数据中心可编程数据平面的调度机制研究综述

刘敬玲,何 种,汪子汐,方 浩,崔 睿,黄家玮*
(中南大学计算机学院,湖南长沙 410083)

摘 要: 数据中心承载着云计算、大数据分析与人工智能等业务,不同流量对时延和吞吐量的需求各异. 时延敏感的任务要求极低延迟,而大规模备份和分析流量则更关注高吞吐量,为此数据中心网络通常采用高效的调度机制以满足不同应用需求. 传统调度机制依赖定制网络设备实现来满足特定的应用需求,其开发成本高且可扩展性差,导致难以在网络中快速部署新的调度算法. 随着可编程交换机等新型网络设备的出现,数据平面可编程能力得到显著提升,网络设备从功能固化转向灵活配置,这为在数据平面设计高性能的调度机制奠定了基础. 近期研究者们提出了大量新颖的可编程调度机制,这些机制极大地提升了网络性能. 本文综述了近年数据中心可编程调度机制的设计思想,并将它们划分为通用表达、高并发流量、公平保障、租户隔离4类机制. 其中,通用表达型机制提供灵活抽象支持多种策略;高并发流量型机制强调海量并发流下的线速处理;公平保障型机制关注在流或租户间实现带宽公平;租户隔离型机制面向多租户环境提供强隔离和分层资源分配. 4类调度机制各有侧重,可互补应对复杂场景需求. 随后,本文分别介绍相应代表性调度机制,并进一步比较了各类调度机制的优缺点,最后展望了基于可编程数据平面的调度机制的未来发展方向.

关键词: 数据中心网络;可编程数据平面;优先级调度;公平性;多租户;时延

基金项目: 国家自然科学基金(No.62302524,No.U24A20245,No.62132022);湖南省自然科学基金(No.2024JJ6531);贵州省科技支撑计划(No.[2025]022);湖南省科技创新计划(No.2024RC1005)

中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112(2025)12-4740-16

电子学报 URL:http://www.ejournal.org.cn **DOI:**10.12263/DZXB.20250334

A Survey of Scheduling Mechanisms for Programmable Data Plane in Data Center

LIU Jing-ling, HE Zhong, WANG Zi-xi, FANG Hao, CUI Rui, HUANG Jia-wei*
(*School of Computer Science and Engineering*, *Central South University*, *Changsha*, *Hunan* 410083, *China*)

Abstract: Data centers host cloud computing, big data analytics, and artificial intelligence workloads, whose traffic exhibits highly heterogeneous latency and throughput requirements. Latency-sensitive tasks require extremely low delay, whereas large-scale backup and analytics flows are more concerned with high throughput; accordingly, data center networks usually adopt efficient scheduling mechanisms to meet these diverse application needs. Traditional packet scheduling mechanisms are implemented on customized network devices to meet specific application requirements, which leads to high development costs and poor scalability and makes it difficult to rapidly deploy new scheduling algorithms in the network. With the emergence of new network devices such as programmable switches, the programmability of the data plane has been significantly enhanced. Network devices have shifted from fixed functionalities to flexible configurations, laying the groundwork for designing high-performance scheduling mechanisms in the data plane. Recently, researchers have proposed numerous innovative programmable scheduling mechanisms, significantly improving network performance. This paper provides a comprehensive survey of recent programmable scheduling mechanisms for data center networking and classifies them into four categories: general-purpose abstraction mechanisms, mechanisms for high-concurrency traffic, mechanisms for fairness assurance, and mechanisms for tenant-level isolation. General-purpose abstraction mechanisms provide flexible abstractions that support a wide range of scheduling policies. Mechanisms for high-concurrency traffic emphasize line-rate processing under massive flow concurrency. Mechanisms for fairness assurance focus on providing fair bandwidth alloca-

收稿日期:2025-04-28;录用日期:2025-12-03;责任编辑:覃怀银
*通讯作者:黄家玮

tion among flows or tenants. Mechanisms for tenant-level isolation target multi-tenant environments by providing strong isolation and hierarchical resource allocation. These four types of mechanisms have different design emphases but can complement each other to cope with complex and diverse data center scenarios. Then, representative scheduling mechanisms for each category are discussed, followed by a comparative analysis of the advantages and disadvantages of various categories. Finally, the paper concludes with a discussion on the future development trends of scheduling mechanisms based on programmable data planes.

Key words: data center networking; programmable data plane; priority scheduling; fairness; multi-tenant; latency

Foundation Item(s): National Natural Science Foundation of China (No.62302524, No.U24A20245, No.62132022); Natural Science Foundation of Hunan Province (No.2024JJ6531); Guizhou Provincial Science and Technology Projects (No.[2025]022); Science and Technology Innovation Program of Hunan Province (No.2024RC1005)

1　引言

1.1　数据中心流量特征与调度需求

当今数据中心承载着云计算、大数据分析、大模型训练与推理等应用的运行,不同应用产生的流量对网络需求各不相同. 例如,交互浏览流量、付费结算流量对延时敏感,而备份存储流量、大数据处理流量对吞吐量敏感. 其中,付费结算、网络游戏应用的时延敏感度较高,时延要求在10 ms以内;网页浏览、视频播放业务对时延要求在50 ms以内;数据备份存储、大数据运算处理业务对时延敏感度较低,时延要求在200 ms以内或更长[1]. 在Google的一项研究中,延迟每增加400 ms会使用户的搜索次数减少0.6%,延迟每增加100 ms使亚马逊的销售额减少1%[2]. 同时,数据中心链路带宽已从10~40 Gbps增长至400~800 Gbps. 面对持续增长的链路带宽,若调度器吞吐量与链路速率不匹配,易引发数据包堆积、传输延迟上升与丢包重传等问题,最终导致资源利用率下降并成为系统性能的瓶颈. 因此,调度器需维持充足且稳定的吞吐量,以在高速链路下支持大规模数据包调度. 例如,在100 Gbps链路下若要实现线速处理数据包,调度器需在纳秒级别的时间内处理每个数据包(如64字节的数据包处理时间需<7 ns),意味着调度器需至少维持150 Mpps的吞吐量. 高效的数据平面调度技术是满足不同应用需求的重要手段,其中数据包调度机制能够显著提升数据中心网络性能[3~9].

数据包调度机制是指通过执行某种调度算法或者调度策略来计划每个数据包何时以何种顺序在网络中传输,从而维持网络性能和保障服务质量. 传统商用交换机为支持线速处理,仅提供先进先出(First In First Out,FIFO)、赤字轮询(Deficit Round Robin,DRR)[10]、严格优先级(Strict Priority,SP)等简单调度算法. 随着网络需求的快速演变,各种新调度算法和调度策略层出不穷. 例如,基于执行最短剩余处理时间优先算法(Shortest Remaining Time First,SRTF)[11]最小化流完成时间,基于加权公平队列算法(Weighted Fair Queuing,WFQ)[12]保障流间或租户间的带宽公平,基于最小松弛时间优先算法(Least Slack Time First,LSTF)[13]最小化拖尾延时. 然而,传统网络设备的控制逻辑和转发逻辑紧密耦合,这使得新功能和新协议依赖于新的硬件设计,因其实现周期长、可扩展性差,进而阻碍了网络演进.

为弥补上述不足,OpenFlow[14]、软件定义网络(Software-Defined Networking,SDN)[15]和可重配置匹配-动作表(Reconfigurable Match-action Tables,RMT)架构[16]等框架相继被提出. 此外,P4可编程语言以及相应的转发模型[16,17]被相继提出,赋予数据平面更高的可编程性,如支持带状态负载均衡[18,19]、大流检测[20~22]和分布式计算[23,24]等高级网络功能,极大地推动了数据平面可编程技术的发展. 可编程技术通过统一编程接口管理单个网络节点上的处理器、存储和队列等资源,实现自定义的数据包处理逻辑,提升了数据平面的可编程能力和数据包处理的灵活性,降低了开发周期与开发成本,为实现新功能和新协议带来了新的解决方案[25]. 例如,可编程交换机如Barefoot的Tofino,Cavium的XPliant[26]和Intel Flexpipe[27]以及Broadcom的Trident4 switch,它们采用"匹配+动作"的处理模型,根据包头字段进行匹配,再执行相应的处理动作[28],同时提供了计数器、寄存器等有限的状态内存和加法、位移与哈希运算等计算原语以支持数据包的灵活处理. 此外,FPGA(Field-Programmable Gate Array)开发板[29,30]、DPU(Data Processing Unit)网卡[31]、DPDK(Data Plane Development Kit)网卡与智能网卡[32]等设备使可编程调度器在真实环境中的部署更加便捷,进而引发学术界与工业界对基于可编程数据平面调度机制的持续关注.

1.2　相关综述工作简述

近年来,随着可编程数据平面技术的发展,相关综述工作相继发表. 本节对其中代表性的工作作简要介绍.

文献[33]以宏观视角呈现了可编程数据平面的演进轨迹与应用趋势,首先讨论了可编程数据平面为何成为下一代网络服务与应用的关键推动力,详细描述了可编程包处理流水线、包解析与调度的抽象,并介绍了相关的编程语言与编译器. 同时,文献[33]也系统梳理了数据平面数据包处理所依赖的数据结构.

文献[34]系统调研了可编程数据平面在机器学习方向的最新进展,将现有工作划分为操作扩展、内存节省以及体系结构设计3类,并深入讨论这3类工作的优缺点.此外,文献[34]综合分析了可编程数据平面对机器学习的加速收益,通过对比机器学习的计算/存储需求与当前可编程数据平面的能力,指出二者之间显著的需求-能力鸿沟.文献[35]面向以P4语言为代表的可编程数据平面技术,综述了2017—2021年间大量基于P4的应用研究,将其分为了监控、流量与拥塞管理、路由与转发、高级网络、网络安全及其他6类工作.此外,文献[35]详细介绍了数据平面编程模型、P4编程语言以及数据平面API(Application Programming Interface)等基础内容,总结了推动P4技术演进的关键技术.

相比于已有综述从编程模型、编程语言、编程架构的角度回顾可编程数据平面技术,本文的创新点在于从网络性能需求的角度出发,围绕可编程数据平面调度机制相关的原理、目标和挑战,系统地介绍其发展过程和研究现状.

