计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
62(5): 1262−1289, 2025

以太网 RDMA 网卡综述

黄曼蒂[1] 李 韬[1] 杨 惠[1] 李成龙[2] 张毓涛[1] 孙志刚[1]
[1]（国防科技大学计算机学院 长沙 410073）
[2]（军事科学院国防科技创新研究院 北京 100071）
（mandih@nudt.edu.cn）

Survey on Ethernet RDMA Network Interface Card

Huang Mandi[1], Li Tao[1], Yang Hui[1], Li Chenglong[2], Zhang Yutao[1], and Sun Zhigang[1]
[1]（*College of Computer Science and Technology*, *National University of Defense Technology*, *Changsha* 410073）
[2]（*Defense Innovation Institute*, *Academy of Military Sciences*, *Beijing* 100071）

Abstract With the rapid expansion of data center and the significant increase in network bandwidth, traditional software network protocol stack has high processor overhead and is difficult to meet the needs of many data center applications in terms of throughput, latency and other aspects. Remote direct memory access (RDMA) technology uses the ideas of zero copy, kernel bypass and processor function offloading to read and write remote host memory data with high bandwidth and low latency. Ethernet-compatible RDMA technology is being applied in data centers, and Ethernet RDMA NIC plays a crucial role in its deployment as the main functional bearer device. This overview analyzes from three aspects: architecture, optimization, and implementation evaluation. 1) We summarize the general architecture of Ethernet RDMA NIC and introduce the key functional components; 2) We focus on the optimization techniques in storage resources, reliable transmission and application-related aspects, including optimization of both connection scalability for NIC cache resources and registration access for host memory resources, optimization of congestion control, flow control and retransmission mechanism for lossy Ethernet to achieve reliable transmission, and optimization of different storage types in distributed storage, database system, cloud storage system, and multi-tenant performance isolation, security and programmability for data center applications; 3) Then we investigate different implementation and evaluation methods. Finally, the summary and outlook are given.

Key words remote direct memory access (RDMA); Ethernet RDMA network interface card (NIC); routable RDMA over converged Ethernet (RoCEv2); NIC architecture; NIC optimization; data center network

摘 要 目前数据中心规模迅速扩大和网络带宽大幅度提升，传统软件网络协议栈的处理器开销较大，并且难以满足众多数据中心应用程序在吞吐、延迟等方面的需求. 远程直接内存访问（remote direct memory access, RDMA）技术采用零拷贝、内核旁路和处理器功能卸载等思想，能够高带宽、低延迟地读写远端主机内存数据. 兼容以太网的 RDMA 技术正在数据中心领域展开应用，以太网 RDMA 网卡作为主要功能承载设备，对其部署发挥重要作用. 综述从架构、优化和实现评估 3 个方面进行分析：1）对以太网 RDMA 网卡的通用架构进行了总结，并对其关键功能部件进行了介绍；2）重点阐述了存储资源、可靠传输和应用相关 3 方面的优化技术，包括面向网卡缓存资源的连接可扩展性和面向主机内存资源的注册访问优化，

收稿日期：2023-12-25；**修回日期**：2024-08-27
基金项目：国家国防科技工业局重点实验室基金项目（WDZC20245250113）
This work was supported by the Key Laboratory Foundation of State Administration for Science, Technology and Industry for National Defense（WDZC20245250113）.
通信作者：杨惠（yanghui@nudt.edu.cn）

面向有损以太网实现可靠传输的拥塞控制、流量控制和重传机制优化，面向分布式存储中不同存储类型、数据库系统、云存储系统以及面向数据中心应用的多租户性能隔离、安全性、可编程性等方面的优化工作；3）调研了不同实现方式、评估方式. 最后，给出总结和展望.

