳节点的拥塞, 是主动式拥塞控制的重大挑战, 虽然 ExpressPass 考虑到这类问题, 但是其方法有很大的局限性, 要求授权包与数据包的路径相同, 这个前提条件在应用细粒度负载均衡的技术背景下不能满足, 因为 1 条流的数据包可能会在多个路径上传输, 随时会根据当下的网络状态进行重路由. 最后结合 AI 训练负载中的低熵和可预测特性, 利用集中式拥塞控制进行全局管控实现零拥塞, 是另一个重要研究方向.

3)乱序处理机制. 硬件实现乱序处理可以显著提升性能, 但是因为片上缓存有限, 采用 Conweave 的方法不能支持过高程度的乱序处理, 沿用 PIFO 的方法是可行的方案, 只用 1 个队列即可实现重排队, 但是重点需要考虑潜在的队头阻塞、包丢失检测机制等问题. 此外, 之所以要实现乱序重排机制的最根本原因在于现有通信协议 TCP, RDMA 等提供的语义不支持向应用提交乱序数据流, 默认应用层需要顺序数据流, 但是在特殊情况, 如 AI 训练参数同步阶段, 每个数据包携带不同层或同层不同位置的参数, 参数之间具有独立性, 乱序包对应用训练并不产生影响. 在这种情况下, 通信协议原则上可以将乱序数据包提交给应用, 因此, 对现有通信语义进行扩展, 也是未来的研究趋势.

4)系统设计. 现有智算中心的设计尚未实际采用“包喷洒”负载均衡机制建设, 而是倾向于使用基于流的调度策略或通过拓扑隔离来规避流量冲突. 尽管如此, 这并不意味着产业界否定了包喷洒机制的潜在优势. 目前, 产业界的部署策略相对谨慎, 但学术界已经在小规模环境中成功实施了这一方案. 预计随着时间的推移, 产业界将逐步采纳并部署包喷洒技术.

综合以上研究, 未来网络不仅需要进一步提升性能, 还需兼顾实际部署的复杂性. 在这些关键技术的推动下, 智算中心网络的设计将会迎来更多创新与突破, 为下一代高效、灵活的网络架构奠定坚实基础.

作者贡献声明: 许晶提出主要思路, 并完成论文的撰写; 王展、元国军、马振龙对论文进行修改; 杨帆完成部分资料的收集; 孙凝晖对论文整体思路进行指导并进行修改.

参 考 文 献

[1] Cao Yihan, Li Siyu, Liu Yixin, et al. A comprehensive survey of AI generated content (AIGC): A history of generative AI from GAN to ChatGPT[J]. arXiv preprint, arXiv: 2303.04226, 2023

[2] Ge Yingqiang, Hua Wenyue, Mei Kai, et al. OPENAGI: When LLM meets domain experts[J]. Advances in Neural Information Processing Systems, 2024, 36: 5539−5568

[3] Duarte F. Number of ChatGPT users (Aug 2024) [EB/OL]. [2024-10-09]. https://explodingtopics.com/blog/chatgpt-users

[4] Wikipedia Contributors. GPT-3[EB/OL]. [2020-05-28]. https://en.wikipedia.org/wiki/GPT-3

[5] Wikipedia Contributors. GPT-4[EB/OL]. [2023-03-14]. https://en.wikipedia.org/wiki/GPT-4

[6] Wikipedia Contributors. Neural scaling law[EB/OL]. [2024-10-09]. https://en.wikipedia.org/wiki/Neural_scaling_law

[7] Owens J D, Houston M, Luebke D, et al. GPU computing[J]. Proceedings of the IEEE, 2008, 96(5): 879−899

[8] Jang J W, Lee S, Kim D, et al. Sparsity-aware and re-configurable NPU architecture for Samsung flagship mobile SoC[C]//Proc of the 48th Annual Int Symp on Computer Architecture. Piscataway, NJ: IEEE, 2021: 15−28

[9] Wang Y E, Wei G Y, Brooks D. Benchmarking TPU, GPU, and CPU platforms for deep learning[J]. arXiv preprint, arXiv: 1907.10701, 2019