2 背景概述

2.1 可编程数据平面

在传统网络设备的架构中,控制平面和数据平面紧密耦合.控制平面负责管理网络设备的全局行为,数据平面则专注于具体的数据包处理包括解析数据包头、匹配转发表、修改数据包内容、转发数据包、过滤不符合调度策略的数据包.控制平面和数据平面间分工明确的架构提供了更高的性能和更好的稳定性.然而,传统设备的功能逻辑由设备制造商在硬件中预先固定,实现新功能或新协议只能依赖新设备的更换,存在开发周期长、替换成本高、难以大规模部署等问题[36].随着网络应用的演进,流量模式、协议标准和服务需求日趋复杂,传统设备难以适应网络环境的快速变化.可编程数据平面通过标准化的接口开放设备功能逻辑,允许用户能够自定义数据包处理流程,为网络协议和架构的创新带来了新的机遇.本节将介绍几种常用的可编程数据平面处理器平台,分别从性能、灵活性和可扩展性上分析它们的优劣势.

通用硬件常用于构建数据中心的商用服务器中,以支持大规模的数据包处理任务.CPU(Central Processing Unit)作为典型的通用硬件,具备强大的可编程能力,它能够通过运行不同的软件框架,灵活适配复杂的网络需求,以实现多种网络功能.但由于通用硬件并非专为网络处理任务设计,它在某些复杂场景下的工作效率较低.例如,数据包处理需要在网卡与CPU之间频繁传输数据,这将增加数据平面单包处理时延,降低链路带宽利用率.尽管通用硬件与专用硬件相比在性能上有所局限,但其高度的灵活性和完善的开发生态(如可编程软件交换平台[37~42]、用户态I/O库[43,44]、网络功能虚拟化平台[45~48])使得通用硬件在网络数据处理领域仍然具有独特优势.通用硬件支持多种网络功能的快速开发与部署,适用于云计算、边缘计算等需快速响应变化的场景.

专用集成电路(Application-Specific Integrated Circuits,ASIC)是专为特定任务或应用场景定制的硬件芯片,传统上不可编程,但在性能上表现出色.随着网络对数据平面可编程性需求的日益增加,业界涌现出了多种灵活的交换机芯片架构,如可重配置匹配-动作表RMT[16]、协议无关交换机架构(the Protocol-Independent Switch Architecture,PISA)等.其中,可编程ASIC被组织为一系列可编程匹配-动作阶段.在数据包进入流水线之前,程序化数据解析器负责解析数据包的各个协议头,通过不同的匹配-动作表匹配协议头对应的动作,利用算术逻辑单元(Arithmetic Logical Units,ALU)修改包头、执行简单计算或者更新内部状态等操作.此外,这些匹配-动作表RMT根据硬件的实现方式具有不同的匹配能力.在流水线末端,解析器将数据包转发到网卡出口或者发送到下一个流水线执行新的匹配动作.许多交换机[26,27]中通常至少有入口和出口两个流水线.

网络处理器(Network Processors Units,NPU)是一种专用于网络数据包处理的加速器,广泛用于数据包转发、加密、负载均衡和分类等操作.NPU具有高度的可编程性,并且不强制规定数据包处理步骤的特定顺序,能够支持循环操作,实现更多复杂的数据包处理功能.此外,NPU具有多个处理核心且优化了并行计算,可并行处理大量数据包.NPU借助分层内存架构(如SRAM、DRAM)能够在不同处理阶段快速访问数据包的头部和有效负载,从而实现复杂数据包操作如入侵检测中的深度数据包检测,有效提升数据流管理能力.然而,与RMT架构相比,NPU的分层内存访问和复杂逻辑处理可能会增加处理数据包的额外延迟,造成系统吞吐量下降.

现场可编程门阵列(Field-Programmable Gate Array,FPGA)[29,30]是基于相互连接的、可配置逻辑块的半导体设备.相比于可编程ASIC,FPGA具有更高的灵活性和可扩展性,可作为昂贵且功能固定的ASIC的高效替代方案[49~51].一方面,FPGA属于通用计算架构,可实现比RMT更完善的数据平面抽象,以支持更多的数据平面功能卸载,从而节省宝贵的CPU计算资源[52];另一方面,FPGA拥有丰富的片内计算和存储资源,并且提供了丰富的接口用于片外扩展,缓解了可编程ASIC的资源受限问题[53].在FPGA中,并行处理的数据能够执行不同的操作,避免了不必要的资源浪费,并消除了额外延

迟[54]. 由于其高效的数据并行性,FPGA能实现更高的加速比,适用于高吞吐量和低延迟的数据包处理场景.

智能网卡(Smart Network Interface Card, SmartNIC)[55],也被称为智能网络适配器,是在传统网卡(Network Interface Card,NIC)基础上发展起来的先进网络设备. 传统NIC通常只负责基础的包转发和协议处理,较为复杂的网络任务一般依赖主机CPU来完成. SmartNIC不仅继承了传统NIC的基本功能,还能够卸载一些计算密集型的网络任务,如数据包转发、流量管理等,从而有效减轻CPU的负担,显著提高数据处理效率. 与传统NIC相比,SmartNIC具备更强的灵活性,允许开发者根据实际需求自定义网络处理流程. 用户通过编程语言(如P4[56]、eBPF[43]等)可以灵活配置数据包的处理、过滤、转发等操作,用以满足不同应用场景的需求. 目前SmartNIC广泛应用于云计算、存储加速等高性能计算领域,有效地提升了网络性能和整体系统效率.

综上所述,通用处理器平台具备较高的可编程性,但在数据包处理能力上相对有限;而专用处理器平台在数据包处理时具备更高的吞吐量和更低的时延,但其可编程性受限. 随着可编程数据平面技术的日益成熟,包处理逻辑能够按需编程与调整. 学术界借助于这类平台探索和实现基于可编程数据平面的数据包调度方案,并结合网络流量特性、应用需求和系统状态进行动态优化调度策略,从而提升资源利用率和服务质量.

2.2　数据中心可编程数据平面调度面临的挑战

尽管可编程数据平面调度已成为研究热点,但从底层的处理平台到上层的应用需求仍面临一系列挑战. 本节将对这些挑战进行系统梳理与归纳.

(1)有限的硬件资源. 许多调度算法在实现上依赖复杂的数据结构和排序机制,需要执行实时计算、队列管理和动态优先级调整,这将引入较大的计算和存储资源开销. 可编程数据平面处理器平台受工艺和成本的限制,它们的计算和存储资源十分有限,难以高效执行复杂运算. 例如,可编程交换机(如Cavium XPliant[26]、Barefoot Tofino和Intel Flexpipe[27])仅支持简单的整数算数运算,而不支持对数、熵值等复杂运算类型及浮点数的计算. 此外,FPGA的硬件资源(如逻辑元素和自适应逻辑模块)在执行复杂操作时也容易迅速耗尽. 例如,Intel Stratix 10 MX 2100仅配备约2 073k个逻辑元素(Logic Elements, LE)和约700k个自适应逻辑模块(Adaptive Logic Modules,ALM),而Stratix V[29]系列的资源更为紧张,最大仅提供622k个LE和234k个ALM. 同时,现有可编程数据平面处理器平台存储资源有限,难以存储大规模流状态信息. 例如,Barefoot Tofino 2中64个100 G端口共享64 MB缓存,而Intel Flexpipe[14]的共享缓存大小仅为7.5 MB,智能网卡Cavium cn2360仅支持4 MB共享二级缓存和16 GB主存[57]. 有限的硬件资源极大地制约了开发者对可编程数据平面调度模型和方法的设计能力,无法适应超大规模数据中心的应用需求. 因此,如何简化可编程数据平面调度机制的内在机理,降低调度算法复杂度,利用有限的计算和存储资源实现高效的调度方案成为学术界和工业界研究焦点.

(2)多样的应用需求. 由于网络流量多样、负载多变以及业务复杂,可编程数据平面调度需在不同条件下满足各类应用的带宽、时延与隔离等目标. 例如,当不同租户或者任务要求公平共享同一网络资源时,需确保不同租户或不同任务之间能够获得合理的资源分配,以防止某些流量长期占用资源,导致其他流量陷入饥饿状态. 对于实时性要求高的应用,如实时通信和在线游戏,需保证流量低延时传输,防止因延时过高导致的卡顿或不同步问题. 对于吞吐量敏感的应用,如大数据处理和高性能计算,则需要在较短时间内处理大量数据,因而要求网络维持高吞吐量和高带宽利用率. 因此,面对复杂网络环境,研究超大规模数据中心的流量调度和建立新型调度范式来满足各种应用需求是重要挑战.

(3)复杂的调度算法. 数据包调度算法为不同类型的应用程序提供高效、公平且符合服务质量要求的数据传输服务. 例如,赤字轮询算法(DRR)[10]、加权公平队列算法(WFQ)[12]、最坏公平加权公平算法(Worst-case Fair Weighted Fair Queuing, WF^2Q+)[58]实现对带宽配额管理,这类工作保留调度算法要求当队列中有数据包等待时必须转发数据包,不允许链路在有业务时处于空闲状态. 漏桶算法(Leaky Bucket Filter,LBF)[59]、速率控制静态优先级算法(Rate-Controlled Static-Priority queuing, RCSP)[60]实现突发流量的平滑,允许一定程度的突发传输,这类非工作保留算法允许队列中等待的数据包到满足调度条件时才开始转发. 另外,优先级调度算法如最短任务优先(Shortest Job First,SJF)[61]、最短剩余时间优先算法(Shortest Remaining Time First,SRTF)[11]、最小松弛时间优先算法(Least Slack Time First,LSTF)[13]、最早截止期限优先算法(Earliest Deadline First,EDF)[62],分配给每个实体如数据流或数据包一个优先级,为高优先级实体预留带宽,低优先级实体在资源空闲时传输. 层级调度算法将流量分到不同的层级组中,不同组中的流实行指定的调度算法,比如共享带宽的租户之间采用公平带宽分配算法,而租户内部流量执行最短剩余时间优先算法. 因此,如何提高可编程数据平面调度机制的表达能力来实现各种各样复杂调度算法的效果也极具挑战.

本节概括了数据中心可编程数据平面调度面临的主要挑战,揭示了从底层硬件到上层应用的多重约束,明确了在可编程数据平面上实现高效调度所需解决的

关键问题. 近年来,可编程调度机制通过强化调度抽象模型提升表达能力,通过利用现有商用网络设备近似实现最优调度,通过实现公平、低时延与高吞吐等目标满足复杂的应用需求. 接下来将围绕数据中心可编程数据平面调度机制面临的关键问题,对现有数据包调度机制进行分类描述.