关键词 远程直接内存访问；以太网 RDMA 网卡；RoCEv2；网卡架构；网卡优化；数据中心网络

中图法分类号 TP393

DOI: 10.7544/issn1000-1239.202331036 **CSTR:** 32373.14.issn1000-1239.202331036

过去 10 年，数据中心（Microsoft, Alibaba Cloud, Amazon 等）经历大规模的增长，其内部的机器数量已经超过了当今最大的超级计算机[1]，现有数据中心内部服务器通常基于以太网互连. 当前以太网的传输速率已经突破 400 Gbps，甚至向着 800 Gbps, 1.6 Tbps 发展[2]，而 CPU 的处理能力未与网络速度一样快速提升. 使用传统软件网络协议栈进行报文处理一方面会极大增加 CPU 用于网络处理和频繁数据搬移的开销，另一方面无法满足数据中心数据密集型以及内存密集型应用程序（分布式存储、大数据处理、机器学习等）对高带宽和低尾部延迟的严格性能需求[3]. 远程直接内存访问（remote direct memory access, RDMA）技术最初[1]应用于高性能计算领域，可以在不需要 CPU 参与的情况下直接读写远端内存，从而实现高带宽低延迟的数据传输，这契合当前数据中心网络传输需求.

目前 RDMA 主要有 3 种主流实现方式，分别是 IB（InfiniBand），RoCE（RDMA over converged Ethernet）和 iWARP. IB 虽然可以提供最佳性能，但自定义了从链路层到传输层的整套规范，需要专用网卡和交换机支持，不兼容以太网且价格昂贵. RoCE 则对 IB 进行了兼容以太网的扩展，其中第 1 个版本 RoCEv1 将 IB 的链路层替换为以太网协议；RoCEv2（routable RoCE）在 RoCEv1 的基础上进一步将网络层替换为 UDP/IP 协议，允许使用标准以太网交换机进行跨网路由，RoCEv2 的部署只需端系统专用网卡支持. iWARP 同样兼容以太网，但需将整个 TCP/IP（Transmission Control Protocol/Internet Protocol）协议栈卸载到硬件，存在成本、实现复杂度、性能方面的限制. RoCEv2 由于兼容以太网、相对较低的复杂度[4]以及极具竞争力的价格[5]，已经是数据中心网络中部署最广泛的 RDMA 技术[6-7]. RoCEv2 可支持数据中心内和数据中心间的路由，并发挥 RDMA 的传输优势，实现数十至数百 Gbps 的传输速率[8]，从而为众多数据中心应用程序提供高性能保证，因此本文聚焦 RoCEv2，展开对以太网 RDMA 网卡的综述.

随着以太网 RDMA 技术在网络存储、云计算等数据中心场景中受到广泛关注和部署，以太网 RDMA 网卡作为其技术实现的关键组件，发挥着愈发重要的作用. 一方面，网卡需要实现以太网 RDMA 技术中最复杂的部分[9]，如协议卸载、可靠传输、拥塞控制等. 而交换机上需要具备的功能通常较为简单，如常见的基于优先级的流控（piority-based flow control, PFC）[10]和显式拥塞通知（explicit congestion notification, ECN）[11]等. 另一方面，卸载到硬件处理意味着网卡需要存储读写远端内存所需的状态信息，然而网卡的片上缓存和处理逻辑资源有限[12-14]，这更需对网卡的微体系结构具备一定的了解以进行设计和优化.

在工业界，Intel[15]，NVIDIA Mellanox[16]，MARVELL[17]，Huawei Hisilicon[18]，Chelsio[19]，Broadcom[20]等众多供应商已经推出了一系列商用以太网 RDMA 网卡. 谷歌[21-25]、微软[1,5,9,26-36]、阿里巴巴[3,37-39]、字节跳动[12,14,40]等互联网厂商针对以太网 RDMA 技术在云存储等数据中心场景的部署、管理和评估开展研究. 在学术界，国内外也有众多研究团队（清华大学[41-47]、南京大学[13,38,48-51]、中国科学技术大学[4,35,52-53]、中国科学院计算技术研究所[54-58]、湖南大学[34,59]、香港科技大学[14,60]、苏黎世联邦理工学院[1,8,61-64]、普渡大学[22,65-68]、威斯康星大学麦迪逊分校[6,21,69]、卡耐基梅隆大学[70-74]等）针对其架构、优化技术进行一系列研究. 然而，目前缺乏针对以太网 RDMA 网卡这一关键功能承载设备的综述，为其架构、优化和实现评估提供参考. 因此，本文对以太网 RDMA 网卡进行系统综述，主要贡献有 3 点.