[10] Tang Zhenheng, Shi Shaohua, Wang Wei, et al. Communication-efficient distributed deep learning: A comprehensive survey[J]. arXiv preprint, arXiv: 2003.06307, 2020

[11] The Research Institution of China Mobile. White paper on the evolution of intelligent computing center networks for AI large models[EB/OL]. 2023 [2024-10-09]. https://13115299.s21i.faiusr.com/61/1/ABUIABA9GAAgpc_LowYotKrezQU.pdf (in Chinese)
(中国移动通信研究院. 面向 AI 大模型的智算中心网络演进白皮书[EB/OL]. 2023 [2024-10-09]. https://13115299.s21i.faiusr.com/61/1/ABUIABA9GAAgpc_LowYotKrezQU.pdf)

[12] Dukkipati N, McKeown N. Why flow-completion time is the right metric for congestion control[J]. ACM SIGCOMM Computer Communication Review, 2006, 36(1): 59−62

[13] Kashem M A, Ganapathy V, Jasmon G B. Network reconfiguration for load balancing in distribution networks[J]. IEEE Proc-Generation, Transmission and Distribution, 1999, 146(6): 563−567

[14] Zhang Yiran, Wang Shangguang, Ren Fengyuan. Survey on traffic management in lossless networks[J]. Journal of Computer Research and Development, 2025, 62(5): 1290−1306 (in Chinese)
(张乙然, 王尚广, 任丰原. 无丢失网络流量管理综述[J]. 计算机研究与发展, 2025, 62(5): 1290−1306)

[15] Gangidi A, Miao Rui, Zheng Shengbao, et al. RDMA over Ethernet for distributed training at Meta scale[C]//Proc of the 38th ACM Conf on SIGCOMM. New York: ACM, 2024: 57−70

[16] Hopps C. Analysis of an Equal Cost Multi Path Algorithm[S/OL]. [2024-10-09]. https://datatracker.ietf.org/doc/html/rfc2992

[17] Al-Fares M, Radhakrishnan S, Raghavan B, et al. Hedera: Dynamic flow scheduling for data center networks[C]//Proc of the 7th USENIX Conf on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2010: 89−92

[18] Ultra Ethernet Consortium. Ultra Ethernet Consortium[EB/OL]. [2024-10-09]. https://ultraethernet.org

[19] ETH+ Consortium Official Website. ETH+ Consortium[EB/OL]. [2024-10-09]. http://www.eth-plus.com/homePage (in Chinese)
(高通量以太网联盟官网. 高通量以太网联盟[EB/OL]. [2024-10-09]. http://www.eth-plus.com/homePage)

[20] Song C H, Khooi X Z, Joshi R, et al. Network load balancing with in-network reordering support for RDMA[C]// Proc of the 37th ACM Conf on SIGCOMM. New York: ACM, 2023: 816−831

[21] Dixit A, Prakash P, Hu C, et al. On the impact of packet spraying in data center networks[C]//Proc of the 2013 IEEE INFOCOM. Piscataway, NJ: IEEE, 2013: 2130−2138

[22] Ghorbani S, Yang Zibin, Godfrey P B, et al. DRILL: Micro load balancing for low-latency data center networks[C]// Proc of the 31st ACM Conf on SIGCOMM. New York: ACM, 2017: 225−238

[23] Huang Cheng, Simitci H, Xu Yikang, et al. Erasure coding in windows azure storage[C]//Proc of the 2012 USENIX Conf on Annual Technical Conf. Berkeley, CA: USENIX Association, 2012: 15−26

[24] Bao Yixin, Peng Yanghua, Chen Yangrui, et al. Preemptive all-reduce scheduling for expediting distributed DNN training[C]//Proc of the 2020 IEEE Conf on Computer Communications. Piscataway, NJ: IEEE, 2020: 626−635

[25] Vanini E, Pan Rong, Alizadeh M, et al. Let it flow: Resilient asymmetric load balancing with flowlet switching[C]//Proc of the 14th USENIX Conf on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2017: 407−420