3 数据中心可编程数据平面调度设计思想和方法

由于当前可编程数据平面调度机制面临的核心问题聚焦于多样调度算法的表达、大规模并发流量的支持、资源的公平分配和服务质量的隔离保障,因此本节围绕数据中心可编程数据平面调度机制面临的核心问题将现有数据包调度机制划分为通用表达、高并发流量、公平保障、租户隔离4类,并详细对比分析各类方案的基本思想、实现方法、调度效果以及优劣势.

3.1 通用表达的调度机制研究

通用表达的调度机制是指用户能在网络设备中根据当下服务需求与实时网络状态,动态定义、调整和优化数据包调度的逻辑,从而实现复杂的流量调度策略. 可编程调度器为每个包分配一个秩值,来决定每个数据包的发送时机和传输顺序. 其中,秩值越小表示数据包的优先级越高,可更早从缓存中调度转发. 理想情况下,数据包调度器应严格按照数据包的秩进行排序转发. 但在实际网络场景下常出现数据包反转现象,即秩较高的包比秩较低的包优先被调度. 这将引发额外的网络延迟和丢包重传,从而降低调度效率和网络性能. 最初提出的PIFO模型[3]具备良好调度策略表达能力,但其硬件实现依赖定制芯片,难以实现快速部署. 因此,后续一系列研究通过设计新颖的数据结构,在现有的可编程交换机上近似实现了PIFO调度行为. 同时,也有部分研究尝试探索精确的PIFO实现,以满足部分网络场景对高精度调度的需求. 本节将介绍几种具有代表性的调度机制:PIFO[3]、PIEO[4]、AIFO[5]、SP-PIFO[6]、PACKS[7]和BMW-Tree[8].

PIFO[3]是首个提出可编程数据包调度的硬件架构,以适应不同应用场景和复杂流量的需求并消除了包反转现象. PIFO维护了一个优先级队列,确保队列中元素按其优先级顺序调度. 具体来说,每个数据包由预先设定的调度算法计算秩值,数据包根据其秩被插入到优先级队列中的相应位置,秩越大优先级越低,PIFO始终从队头转发数据包,以保证秩较小的数据包被优先转发. 如图1(*a*)所示,新到达的包首先会根据调度算法进行秩的计算(图中包的秩为2),随后插入到队列中合适的位置,等待转发. PIEO[4]则是PIFO的一种扩展,允许从队列中任意位置转发数据包,能支持更多灵活的调度算法. 但由于两者复杂的排序操作,它们都难以实现线速数据包处理,且需定制硬件,难以实现大规模部署.

为能在商用网络设备上部署,多种近似实现PIFO行为的方案被提出. 例如,AIFO[5]仅使用单个FIFO队列,并通过准入控制近似实现PIFO. 其核心思想是使用滑动窗口对近期到达数据包的秩做相对估计,再结合当前队列占用情况自适应设定队列的准入控制条件. 如图1(*b*)所示,秩为2的数据包达到时,满足准入控制条件,允许其进入队列等待转发. 但AIFO的准入控制机制难以缓解包反转现象,且受限于单个FIFO队列的容量,在高负载或突发流量下易导致丢包率升高,影响网络性能与流量公平性. SP-PIFO[6]则通过动态调整数据包秩与严格优先级队列之间的映射机制近似实现PIFO,每当数据包到达时,SP-PIFO从最低优先级队列开始向上扫描队列,并比较数据包的秩与队列边界,最终将其放入首个边界值不大于数据包秩值的队列. 随后,将队列边界更新为该数据包的秩,确保秩更低的数据包不会进入该队列,以减少调度错误. 在最高优先级队列中检测到包反转时,SP-PIFO通过减少所有队列边界来修正映射关系. 例如,如图1(*c*)所示,数据包到达的顺序为[4,3,1,2],秩为2的数据包从下至上扫描各队列边界,放入第一个边界值不大于其秩值的优先级队列. SP-PIFO能够在使用较少优先级队列的条件下,高度匹配PIFO调度效果,并在现有硬件设备上以线速运行,支持大规模流量调度. 但SP-PIFO无法解决调度中的饥饿问题,具体来说,低优先级队列中的数据包会长期滞留在队列中得不到转发,从而影响网络性能.

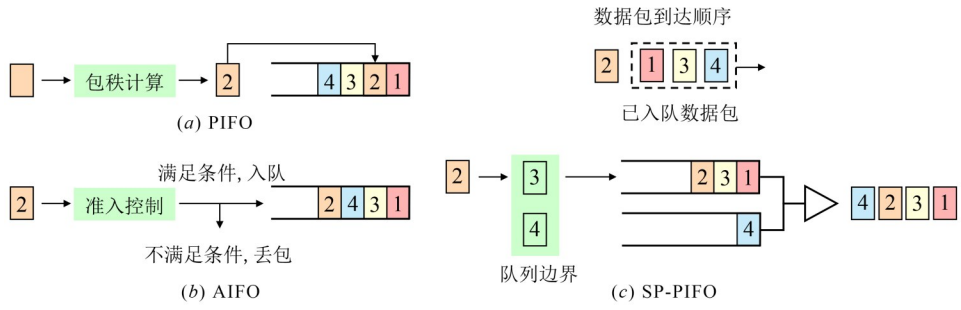

图1 数据包在不同调度器下的转发情况

Alcoz等人[7]认为,过去近似实现PIFO的方案没有全面考虑PIFO的调度特性. 例如,AIFO仅对数据包进行准入控制,导致包反转现象依然频繁;SP-PIFO虽大幅降低了包反转现象,但因不主动丢弃高秩数据包,限制了调度效率的提升. 针对此问题,他们提出了PACKS[7],该方案结合了AIFO的准入控制机制和SP-PIFO的多队列调度策略,以实现高效的数据包调度. 具体来说,当数据包到达时,PACKS首先采用类似AIFO的准入控制机制,对数据包进行初步筛选. 在此基础上,使用SP-PIFO中数据包秩与严格优先级队列映射关系,进行数据包转发. 该方案有效融合了这两者的优势,不仅减少了调度错误,而且进一步降低了流平均完成时间. 但PACKS依赖滑动窗口对数据包等级分布进行预测,并基于此做出包的准入决策,在面对突发流量时,可能因无法准确捕捉短期流量变化造成错误丢包,进而影响系统吞吐量.

尽管近似实现PIFO的方案降低了部署门槛,但仍有部分研究致力于实现具备精确调度行为的PIFO,以满足高精度调度场景的需求. BMW-Tree[8]提出了一种基于平衡多路排序树实现的PIFO,兼顾大规模流量调度和高吞吐量. 平衡多路排序树是一种满足小根堆性质的数据结构,不同的是,每个节点可以容纳多个元素,且每个元素将存储数据包的秩、其子树负载状态以及数据包的元数据. 元素入队时,从根节点递归地比较树中元素秩的大小和其负载状态,最终将选择子树负载最小的位置进行插入. 出队时,始终从根节点删除最小元素. 该结构避免了复杂的跨层交互,且每个节点的操作仅依赖父节点和子节点,从而使BMW-Tree能够稳定高效地支持流水线调度.

上述机制显著提升了网络设备的灵活性,允许用户根据实际网络需求动态调整调度策略. 然而,因可编程交换机在计算和存储资源等方面的限制,可编程调度器在实际部署中仍面临诸多挑战. 本节讨论的调度机制及其对比分析如表1所示.

表1　通用表达的调度机制研究

调度机制	工作原理	队列管理	调度精度	优势	劣势
PIFO[3]	新到达的数据包根据其秩值被插入到队列中合适的位置,并始终从队头进行转发数据包.	单队列	精确调度	能够实现按秩值严格排序的包调度,在理想条件下避免包反转.	在高速网络环境中,其硬件实现困难,且资源消耗高,可扩展性受限.
PIEO[4]	在PIFO的基础上,能够根据数据包的转发条件,将其从队列中的任意位置进行转发.	单队列	精确调度	相较于PIFO,能够支持更多、更复杂调度算法,且更具有扩展性.	在数据平面中的排序需要大量SRAM资源,导致该方案目前难以部署在现有的可编程交换机上.
AIFO[5]	仅使用单个FIFO队列,在入队时基于滑动窗口估计数据包秩的相对大小,并基于此进行准入控制.	单队列	近似调度	仅使用单个FIFO队列,资源消耗小,不需要对每个数据包进行复杂排序.	调度过程中数据包反转现象仍然严重,且因单FIFO队列容量有限,易产生过多丢包.
SP-PIFO[6]	使用多个FIFO队列近似PIFO调度,根据到达数据包的秩大小动态更新队列入队阈值.	多队列	近似调度	利用多个FIFO调度,大幅减少数据包调度错误,且可在现有交换机上部署.	无法解决饥饿问题,高rank的数据包可能长期滞留在低优先级队列中,无法得到及时的转发.
PACKS[7]	先通过准入控制,丢弃秩较大的包,再动态调整包秩与FIFO队列间的映射关系,减少调度错误.	多队列	近似调度	兼顾了AIFO和SP-PIFO的优势,大幅地减少了数据包调度错误.	依赖于滑动窗口预测数据包rank分布情况,当面对突发流时,可能无法准确捕捉数据包rank的变化情况.
BMW-Tree[8]	通过引入多路分支的平衡树结构来优化插入和删除操作.	分层队列	精确调度	首个精确实现PIFO的方案,吞吐量得到显著提升.	新的流水线设计复杂度高,且需大量SRAM存储节点信息,部署难度大.

3.2　高并发流量的调度机制研究

高并发流量的调度机制主要关注在高带宽和大规模流量场景下如何有效地提升调度器的并发处理能力和系统吞吐量. 它们在保持基本调度功能的同时,通过优化调度路径、简化排序逻辑或分层管理调度状态等方式,降低硬件实现成本,提升可扩展性与运行效率. 这类机制中具有代表性的方案有:PCSQ[63]、CIPO[64]、DR-PIFO[65]、PCQ[66]、Sifter[67]、Eiffel[68]和BBQ[69].

针对以往的调度机制(如PIFO、PIEO等)难以适应循环调度和无法提供严格的端到端延迟保证等问题,PCSQ[63]提出了一种可编程循环队列,能够高效处理大量并发流,通过循环调度机制在网络中实现高精度的确定性数据传输. 具体来说,数据包入队列解析SRv6头部,确定其端口和循环时间. 并采用时钟和频率同步模块维护当前传输队列,按固定时间间隔轮转队列,确保了端到端可预测的低时延与低抖动传输. 最终,PCSQ通过循环映射机制计算数据包的下一个循环标签,使数据包在下一跳节点正确入队. 但因其计算开销、缓存资源消耗等问题,PCSQ难以在大规模复杂网络中部署. CIPO[64]提出了一种分类入队-推送出队机制,增强可扩展性. 该机制通过滑动窗口跟踪最近数据包的秩和转发条件,综合考虑时间维度和空间维度,将数据包分类至

不同优先级队列，确保紧急数据包优先传输，减少关键流量的延迟．出队时，CIPO维护每个FIFO队列的上一次调度时间戳，以检测队列的调度状态．当某个FIFO头部数据包已满足调度需求，但长时间仍未被调度时，系统会提升该数据包的优先级，防止低优先级数据包出现饥饿现象．CIPO在资源占用更低的前提下，支持更多并发流量的调度需求．但CIPO依赖滑动窗口估算包转发时间进行数据包调度，当流量突发时，可能造成估算失误，影响系统调度效果．DR-PIFO[65]针对传统PIFO无法动态调整数据包的秩这一问题，提出了一种动态调整数据包秩的调度机制．其通过重新排名单元检查是否需要调整队列中数据包的秩，并更新流的优先级信息，以确保重要流量能够及时处理．DR-PIFO仅进行局部小规模排序与秩的动态调整，这一设计使其能够在高并发流量环境下仍保持较好的调度性能．

PCQ[66]提出了一种可编程日历队列机制，将日历队列(Calendar Queue，CQ[70])调度机制与可编程数据包处理相结合，旨在解决传统包调度器在动态优先级调整和可扩展性方面的局限性，从而满足不同的网络调度需求，并提升对高并发流量的处理能力．其结构如图2所示，每个物理FIFO队列具有相同服务时间窗口，每个队列对应未来的特定时间段．当数据包达到时，PCQ根据调度策略，将其插入相应队列，并始终从当前时间队列(如Day1)转发数据包．日历队列以固定时间间隔轮转，可满足秩随时间推移逐渐提升的数据包的进队需求，从而支持复杂的调度算法．但因队列数量有限，PCQ支持秩的范围受到约束，可能导致频繁的数据包循环与跨队列迁移，从而增加过多的内存访问开销．为同时兼顾调度精度和队列容量，Sifter[67]设计了一种无包反转且大容量的可编程包调度器，其结构如图3所示．旋转日历队列(Rotating Calendar Queue，RCQ)由多个FIFO队列结构组成，提供了较大的队列容量，根据数据包的秩进行粗略划分，有效降低了调度排序的复杂度，提升了系统在高并发流量下的处理效率．其中，Mini-PIFO负责将数据包严格按照秩大小排序并转发．当Mini-PIFO中数据包数量低于阈值时，系统将从RCQ中筛选排名较小的数据包进行转发，以确保充分利用带宽．然而，若Mini-PIFO筛选阈值设置不当，可能导致RCQ中数据包频繁筛选，影响调度性能．

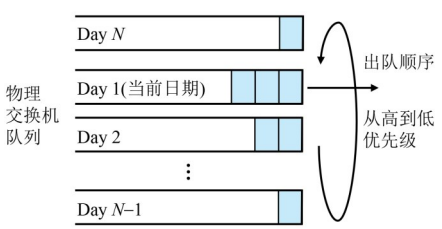

图2 PCQ数据包出队顺序

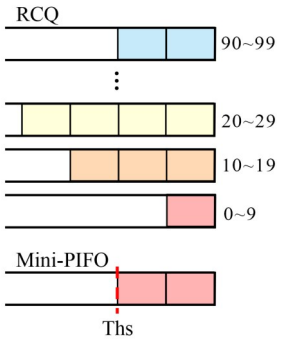

图3 Sifter队列结构

与硬件调度器相比，软件调度器具备开发周期短、功能扩展性强以及部署位置灵活等优势．基于数据包排名本质上是整数排序，Eiffel[68]提出了一种基于查找首个位置(Find First Set，FFS)指令的整数优先级队列，旨在实现数据包的快速出队操作，以应对高并发流量．具体而言，Eiffel利用FFS指令在$O(1)$时间内快速查找秩最小的数据包，并采用循环位图结构存储队列状态，确保能够容纳更大范围的秩值，同时提升队列操作效率．当秩范围超出FFS的适用范围时，Eiffel将采用基于曲率函数的梯度队列，以代数计算替代FFS位运算，在$O(1)$时间内近似定位到秩最小的数据包．然而，该方法依赖近似计算，可能导致调度精度下降，难以保障时延敏感应用的服务质量．

此外，BBQ[69]也在设计中引入了FFS指令，旨在降低队列的排序开销．不同的是，BBQ是一种为硬件调度器设计的整数优先级队列结构，将优先级量化为有限的整数范围，通过分层位图树递归查找最高有效位快速定位目标优先级桶，并结合双向链表实现高效插入与删除．同时，BBQ采用深度流水线化硬件架构，将操作拆分为多级并行阶段，配合使用子树计数器和推测执行技术消除数据冲突，确保流水线无阻塞操作．BBQ位图树结构支持逻辑分区，允许单物理队列划分为多逻辑队列(如按租户或端口)，实现资源复用．但因其基于整数优先级队列，调度精度受限于优先级量化范围，难以满足高精度调度需求．

面对高速网络和大规模流量时，数据包调度器的可扩展性和调度精度尤为重要．例如，调度器若仅支持有限的秩范围，将出现秩压缩问题，导致本应被赋予不同秩的数据包被映射到同一秩，极大降低调度精度，也会增加网络的延迟和抖动，将对需要高精度调度和实时性要求高的场景(如工业物联网、远程医疗、自动驾驶)产生严重影响．数据包调度器难以同时兼得可扩展性和高精度，尽管上述方案通过多种不同方式，在调度精度和扩展性之间找到了平衡，但是提升可编程调度器的可扩展性仍是未来值得深入探讨的重要研究方向之一．本节讨论的调度机制及其对比分析如表2所示．

表2　高并发流量的调度机制研究

调度机制	工作原理	队列管理	调度精度	优势	劣势
PCSQ[63]	每个数据包进入队列时通过解析SRv6头部,确定其传输端口和循环时间.随后,按固定时间间隔轮转队列,确保数据的可靠传输.	多队列	精确调度	严格保障数据包传输延迟的上限,适用于确定性传输的应用场景.	因其计算开销、时间同步要求、缓存资源消耗等问题,难以大规模部署.
CIPO[64]	通过滑动窗口机制,实时跟踪近期数据包的秩和转发条件,基于二维分类算法决定包出列顺序.	多队列	近似调度	近似实现PIEO,具有饥饿检测和预防机制,且仅消耗较少硬件资源.	产生流量突发时,滑动窗口可能错误地估算包转发时间,影响系统调度效果.
DR-PIFO[65]	根据实时流量变化灵活调整数据包的秩,并集成了强大的错误检测机制,避免在秩动态变化的情况下出现饥饿问题.	单队列	近似调度	可动态提升数据包的秩,解决饥饿问题.	频繁秩更新可能导致数据包出队顺序波动,降低调度可预测性,影响高QoS需求应用的服务质量.
PCQ[66]	结合了日历队列思想和可编程数据平面技术,能够支持动态调整数据包秩的调度算法.	多队列	近似调度	支持数据包秩的动态调整,实现更多调度算法.	因队列数量有限,存在频繁数据包循环与跨队列迁移,易造成较高内存访问开销.
Sifter[67]	首先采用RCQ根据数据包rank进行粗粒度划分,再通过Mini-PIFO精确调度.	多队列	精确调度	设计了一种大容量且无包反转的调度器.	Mini-PIFO易成为调度瓶颈,其容量大小的选择和阈值的设定将对数据包调度性能造成影响.
Eiffel[68]	通过扩展PIFO模型并使用FFS指令和近似梯队队列.	多队列	近似调度	将数据包的入队操作保持在$O(1)$复杂度.	近似梯度队列依赖近似计算,难保障时延敏感应用需求.
BBQ[69]	将数据包的秩量化为有限的整数范围,并通过分层位图树快速查找,实现快速的数据包处理操作.	分层队列	近似调度	能够在$O(1)$复杂下完成数据包处理,并支持队列逻辑分区.	其深度流水线设计需要复杂的硬件实现.优先级范围被离散化,降低了调度精度.

3.3　公平保障的调度机制研究

公平保障的调度机制通常指系统能够根据流的权重或需求,合理分配带宽资源,避免某些流长期占据优势而导致其他流得不到服务.这类机制通常采用近似方式模拟经典公平策略,常见方法包括速率控制、轮转调度、流分类与分组排序等,旨在有限的硬件资源条件下实现流级别的公平调度.本节将介绍一系列适用于数据中心网络中可编程调度场景的经典流量公平调度方案:AFQ[28]、EFQ[71]、A^{2}FQ[72]、Gearbox[73]、Cebinae[74]、HLS[75]和SQ-EWFQ[76].

AFQ[28]提出了一种在可编程交换机上实现近似公平队列的调度机制,旨在降低传统公平排队的计算与存储成本.AFQ采用Count-Min Sketch[77]来存储每流的排队信息,以减少交换机的内存需求,同时结合多个FIFO队列近似实现公平调度.具体来说,AFQ设计了一种轮转严格优先级调度算法,数据包根据流的发送字节数计算出发轮次,并被分配到对应的FIFO队列.出队时,按轮次顺序清空队列,并在队列清空后降低其优先级.该方式避免了复杂的每流状态管理和排序操作,能够在高吞吐量的交换机上以线速运行.然而,商用交换机支持的队列数量不足以支撑AFQ应对突发流量,从而易发生过多丢包问题.如图4所示,在当前时刻,流1已将系统为其分配的缓存占满;在下一时刻,虽然队列中有足够的缓存空间,但流1新到达的数据包仍被丢弃.