[26] Zats D, Das T, Mohan P, et al. DeTail: Reducing the flow completion time tail in datacenter networks[C]// Proc of the 26th ACM Conf on SIGCOMM. New York: ACM, 2012: 139−150

[27] Zhang Yiran, Liu Yifan, Meng Qingkai, et al. Congestion detection in lossless networks[C]// Proc of the 35th ACM Conf on SIGCOMM. New York: ACM, 2021: 370−383

[28] Cisco. Per packet load balancing[EB/OL]. [2024-10-09]. http://www.cisco.com/en/US/docs/ios/12 0s/feature/guide/pplb.html

[29] Adams R. Active queue management: A survey[J]. IEEE Communications Surveys & Tutorials, 2012, 15(3): 1425−1476

[30] Lin Dong, Morris R. Dynamics of random early detection[C]// Proc of the 11th ACM Conf on SIGCOMM. New York: ACM, 1997: 127−137

[31] Alizadeh M, Edsall T, Dharmapurikar S, et al. CONGA: Distributed congestion-aware load balancing for datacenters[C]// Proc of the 28th ACM Conf on SIGCOMM. New York: ACM, 2014: 503−514

[32] Kandula S, Katabi D, Sinha S, et al. Dynamic load balancing without packet reordering[J]. ACM SIGCOMM Computer Communication Review, 2007, 37(2): 51−62

[33] Wikipedia Contributors. Virtual extensible LAN[EB/OL]. [2024-10-09]. https://en.wikipedia.org/wiki/Virtual_Extensible_LAN

[34] Mitzenmacher M. The power of two choices in randomized load balancing[J]. IEEE Transactions on Parallel and Distributed Systems, 2001, 12(10): 1094−1104

[35] NVIDIA. Mellanox Connect X-6[EB/OL]. 2023 [2024-10-09]. https://www.nvidia.com/en-sg/networking/ethernet/connectx-6

[36] Raiciu C, Barre S, Pluntke C, et al. Improving datacenter performance and robustness with multipath TCP[J]. ACM SIGCOMM Computer Communication Review, 2011, 41(4): 266−277

[37] Zhang Hong, Zhang Junxue, Bai Wei, et al. Resilient datacenter load balancing in the wild[C]// Proc of the 31st ACM Conf on SIGCOMM. New York: ACM, 2017: 253−266

[38] Guo Chuanxiong, Yuan Lihua, Xiang Dong, et al. Pingmesh: A large-scale system for data center network latency measurement and analysis[C]//Proc of the 29th ACM Conf on Special Interest Group on Data Communication. New York: ACM, 2015: 139−152

[39] Floyd S. TCP and explicit congestion notification[J]. ACM SIGCOMM Computer Communication Review, 1994, 24(5): 8−23

[40] Kompella R R, Yates J, Greenberg A, et al. Detection and localization

of network black holes[C]//Proc of the 26th IEEE Int Conf on Computer Communications. Los Alamitos, CA: IEEE Computer Society, 2007: 2180−2188

[41] Pfister G F. An introduction to the infiniband architecture[J]. High Performance Mass Dtorage and Parallel I/O, 2001, 42(10): 617−632

[42] Lu Yuanwei, Chen Guo, Li Bojie, et al. Multi-Path transport for RDMA in datacenters[C]//Proc of the 15th USENIX Conf on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2018: 357−371

[43] Alizadeh M, Greenberg A, Maltz D A, et al. Data center TCP (DCTCP)[C]// Proc of the 24th ACM Conf on SIGCOMM. New York: ACM, 2010: 63−74

[44] Zhu Yibo, Eran H, Firestone D, et al. Congestion control for large-scale RDMA deployments[J]. ACM SIGCOMM Computer Communication Review, 2015, 45(4): 523−536

[45] Li Yuliang, Miao Rui, Liu H H, et al. HPCC: High precision congestion control[C]// Proc of the 33rd ACM Conf on SIGCOMM. New York: ACM, 2019: 44−58