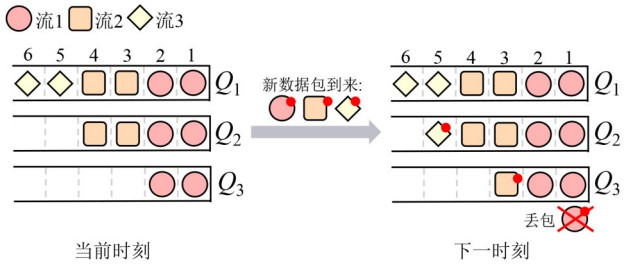

图4　在AFQ方案下实现流间公平

为缓解AFQ因队列数量受限导致的丢包问题,EFQ[71]提出了一种弹性公平队列机制,该机制能够充分利用队列缓存空间.EFQ根据当前活跃流的数量,为每流均匀地分配缓存空间.当数据包到达交换机时,优先进入有可用缓存的最高优先级队列.出队时,EFQ采用类似AFQ机制中的轮转调度方式,确保数据包的按序传输.但EFQ计算缓存分配时依赖于活跃流数,在高并发场景下将退化为AFQ调度.针对同一问题,A^{2}FQ[72]提出了一种自适应队列数量调整机制,通过监控队列深度动态调节有效队列数量,以在共享缓存架构下实现高效缓存利用率和流间带宽公平分配.A^{2}FQ设计了Q_low和Q_high两个阈值动态调整有效队列数量.在队列深度低于Q_low时增加有效队列,提升流缓存分配以优化吞吐量;高于Q_high时减少队列,限制缓存占用,防止新流无缓存可用.但A^{2}FQ仅在数据包出队时评估队列长度并调整有效队列数量,这导致在突

发场景下，A²FQ可能无法及时减少有效队列数以预留缓存空间，造成新流数据因剩余缓存空间不足而丢包，进而影响整体公平性.

Gearbox[73]的设计允许数据包的转发时间在一定范围内偏离严格WFQ的理想转发时间. 它由逻辑上的分层队列结构组成，随着层次的上升，调度的粒度逐渐变粗. 数据包到达时将根据其预计转发时间和每层的调度粒度，精确计算数据包入队的目标队列. 这种设计能够在使用相对较少的FIFO情况下，容纳广泛的数据包转发时间范围，并消除数据包反转问题. Gearbox的层次化数据包管理虽简化了调度器实现，但可能引入额外的延迟. 在最坏情况下，数据包的延迟可能达到WFQ调度器中预期延迟的两倍. Cebinae[74]通过对超出最大-最小公平份额的流量施加惩罚，以增强网络中的公平性. 具体来说，当出端口处于饱和状态时，Cebinae将在出端口利用Heavy-Hitter Cache监控每流的字节数，并基于字节数大小将流划分为瓶颈流或非瓶颈流. 在调度阶段，瓶颈流被分配到低优先级队列，而非瓶颈流被分配到高优先级队列. 通过漏桶机制，Cebinae将瓶颈流超出带宽分配的数据包实施延迟或丢弃操作，从而达到带宽惩罚的效果. HLS[75]则提出了一种采用树状结构的层次链路共享策略，每个节点代表一个类（表示一种特定的流量类别），根类控制整个链路资源，叶子类负责实际传输数据包，内部类介于两者之间，负责将资源逐级分配给叶子类. HLS资源分配流程分为主轮次和剩余轮次. 在主轮次中，HLS通过自上而下的方式为每个有资源需求的类计算配额，资源根据各类的权重逐级分配到叶子类，未使用的资源则保留为残余值，用于后续轮次. 在主轮次结束后，若仍有类持有未使用的资源，HLS将启动剩余轮次，进一步分配剩余资源.

过去实现WFQ算法的调度器都隐含地假设网络流中的数据包是独立的，即丢弃一个数据包不会影响流中后续数据包的到达. 因此，调度器通过丢弃超出公平份额的流量来实现公平性. 然而数据中心里99%的流为TCP(Transmission Control Protocol)流[78]，上述假设并不适用于TCP流，因为TCP发送方通常会以突发的方式发送数据包，并根据数据包丢失等反馈调整发送窗口. 针对这一问题，SQ-EWFQ[76]提出了一种适应TCP流的近似WFQ调度机制，且仅需一个FIFO队列，有效降低了资源消耗. 该机制通过临时提高突发流的权重来实现对TCP流的公平带宽分配，并依靠累计入队字节数与轮次值r的比较确保长期公平性，其中轮次r能反映在过去应获得的公平带宽. 在数据包出队时，r值将根据队列长度与队列深度的比值进行自适应更新，从而有效降低丢包率，还能够防止端口饥饿，实现了突发容忍与资源高效利用的平衡. SQ-EWFQ虽能短期提升突发流权重，但需要一定时间恢复公平性，可能对时延敏感业务造成影响.

上述机制的关键是实现流量调度过程中的公平性，强调按需分配与资源共享的协同. 这类机制能够在资源受限的可编程交换机中，以较低开销实现流级公平调度，具有良好的可扩展性. 然而，由于它们多采用近似的实现方法，在复杂流量场景下，这类机制可能难以保障调度效果. 本节讨论的调度机制及其对比分析如表3所示.

3.4 租户隔离的调度机制研究

租户隔离的调度机制是面向多租户数据中心场景中多个租户（即用户或组织）共享同一网络资源（如网络带宽、网络设备的处理能力）. 能够以租户粒度独立分配和控制资源，避免租户间发生资源争用. 每个租户有独特的业务需求和流量特征，为满足服务水平协议，需确保租户隔离和有保障的资源供应. 本节将介绍一系列经典的面向多租户场景设计的调度方案：vPIFO[9]、QVISOR[79]、HCSFQ[80]、HUG[81]、AHAB[82]、PANIC[83]和Loom[84].

vPIFO[9]通过灵活构建PIFO树支持可编程分层调度，可满足多租户数据中心的服务需求. 在vPIFO中，每个PIFO实例采用BMW-Tree结构[8]实现，并通过轮转放置以实现高效存储. 具体来说，运营商可使用调度描述语言，对不同租户的流量进行规划，并生成相应结构的PIFO树. 在硬件层，由PIFO Engine和PIFO Visor两个核心组件管理多个PIFO实例，并根据编译条件将数据包的P/P操作转化为某个具体PIFO实例的P/P操作以实现精确调度. 但vPIFO实现仍面临着复杂的流水线设计和需要大量存储计算资源等问题. 为高效处理不同租户间调度策略的冲突，QVISOR[79]提出了一种虚拟化数据包调度策略，旨在通过虚拟化调度资源，解决多个租户同时运行不同的调度策略时面临的冲突，从而有效利用网络共享资源. QVISOR充当租户和底层硬件之间的中介，允许租户为其流量自定义策略，以满足各自的服务需求；也允许运营商定义如何分配网络共享资源和指定租户间的调度优先级. 基于这些输入，QVISOR将生成联合调度策略，并将其部署到硬件. QVISOR为传统交换机带来了多租户可编程调度，但其生成的联合调度策略是一种近似调度方案，可能导致部分租户服务质量下降.

针对多租户数据中心租户间的带宽竞争，已有研究提出了多种公平分配方案. 例如，HCSFQ[80]在核心无状态公平性调度（Core-Stateless Fair Queueing, CSFQ）[85]基础上，推广至多级层次结构. HCSFQ利用单个FIFO队列进行数据处理，无需逐包调度和维护每流状态，规避了传统公平队列在存储开销、计算复杂度和扩展性等方面的瓶颈. HCSFQ采用递归计算的方式，利

表3　公平保障的调度机制研究

调度机制	工作原理	优势	劣势
AFQ[28]	根据每流已发送字节数,为数据包计算出发轮次,并结合多FIFO队列实现近似公平调度.	使用了旋转优先级调度,降低了排序的复杂性并保证了低延迟.	其公平性依赖于交换机中可用队列数量,当面对大规模流量时,较少的队列数量难以保持良好的公平性.
EFQ[71]	根据活跃流的数量,为每流动态分配可用缓存,以近似公平调度.	充分利用了交换机的缓存,在应对突发流量时减少不必要的数据包丢失.	在共享缓存架构下,该机制难以保证各流之间的公平带宽分配.
A^{2}FQ[72]	通过动态调整有效队列数,实现高效缓存利用,并保证流间公平性.	在共享缓存架构下,能高效利用缓存空间,同时兼顾低丢包率与调度公平性.	在多突发场景下,难以及时有效调整有效队列数量,影响调度性能.
Gearbox[73]	通过多个逻辑分层队列组合,近似地实现了WFQ调度效果.	仅利用有限的FIFO队列容纳具有广泛转发时间范围的数据包.	分层队列会带来额外的调度延迟,最坏情况下,数据包延迟可能翻倍.
Cebinae[74]	通过对超出公平份额的流量进行概率丢包,以实现近似公平调度.	能够使用较少数量的队列实现公平调度,并兼顾各种拥塞控制协议.	在突发场景下,可能对带宽瓶颈检测不及时,导致调度公平性下降.
HLS[75]	流量带宽分配包括主轮次和剩余轮次,主轮次按权重为每类流分配基础资源,剩余轮次再分配剩余资源.	其带宽分配机制,能够确保每流获得最低的带宽保障,且不因某些应用谎报流量需求而影响带宽分配公平性.	采用多轮机制进行资源分配,在流量频繁波动或带宽需动态调整时,该方法的资源分配可能滞后于网络中实际的需求变化.
SQ-EWFQ[76]	当检测到突发流量时,临时提高该流量的权重,通过限制累计传输字节数确保整体调度的公平性.	具备适应TCP流近似公平调度的能力.	临时提升某些流的权重,可能导致短期内带宽分配不公.

用指数加权平均机制估算每个内部节点的到达速率.并根据估算结果,推算其分配给子节点的公平共享速率.最终,通过概率丢包来限制每个内部节点的带宽,确保层次化的公平调度.但实际应用中,HCSFQ原型模型系统在Tofino 1交换机上最多支持四层,可能难以满足细粒度的流量控制.HUG[81]提出了一种两阶段分配策略,旨在实现最优隔离保障和最大化资源利用率,并避免租户谎报带宽需求.具体来说,HUG首先通过一种基于主导资源的公平分配算法(Dominant Resource Fairness,DRF)计算最优隔离保证,为每个租户分配最小资源份额.在此基础上,再通过局部最大-最小公平性分配剩余资源,并设定租户特定的上限,充分利用剩余带宽.但HUG依赖租户上报的相关性向量来决定资源分配,难以精准捕捉租户的实际需求,若需求在短期内剧烈变化,可能导致资源分配滞后,从而影响整体利用率.AHAB[82]则使用Count-Min Sketch[77]区分流量大小,避免了为数百万用户分配独立的内存,并结合低通滤波估计每个用户的发送速率,对超出速率限制的流进行概率性丢包,从而实现最大-最小公平性.虽然AHAB能够快速完成公平带宽分配,但因数据平面计算能力的限制,对流速率的估算存在误差,可能影响带宽分配的准确性.

PANIC[83]设计了一种混合推/拉式调度器,能够在多个卸载引擎间实现跨租户隔离和低延迟负载均衡.PANIC采用RMT流水线架构,对数据包进行解析和调度,以提升调度灵活性.同时,采用Crossbar互连结构,确保数据包的高效转发.此外,PANIC采用信用管理机制,动态调整拉取式和推送式调度模式以适应不同负载情况,确保计算单元的负载均衡与跨租户的公平调度.但PNIC依赖于PIFO队列进行优先级调度,在超高速网络情况下,PIFO可能成为调度瓶颈,影响系统性能.Loom[84]提出了一种基于有向无环图(Directed Acyclic Graph,DAG)的调度机制,旨在灵活高效地管理不同租户的网络流量.Loom依赖操作系统预先提供调度元数据,并利用Match-Action流水线在数据包读取前计算秩,以降低调度延迟并减少CPU开销.同时,Loom将网络策略抽象为DAG,使调度与速率控制能够以层次化的方式灵活表达,提供更精准的流量管理.此外,其通过批量写入方式合并多个队列的更新请求,有效减少PCIe事务,从而提高数据传输效率.

多租户调度方案通常支持对调度策略的虚拟化映射与策略隔离,普遍具备一定程度的资源划分能力与租户策略自治特性.其优势在于具备良好的管理灵活性与租户隔离性,但在资源开销与调度复杂度方面也引入了新的挑战.本节讨论的调度机制及其对比分析如表4所示.

3.5　小结

通用表达的调度机制旨在提供高度灵活和通用的调度抽象,高并发流量的调度机制旨在支持大量并发流的同时保持线速调度性能,公平保障的调度机制旨在保障不同用户或流量获得公平的带宽,租户隔离的调度机制旨在为多租户提供强隔离性和层次化的资源

表4 租户隔离的调度机制研究

调度机制	工作原理	优势	劣势
vPIFO[9]	通过虚拟化物理PIFO,实现多个逻辑PIFO实例共享一个物理PIFO的计算资源,能够较好地实现租户间的资源隔离.	适应多租户的环境,满足用户多样化需求,同时兼顾高速网络性能.	该方案高性能表现依赖于FPGA和ASIC等硬件平台,在现有交换机上难以部署.
QVISOR[79]	允许租户根据特定需求定义独立调度策略,并将其与运营商的资源分配方案融合,形成联合调度策略.	允许租户和运营商自定义调度策略,灵活性高.	生成的联合调度策略是一种近似调度策略方案,可能导致部分租户服务质量下降.
HCSFQ[80]	将CSFQ推广至多级层次结构,仅维护每个节点聚合流状态,递归地估算内部节点速率并为其分配带宽.	无需维护每流状态和逐包调度,有效降低了存储和计算开销.	在现有可编程交换机上,支持层数受限,难以满足细粒度的流量控制.
HUG[81]	先利用DRF分配策略确保租户能够获得策略安全的最低资源保障,再限制每个租户在各资源上的最大使用量.	资源利用充分,确保租户无法通过谎报获取不正当的资源分配,保障了良好的公平性.	依赖租户上报的相关性向量分配资源,难以精准匹配实际需求,短期剧变可能导致资源分配滞后.
AHAB[82]	通过Count-Min Sketch区分流量大小,结合低通滤波估计流量发送速率,对超速流量进行概率丢包以实现公平调度.	避免为数百万的用户分配独立的内存,能够快速完成公平带宽分配.	存储机制和数据平面计算能力的限制,对于流速率的估算存在误差,且计算精度有所下降,影响带宽分配的准确性.
PANIC[83]	通过集中式调度器,并结合信用管理机制,在不同流量负载下动态调整拉取式和推送式调度模式,确保租户间的公平调度.	推/拉式混合调度能够达到纳秒级的数据包调度和负载均衡,并能够确保租户间的公平调度.	集中式调度器基于PIFO实现,在超高速网络下可能成为调度性能瓶颈.
Loom[84]	将网络策略建模为有向无环图,可以以层次方式表达调度逻辑,从而更灵活地管理不同类型的流量.	基于DAG的层次调度设计,有助于在网络负载较高时维持各应用之间的公平资源分配.	更适用于单机多租户环境,对于需要跨多个节点协调的分布式数据中心,其调度能力存在一定局限性.

分配. 尽管它们各自侧重于解决不同的核心问题,使得它们在表达力、吞吐率、公平性与隔离性表现不尽相同,但它们可协同合作实现多应用目标. 例如,在多租户场景下,租户间使用租户隔离的调度机制实现差异化服务,租户内则可使用公平保障的调度机制实现公平带宽分配或采用通用表达的调度机制根据实际需求灵活修改调度逻辑. 高并发流量的调度机制以提升并发处理能力为目标,同样可扩展至多租户场景中以支持更多租户数量. 总体上,通用表达、高并发流量、公平保障和租户隔离的调度机制各有优点和缺点,如表5所示.

表5 各类调度机制的综合对比

机制类型	优点	缺点	表达力	吞吐率	公平性	隔离性
通用表达的调度机制	支持灵活的调度策略配置,部署在商用硬件.	近似方案调度精度受限,精确方案硬件资源消耗高.	可表达多种调度算法	轻负载下性能好	取决于调度算法	未考虑租户间的隔离性
高并发流量的调度机制	支持大规模的流并发传输,可扩展能力强.	对参数配置敏感,适应复杂场景的能力有限.	可表达多种调度算法	获得较高的吞吐率	未考虑流间的公平性	未考虑租户间的隔离性
公平保障的调度机制	避免复杂的流量管理,低开销实现流间带宽公平.	近似实现公平调度,难以应对多突发和复杂网络场景.	仅能表达公平类调度算法	轻负载下性能好	保证流间带宽公平分配	实现了流间的隔离性
租户隔离的调度机制	支持不同租户独立配置调度策略,实现资源隔离.	调度架构复杂,系统开销较大,且资源划分粒度较粗.	仅能表达租户隔离类的调度算法	轻负载下性能好	实现租户间的带宽公平分配	实现了租户间的隔离性

4 结束语

基于可编程数据平面的调度机制已经得到学术界和工业界的广泛关注和研究. 本文针对“需求发掘—机制设计—方案实施”3个层面所面临的问题,自顶向下地梳理并展望了数据中心可编程数据平面的调度机制的新研究方向.

(1)调度机制应满足新型应用需求. 例如,调度机制需满足分布式深度学习训练(Deep Learning Training, DLT)需求. 分布式DLT模型训练过程中,计算节点之间在同步模型参数与梯度等信息时所产生的流量具有低熵、强突发和同步要求等特性[86],同时并发DLT作业之间存在通信争用导致计算资源利用率低的问题. 然而,当前数据包调度机制仅考虑网络资源而忽视了计算资源的使用情况,难以适用于大规模AI模型训练场景. 例如,调度机制需考虑全局端到端时延约束. 在数据中心网络中,具有端到端延迟约束的流量一旦传输超时便失效,直接影响用户体验与业务收益[87]. 然而,

现有数据包调度方案仅考虑本地最优调度忽略了多跳路径上的全局拥塞,难以满足确定性的端到端延时目标. 因此,如何设计合适的数据包调度机制应对新型应用需求值得进一步研究.

(2)调度机制需与缓存管理协同设计. 通常,交换机内部数据包处理呈现层次化结构,新到达的数据包先在缓存管理模块做准入控制,再由数据包调度机制决定传输顺序. 然而,当前数据包调度机制忽略了缓存管理的作用,易导致低优先级数据包长期占据大量缓存,使得后续到达的高优先级数据包因缓存不足而被丢弃. 该问题在每端口每Gbps的缓存空间越来越小的趋势下变得尤为突出[88]. 因此,如何协同缓存管理模块和数据包调度模块值得深入研究.

(3)调度机制的实施需考虑资源限制. 首先,调度机制卸载至数据处理器的收益与延迟代价需权衡. 为降低CPU开销,通常从主机卸载部分数据包处理与调度任务到数据处理器(DPU)[89]. 然而,由于DPU的计算与内存资源有限,高负载场景中易引入额外的排队与处理延迟,进而增加尾延迟[90]. 其次. 调度机制与智能数据平面(Intelligent Data Plane,IDP)协同设计受到片上计算与存储资源的限制. 相较于在控制平面和终端主机部署机器学习模型,IDP在可编程数据平面技术部署机器学习模型,能在保持线速处理的同时有效降低决策延迟. 已有研究在计算与存储资源受限的可编程交换机上成功部署了简单的决策树模型,用于实现流量分类、异常检测等分析任务[91~94]. 但由于调度算法的复杂性和多样性,目前缺乏IDP在流量调度方面的应用和部署. 因此,在有限的计算和存储资源下,如何更好地实施调度机制亟需进一步研究.

参考文献

[1] 中国信息通信研究院云计算与大数据研究所. 数据中心产业图谱研究报告[R/OL]. (2022-01)[2025-11-10]. http://www.caict.ac.cn/kxyj/qwfb/ztbg/202201/P020220125529907466991.pdf.

[2] NOORMOHAMMADPOUR M, RAGHAVENDRA C S. Datacenter traffic control: Understanding techniques and tradeoffs[J]. IEEE Communications Surveys & Tutorials, 2018, 20(2): 1492-1525.

[3] SIVARAMAN A, SUBRAMANIAN S, ALIZADEH M, et al. Programmable packet scheduling at line rate[C]//Proceedings of the 2016 ACM SIGCOMM Conference. New York: ACM, 2016: 44-57.

[4] SHRIVASTAV V. Fast, scalable, and programmable packet scheduler in hardware[C]//Proceedings of the ACM Special Interest Group on Data Communication. New York: ACM, 2019: 367-379.

[5] YU Z L, HU C H, WU J F, et al. Programmable packet scheduling with a single queue[C]//Proceedings of the ACM SIGCOMM 2021 Conference. New York: ACM, 2021: 179-193.

[6] ALCOZ A G, DIETMULLER A. SP-PIFO: Approximating push-in first-out behaviors using strict-priority queues[C]//Symposium on Networked Systems Design and Implementation. California: USENIX Association, 2020: 59-76.