[46] Mittal R, Lam V T, Dukkipati N, et al. TIMELY: RTT-based congestion control for the datacenter[J]. ACM SIGCOMM Computer Communication Review, 2015, 45(4): 537−550

[47] Kumar G, Dukkipati N, Jang K, et al. Swift: Delay is simple and effective for congestion control in the datacenter[C]// Proc of the 34th ACM Conf on SIGCOMM. New York: ACM, 2020: 514−528

[48] Jiang Nan, Becker D U, Michelogiannakis G, et al. Network congestion avoidance through speculative reservation[C]//Proc of the 18th IEEE Int Symp on High-Performance Comp Architecture. Piscataway, NJ: IEEE, 2012: 1−12

[49] Gao P X, Narayan A, Kumar G, et al. pHost: Distributed near-optimal datacenter transport over commodity network fabric[C]//Proc of the 11th ACM Conf on Emerging Networking Experiments and Technologies. New York: ACM, 2015: 1−12

[50] Shao C, Deng H, Pazhyannur R, et al. RFC 7494 IEEE 802.11 Medium Access Control (MAC) Profile for Control and Provisioning of Wireless Access Points (CAPWAP)[S/OL]. [2024-10-09]. https://datatracker.ietf.org/doc/html/rfc7494

[51] Yaar A, Perrig A, Song D. SIFF: A stateless Internet flow filter to mitigate DDoS flooding attacks[C]//Proc of the 25th IEEE Symp on Security and Privacy. Piscataway, NJ: IEEE, 2004: 130−143

[52] Alizadeh M, Yang Shuang, Sharif M, et al. pFabric: Minimal near-optimal datacenter transport[J]. ACM SIGCOMM Computer Communication Review, 2013, 43(4): 435−446

[53] Montazeri B, Li Yilong, Alizadeh M, et al. Homa: A receiver-driven low-latency transport protocol using network priorities[C]// Proc of the 32nd ACM Conf on SIGCOMM. New York: ACM, 2018: 221−235

[54] Hu Shuihai, Bai Wei, Zeng Gaoxiong, et al. Aeolus: A building block for proactive transport in datacenters[C]// Proc of the 34th ACM Conf on SIGCOMM. New York: ACM, 2020: 422−434

[55] Cho I, Han Dongsu, Jang K. ExpressPass: End-to-end credit-based congestion control for datacenters[J]. arXiv preprint, arXiv: 1610.04688, 2016

[56] Iyengar J, Ros D, Arcia A, et al. RFC 5690 Adding Acknowledgement Congestion Control to TCP[S/OL]. [2024-10-09]. https://datatracker.ietf.org/doc/rfc5690

[57] Sivaraman A, Subramanian S, Alizadeh M, et al. Programmable packet scheduling at line rate[C]// Proc of the 30th ACM Conf on SIGCOMM. New York: ACM, 2016: 44−57

[58] Wang Zilong, Luo Layong, Ning Qingsong, et al. SRNIC: A scalable architecture for RDM NICs[C]//Proc of the 20th USENIX Symp on Networked Systems Design and Implementation (NSDI 23). Berkeley, CA: USENIX Association, 2023: 1−14

[59] Wang Hao, Tian Han, Chen Jingrong, et al. Towards domain-specific network transport for distributed DNN training[C]//Proc of the 21st USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2024: 1421−1443

[60] Qian Kun, Xi Yongqing, Cao Jiamin, et al. Alibaba HPN: A data center network for large language model training[C]// Proc of the 38th ACM Conf on SIGCOMM. New York: ACM, 2024: 691−706

[61] Denis M, Yao Yuanjun, Hatch A, et al. EBB: Reliable and evolvable express backbone network in Meta[C]// Proc of the 37th ACM Conf on SIGCOMM. New York: ACM, 2023: 346−359

[62] Schlinker B, Kim H, Cui T, et al. Engineering egress with edge fabric: Steering oceans of content to the world[C]// Proc of the 31st ACM Conf on SIGCOMM. New York: ACM, 2017: 418−431