[7] ALCOZ A G, VASS B, RETVARI G, et al. Everything matters in programmable packet scheduling[EB/OL]. (2023-08-01)[2025-04-22]. https://arxiv.org/abs/2308.00797.

[8] YAO R Y, ZHANG Z Y, FANG G J, et al. BMW tree: Large-scale, high-throughput and modular PIFO implementation using balanced multi-way sorting tree[C]//Proceedings of the ACM SIGCOMM 2023 Conference. New York: ACM, 2023: 208-219.

[9] ZHANG Z Y, CHEN S L, YAO R Y, et al. vPIFO: Virtualized packet scheduler for programmable hierarchical scheduling in high-speed networks[C]//Proceedings of the ACM SIGCOMM 2024 Conference. New York: ACM, 2024: 983-999.

[10] SHREEDHAR M, VARGHESE G. Efficient fair queueing using deficit round robin[C]//Proceedings of the Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. New York: ACM, 1995: 231-242.

[11] HARCHOL-BALTER M, SCHROEDER B, BANSAL N, et al. Size-based scheduling to improve web performance[J]. ACM Transactions on Computer Systems, 2003, 21(2): 207-233.

[12] DEMERS A, KESHAV S, SHENKER S. Analysis and simulation of a fair queueing algorithm[C]//Symposium Proceedings on Communications Architectures & Protocols. New York: ACM, 1989: 1-12.

[13] LEUNG J Y T. A new algorithm for scheduling periodic, real-time tasks[J]. Algorithmica, 1989, 4(1): 209-219.

[14] MCKEOWN N, ANDERSON T, BALAKRISHNAN H, et al. OpenFlow: Enabling innovation in campus networks[J]. ACM SIGCOMM Computer Communication Review, 2008, 38(2): 69-74.

[15] KREUTZ D, RAMOS F M V, ESTEVES VERISSIMO P, et al. Software-defined networking: A comprehensive survey[J]. Proceedings of the IEEE, 2015, 103(1): 14-76.

[16] BOSSHART P, GIBB G, KIM H S, et al. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN[J]. ACM SIGCOMM Computer

Communication Review, 2013, 43(4): 99-110.
[17] CHOLE S, FINGERHUT A, MA S, et al. dRMT: Disaggregated programmable switching[C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. New York: ACM, 2017: 1-14.
[18] MIAO R, ZENG H Y, KIM C, et al. SilkRoad: Making stateful layer-4 load balancing fast and cheap using switching asics[C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. New York: ACM, 2017: 15-28.
[19] LEE J, MIAO R, KIM C, et al. Stateful layer-4 load balancing in switching ASICs[C]//Proceedings of the SIGCOMM Posters and Demos. New York: ACM, 2017: 133-135.
[20] SIVARAMAN V, NARAYANA S, ROTTENSTREICH O, et al. Heavy-hitter detection entirely in the data plane[C]//Proceedings of the Symposium on SDN Research. New York: ACM, 2017: 164-176.
[21] POPESCU D A, ANTICHI G, MOORE A W. Enabling fast hierarchical heavy hitter detection using programmable data planes[C]//Proceedings of the Symposium on SDN Research. New York: ACM, 2017: 191-192.
[22] HARRISON R, CAI Q Z, GUPTA A, et al. Network-wide heavy hitter detection with commodity switches[C]//Proceedings of the Symposium on SDN Research. New York: ACM, 2018: 1-7.
[23] SAPIO A, ABDELAZIZ I, ALDILAIJAN A, et al. In-network computation is a dumb idea whose time has come[C]//Proceedings of the 16th ACM Workshop on Hot Topics in Networks. New York: ACM, 2017: 150-156.
[24] SAPIO A, ABDELAZIZ I, CANINI M, et al. DAIET: A system for data aggregation inside the network[C]//Proceedings of the 2017 Symposium on Cloud Computing. New York: ACM, 2017: 626-626.
[25] 李丹, 陈贵海, 任丰原, 等. 数据中心网络的研究进展与趋势[J]. 计算机学报, 2014, 37(2): 259-274.
LI D, CHEN G H, REN F Y, et al. Data center network research progress and trends[J]. Chinese Journal of Computers, 2014, 37(2): 259-274. (in Chinese)
[26] CAVIUM NETWORKS. Xpliant ethernet switch silicon [EB/OL]. (2013-01-04)[2025-10-10]. https://datasheet.octopart. com/CN6130-110SV-G-Cavium-Networks-datasheet-26366670.pdf.
[27] INTEL. FlexPipe packet processing pipeline[EB/OL]. (2017-07) [2025-10-10]. https://www. intel. com/content/dam/www/public/us/en/documents/datasheets/ethernet-switch-fm5000-fm6000-datasheet.pdf.
[28] SHARMA N K, LIU M, ATREYA K, et al. Approximating fair queueing on reconfigurable switches[C]//Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation. New York: ACM, 2018: 1-16.
[29] INTEL. Stratix V FPGA[EB/OL]. (2024-12-20)[2025-10-10]. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/stratix-v/stx5_51001.pdf.
[30] INTEL. Stratix 10 GX/SX FPGA[EB/OL]. (2024-06-28) [2025-10-10]. https://www.intel.com/content/dam/www/programmable/us/en/pdfs/literature/hb/stratix-10/s10-over - view.pdf.
[31] BURSTEIN I. Nvidia data center processing unit (DPU) architecture[C]//2021 IEEE Hot Chips 33 Symposium. Piscataway: IEEE, 2021: 1-20.
[32] NVIDIA. BlueField SmartNIC product brief[EB/OL]. (2020)[2025-10-01]. https://network. nvidia. com/files/doc-2020/pb-bluefield-smart-nic.pdf.
[33] MICHEL O, BIFULCO R, RETVARI G, et al. The programmable data plane: Abstractions, architectures, algorithms, and applications[J]. ACM Computing Surveys, 2021, 54(4): 1-36.
[34] LIU W X, LIANG C, CUI Y, et al. Programmable data plane intelligence: Advances, opportunities, and challenges[J]. IEEE Network, 2023, 37(5): 122-128.
[35] HAUSER F, HÄBERLE M, MERLING D, et al. A survey on data plane programming with P4: Fundamentals, advances, and applied research[J]. Journal of Network and Computer Applications, 2023, 212: 103561.
[36] NUNES B A A, MENDONCA M, NGUYEN X N, et al. A survey of software-defined networking: Past, present, and future of programmable networks[J]. IEEE Communications Surveys & Tutorials, 2014, 16(3): 1617-1634.
[37] PFAFF B, PETTIT J, KOPONEN T, et al. The design and implementation of open vSwitch[C]//Proceedings of the 12th USENIX Conference on Networked Systems Design and Implementation. New York: ACM, 2015: 117-130.
[38] PANDA A, HAN S J, JANG K, et al. NetBricks: Taking the V out of NFV[C]//12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16). California: USENIX Association, 2016: 203-216.
[39] MOLNÁR L, PONGRÁCZ G, ENYEDI G, et al. Dataplane specialization for high-performance OpenFlow software switching[C]//Proceedings of the 2016 ACM SIGCOMM Conference. New York: ACM, 2016: 539-552.
[40] HAN S J, JANG K, PANDA A, et al. SoftNIC: A software NIC to augment hardware[EB/OL]. (2015-5-27)[2025-

4-22]. https://courses. grainger. illinois. edu/ece598hpn/fa2022/papers/softnic.pdf.

[41] HAN S J, JANG K, PARK K, et al. PacketShader: A GPU-accelerated software router[J]. ACM SIGCOMM Computer Communication Review, 2010, 40(4): 195-206.

[42] BARACH D, LINGUAGLOSSA L, MARION D, et al. High-speed software data plane via vectorized packet processing[J]. IEEE Communications Magazine, 2018, 56(12): 97-103.

[43] HØILAND-JØRGENSEN T, BROUER J D, BORKMANN D, et al. The eXpress data path: Fast programmable packet processing in the operating system kernel[C]//Proceedings of the 14th International Conference on Emerging Networking EXperiments and Technologies. New York: ACM, 2018: 54-66.

[44] RIZZO L. Netmap: A novel framework for fast packet I/O[C]//21st USENIX Security Symposium (USENIX Security 12). California: USENIX Association, 2012: 101-112.

[45] TOOTOONCHIAN A, PANDA A, LAN C, et al. ResQ: Enabling SLOs in network function virtualization[C]//15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18). California: USENIX Association, 2018: 283-297.

[46] SUN C, BI J, ZHENG Z L, et al. NFP: Enabling network function parallelism in NFV[C]//Proceedings of the Conference of the ACM Special Interest Group on Data Communication. New York: ACM, 2017: 43-56.

[47] KULKARNI S G, ZHANG W, HWANG J, et al. NFVnice: Dynamic backpressure and scheduling for NFV service chains[J]. ACM Transactions on Networking, 2020, 28(2): 639-652.

[48] KATSIKAS G P, BARBETTE T, KOSTIC D, et al. Metron: NFV service chains at the true speed of the underlying hardware[C]//Symposium on Networked Systems Design and Implementation. California: USENIX Association, 2018: 171-186.

[49] CARPEGNA A, SAVINO A, DI CARLO S. Spiker+: A framework for the generation of efficient spiking neural networks FPGA accelerators for inference at the edge[J]. IEEE Transactions on Emerging Topics in Computing, 2025, 13(3): 784-798.

[50] BREBNER G, JIANG W R. High-speed packet processing using reconfigurable computing[J]. IEEE Micro, 2014, 34(1): 8-18.

[51] LIU H D, QIAN Y J, LIANG Y Q, et al. A high-performance accelerator for real-time super-resolution on edge FPGAs[J]. ACM Transactions on Design Automation of Electronic Systems, 2024, 29(3): 1-25.

[52] HE Z H, KOROLIJA D, ZHU Y, et al. ACCL+: An FPGA-Based collective engine for distributed applications[C]//18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24). California: USENIX Association, 2024: 211-231.

[53] KAWSER AHMED M, PANOFF KEALOHA M, MANDEBI MBONGUE J, et al. Multi-tenant cloud FPGA: A survey on security, trust, and privacy[J]. ACM Transactions on Reconfigurable Technology and Systems, 2025, 18(2): 1-44.