[63] Facebook. Facebook/OpenR: Distributed platform for building autonomic network functions[CP/OL]. [2024-10-09]. https://github.com/facebook/openr

[64] Feamster N, Winick J, Rexford J. A model of BGP routing for network engineering[J]. ACM SIGMETRICS Performance Evaluation Review, 2004, 32(1): 331−342

[65] Cisco. Cisco Silicon one product family white paper[EB/OL]. [2024-10-09]. https://www.cisco.com/c/en/us/solutions/collateral/silicon-one/silicon-one-wp.html#Machinelearningandartificialintelligence

[66] Cisco. Cisco Silicon one Q200 and Q200L processors data sheet [EB/OL]. 2022 [2024-10-09]. https://www.cisco.com/c/en/us/solutions/collateral/silicon-one/datasheet-c78-744312.html

[67] Broadcom. Cognitive routing in the Tomahawk 5 data center switch[EB/OL]. [2024-10-09]. https://www.broadcom.com/blog/cognitive-routing-in-the-tomahawk-5-data-center-switch

[68] Zilberman N, Bracha G, Schzukin G. Stardust: Divide and conquer in the data center network[C]//Proc of the 16th USENIX Conf on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2019: 141−160

[69] NVIDIA. NVIDIA Spectrum-X networking platform[EB/OL]. 2022 [2024-10-09]. https://www.nvidia.com/en-us/networking/spectrumx

[70] NVIDIA. NVIDIA Spectrum-X Ethernet: Accelerated networking for AI workloads NADDOD blog[EB/OL]. 2022 [2024-10-09]. https://www.naddod.com/blog/nvidia-spectrum-x-an-ethernet-network-platform-specifically-designed-for-ai

[71] NVIDIA. NVIDIA Spectrum−4 datasheet[EB/OL]. 2022 [2024-10-09]. https://resources.nvidia.com/en-us-accelerated-networking-resource-library/ethernet-switches-pr?xs=425229#page=1

[72] NVIDIA. NVIDIA BlueField networking platform[EB/OL]. 2022 [2024-10-09]. https://www.nvidia.com/en-us/networking/products/data-processing-unit



Xu Jing, born in 1992. PhD, engineer. Member of CCF. Her main research interests include high performance interconnection network and ASIC design.

许　晶, 1992 年生. 博士. 工程师. CCF 会员. 主要研究方向为高性能互连网络、交换芯片设计.



Wang Zhan, born in 1986. PhD, professor, PhD supervisor. Member of CCF. His main research interests include high performance interconnection network and distributed system architecture.

王　展, 1986 年生. 博士, 研究员, 博士生导师. CCF 会员. 主要研究方向为高性能互联网络、分布式系统架构.



Yuan Guojun, born in 1983. PhD, associate professor, master supervisor. Member of CCF. His main research interests include optical networks and system interconnection.

元国军, 1983 年生. 博士, 副研究员, 硕士生导师. CCF 会员. 主要研究方向为光网络、系统互联. (yuanguojun@ncic.ac.cn)



Ma Zhenlong, born in 1996. PhD. His main research interests include high performance interconnection and RDMA.

马振龙, 1996 年生. 博士. 主要研究方向为高性能互连网络、RDMA. (mazhenlong@ncic.ac.cn)



Yang Fan, born in 1992. PhD, engineer. Member of CCF. His main research interests include high performance interconnection network, in-network computing, and high-performance storage system.

杨　帆, 1992 年生. 博士, 工程师. CCF 会员. 主要研究方向为高性能互联网络、在网计算、高性能存储系统. (yangfan@ncic.ac.cn)



Sun Ninghui, born in 1968. PhD, professor, PhD supervisor. Member of CCF. His research interests include computer architecture, operating systems, and parallel algorithms.

孙凝晖, 1968 年生. 博士, 研究员, 博士生导师. CCF 会员. 主要研究方向为计算机体系结构、操作系统、并行算法. (snh@ict.ac.cn)