[54] 唐维伟, 钟胜, 卢金仪, 等. 基于 FPGA 的 Skynet 网络结构优化及高时效实现[J]. 电子学报, 2023, 51(2): 314-323. TANG W W, ZHONG S, LU J Y, et al. Network structure optimization and high-efficiency implementation of skynet based on FPGA[J]. Acta Electronica Sinica, 2023, 51(2): 314-323. (in Chinese)

[55] MELLANOX TECHNOLOGIES. Mellanox Innova™-2 Flex open programmable SmartNIC[EB/OL]. (2020)[2025-10-10]. https://network. nvidia. com/files/doc-2020/pb-innova-2-flex.pdf.

[56] BOSSHART P, DALY D, GIBB G, et al. P4: Programming protocol-independent packet processors[J]. 2014, 44(3): 87-95.

[57] GRANT S, YELAM A, BLAND M, et al. SmartNIC performance isolation with FairNIC: Programmable networking for the cloud[C]//Proceedings of the Annual Conference of the ACM Special Interest Group on Data Communication on the Applications, Technologies, Architectures, and Protocols for Computer Communication. New York: ACM, 2020: 681-693.

[58] BENNETT J C R, ZHANG H. Hierarchical packet fair queueing algorithms[J]. IEEE/ACM Transactions on Networking, 1997, 5(5): 675-689.

[59] TURNER J. New directions in communications (or which way to the information age?) [J]. IEEE Communications Magazine, 1986, 24(10): 8-15.

[60] ZHANG H, FERRARI D. Rate-controlled static-priority queueing[C]//Proceedings of the IEEE INFOCOM 1993 Conference. Piscataway: IEEE, 1993: 227-236.

[61] ZHENG J Q, JIANG Y N, TIAN B C, et al. Supporting multi-dimensional and arbitrary numbers of ranks for software packet scheduling[C]//2020 IEEE/ACM 28th International Symposium on Quality of Service. Piscataway: IEEE, 2020: 1-10.

[62] LIU C L, LAYLAND J W. Scheduling algorithms for

multiprogramming in a hard-real-time environment[J]. Journal of the ACM, 1973, 20(1): 46-61.

[63] HUANG Y D, WANG S, ZHU S Y, et al. Poster: Programmable cycle-specified queue for deterministic networking[C]//Proceedings of the ACM SIGCOMM 2023 Conference. New York: ACM, 2023: 1132-1134.

[64] GUO F, SUN S D, HU J J, et al. CIPO: Efficient, lightweight and programmable packet scheduling[J]. Computer Networks, 2024, 245: 110355.

[65] ELBEDIWY M, PONTIKAKIS B, GHAFFARI A, et al. DR-PIFO: A dynamic ranking packet scheduler using a push-in-first-out queue[J]. IEEE Transactions on Network and Service Management, 2024, 21(1): 355-371.

[66] SHARMA N K, ZHAO C X Y, LIU M, et al. Programmable calendar queues for high-speed packet scheduling[C]//17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20). California: USENIX Association, 2020: 685-699.

[67] GAO P X, DALLEGGIO A, LIU J J, et al. Sifter: An inversion-free and large-capacity programmable packet scheduler[C]//21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24). California: USENIX Association, 2024: 75-95.

[68] SAEED A, ZHAO Y M, DUKKIPATI N, et al. Eiffel: Efficient and flexible software packet scheduling[C]//16th USENIX Symposium on Networked Systems Design and Implementation (NSDI 19). California: USENIX Association, 2019: 17-32.

[69] ATRE N, SADOK H, SHERRY J. BBQ: A fast and scalable integer priority queue for hardware packet scheduling[C]//21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24). California: USENIX Association, 2024: 455-475.

[70] BROWN R. Calendar queues: A fast 0 (1) priority queue implementation for the simulation event set problem[J]. Communications of the ACM, 1988, 31(10): 1220-1227.

[71] LIU J L, HUANG J W, JIANG N, et al. Achieving high utilization for approximate fair queueing in data center[C]//2020 IEEE 40th International Conference on Distributed Computing Systems. Piscataway: IEEE, 2021: 932-942.

[72] SHAN D F, PENG G Y, REN S Z, et al. Adaptive approximate fair queueing for shared-memory programmable switches[J]. IEEE Transactions on Network Science and Engineering, 2024, 11(4): 3563-3576.

[73] GAO P X, DALLEGGIO A, XU Y, et al. Gearbox: A hierarchical packet scheduler for approximate weighted fair queuing[C]//19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22). California: USENIX Association, 2022: 551-565.

[74] YU L C, SONCHACK J, LIU V. Cebinae: Scalable in-network fairness augmentation[C]//Proceedings of the ACM SIGCOMM 2022 Conference. New York: ACM, 2022: 219-232.

[75] LUANGSOMBOON N, LIEBEHERR J. HLS: A packet scheduler for hierarchical fairness[C]//2021 IEEE 29th International Conference on Network Protocols. Piscataway: IEEE, 2021: 1-11.

[76] CHEN W, TIAN Y, YU X, et al. Enhancing fairness for approximate weighted fair queueing with a single queue[J]. IEEE/ACM Transactions on Networking, 2024, 32(5): 3901-3915.

[77] CORMODE G, MUTHUKRISHNAN S. An improved data stream summary: The count-Min sketch and its applications[J]. Journal of Algorithms, 2005, 55(1): 58-75.

[78] ALIZADEH M, GREENBERG A, MALTZ D A, et al. Data center TCP (DCTCP)[C]//Proceedings of the ACM SIGCOMM 2010 Conference. New York: ACM, 2010: 63-74.

[79] ALCOZ A G, VANBEVER L. QVISOR: Virtualizing packet scheduling policies[C]//Proceedings of the 22nd ACM Workshop on Hot Topics in Networks. New York: ACM, 2023: 238-244.

[80] YU Z L, WU J F, BRAVERMAN V, et al. Twenty years after: Hierarchical Core-Stateless fair queueing[C]//18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21). California: USENIX Association, 2021: 29-45.

[81] CHOWDHURY M, LIU Z H, GHODSI A, et al. HUG: Multi-Resource fairness for correlated and elastic demands[C]//Symposium on Networked Systems Design and Implementation. California: USENIX Association, 2016: 407-424.

[82] MACDAVID R, CHEN X Q, REXFORD J. Scalable real-time bandwidth fairness in switches[C]//IEEE INFOCOM 2023 - IEEE Conference on Computer Communications. Piscataway: IEEE, 2023: 1423-1434.

[83] LIN J J, PATEL K, STEPHENS B E, et al. PANIC: A high-performance programmable NIC for multi-tenant networks[C]//USENIX Symposium on Operating Systems Design and Implementation. California: USENIX Association, 2020: 243-259.

[84] STEPHENS B E, AKELLA A, SWIFT M M. Loom: Flexible and efficient NIC packet scheduling[C]//Sympo-

sium on Networked Systems Design and Implementation. California: USENIX Association, 2019: 33-46.

[85] STOICA I, SHENKER S, ZHANG H. Core-stateless fair queueing: Achieving approximately fair bandwidth allocations in high speed networks[C]//Proceedings of the ACM SIGCOMM'98 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication. New York: ACM, 1998: 118-130.

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

[87] XU Z C, CHEN X L, ZHU Z Q. INT-assisted adaptive packet scheduling in PDP switches for end-to-end latency control[C]//Proceedings of the IEEE INFOCOM 2025 Conference. Piscataway: IEEE, 2025: 1-10.

[88] BAI W, HU S H, CHEN K, et al. One more config is enough: Saving (DC)TCP for high-speed extremely shallow-buffered datacenters[EB/OL]. (2021) [2025-10-10]. http://baiwei0427.github.io/papers/bcc-infocom2020.pdf.

[89] KASHYAP A, LI Y K, NG D, et al. Understanding the idiosyncrasies of emerging BlueField DPUs[C]//Proceedings of the 39th ACM International Conference on Supercomputing. New York: ACM, 2025: 807-821.

[90] WEI X D, CHENG R X, YANG Y W, et al. Characterizing off-path SmartNIC for accelerating distributed systems[C]//USENIX Symposium on Operating Systems Design and Implementation. California: USENIX Association, 2023: 987-1004.

[91] ZHOU G M, LIU Z T, FU C P, et al. An efficient design of intelligent network data plane[C]//Proceedings of the 32nd USENIX Conference on Security Symposium. New York: ACM, 2023: 6203-6220.

[92] YAN J Z, XU H T, LIU Z T, et al. Brain-on-Switch: Towards advanced intelligent network data plane via NN-Driven traffic analysis at Line-Speed[C]//21st USENIX Symposium on Networked Systems Design and Implementation (NSDI 24). California: USENIX Association, 2024: 419-440.

[93] ZHANG Y C, YAO S, FENG Y, et al. Pegasus: A universal framework for scalable deep learning inference on the dataplane[C]//Proceedings of the ACM SIGCOMM 2025 Conference. New York: ACM, 2025: 692-706.

[94] JAFRI S U, RAO S G, SHRIVASTAV V, et al. Leo: Online ML-based traffic classification at multi-terabit line rate[C]// Symposium on Networked Systems Design and Implementation. California: USENIX Association, 2024: 1573-1591.

作者简介

刘敬玲　女,1994年1月出生于湖南省湘潭市. 现为中南大学计算机学院讲师. 主要研究方向为数据中心网络、互联网视频以及分布式机器学习.
E-mail: jinglingliu@csu.edu.cn

方　浩　男,2002年3月出生于江西省上饶市. 现为中南大学计算机学院硕士研究生. 主要研究方向为数据中心网络流量调度.
E-mail: haofang@csu.edu.cn

何　种　男,2002年7月出生于湖南省郴州市. 现为中南大学计算机学院硕士研究生. 主要研究方向为数据中心网络可编程数据包调度.
E-mail: zhonghe@csu.edu.cn

崔　睿　男,2000年10月出生于山西省晋中市. 现为中南大学计算机学院硕士研究生. 主要研究方向为数据中心网络缓存管理.
E-mail: ruicui@csu.edu.cn

汪子汐　女,2001年5月出生于陕西省宝鸡市. 现为中南大学计算机学院硕士研究生. 主要研究方向为数据中心网络分布式训练流量负载均衡.
E-mail: zixiwang@csu.edu.cn

黄家玮　男,1976年10月出生于湖南省长沙市. 现为中南大学计算机学院教授. 主要研究方向为无线网络和数据中心网络的性能建模、分析与优化.
E-mail: jiaweihuang@csu.edu.cn