1）梳理以太网 RDMA 网卡承担功能，根据目前工业界产品和学术界研究总结网卡的通用架构，并对其报文处理、DMA、可靠传输等关键功能部件进行分析介绍.

2）从存储资源、网络传输和应用相关 3 方面对以太网 RDMA 网卡的优化工作进行分类阐述，主要包括：面向网卡缓存资源的连接可扩展性和面向主

机内存资源的注册访问优化;面向有损以太网实现可靠传输的拥塞控制、流量控制和重传机制优化;面向分布式存储中不同存储类型、数据库系统、云存储系统及数据中心应用的多租户性能隔离、安全性、可编程性等优化工作.

3)梳理工业界和学术界以太网 RDMA 网卡基于 ASIC、FPGA、SoC 和软件等实现方式,并总结目前以太网 RDMA 网卡的评估方式.

1 研究背景

1.1 RDMA 传输机制

1)RDMA 建立连接. RDMA 允许本端用户态应用程序快速读写远端内存,避免了用户态和内核态切换、软件网络协议栈处理的开销. 在正式通信之前,RDMA 本端应用程序需要首先申请注册内存区域(memory region, MR)用于存放收发数据,并与通信远端进行建链协商,提前获取远端主机信息和内存访问权限等,该过程需要双端 CPU 参与. iWARP 和 RoCEv1 基于 CM(communication manager)接口建链,RoCEv2 则同时支持基于 CM 接口和 Socket 接口 2 种建链方式.

2)RDMA 通信原语. RDMA 通过 verbs 编程接口[75]为上层应用提供 2 类通信原语,分别为双边通信原语和单边通信原语. 其中双边通信原语 SEND 和 RECV 配合使用,需要远端 CPU 在本端 SEND 发送消息前提前下发 RECV 指定接收地址;单边通信原语 READ, WRITE, ATOMIC 则不涉及远端 CPU 参与直接读/写/原子性读-修改-写远端内存. 此外,SEND 和 WRITE 还可以携带额外的 4 B 立即数据(使用 SEND/WRITE with imm),该数据不直接写入内存,而是作为状态信息返回给上层应用(需要远端 CPU 参与). 相比双边通信原语,单边通信原语可实现更低的延迟和更高的带宽[7],因此双边通信原语常用于控制、状态信息传递及需要显式通信的场景,单边通信原语 READ, WRITE 常用于大数据块远程搬移. ATOMIC 操作则可用于并发控制和同步,以及帮助实现分布式锁[76]等分布式事务处理所需机制.

在建链协商完成后,应用程序通过 RDMA 通信原语以工作请求(work request, WR)的形式下发(post)远端访问请求. 以 RDMA WRITE 为例,其请求报文格式如图 1 所示,对应的通信流程如图 2 所示. 对于应用程序下发的 WR,由用户态驱动将其转化为工作队列元素(work queue element, WQE)并放入队列对(queue pair, QP)中(图 2 中①). QP 是 RDMA 双端通信的基本单位[57],由发送队列(send queue, SQ)和接收队列(receive queue, RQ)组成,由 SQ 接收应用程序下发的 SEND, READ, WRITE 等请求,由 RQ 接收应用程序下发的 RECV(对应 SEND 请求). 对于 RDMA WRITE 操作,由网卡在获取 RDMA 传输请求(图 2 中②)后取出本端内存数据(图 2 中③)并执行远端内存写入(图 2 中④).

3)RDMA 传输方式. QP 可以被配置为可靠连接(reliable connection, RC)、不可靠连接(unreliable connection, UC)、不可靠数据报(unreliable datagram,

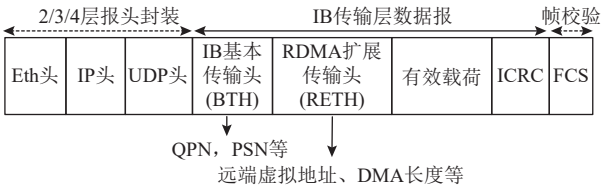


Fig. 1 RoCEv2 WRITE request message format

图 1 RoCEv2 WRITE 请求报文格式

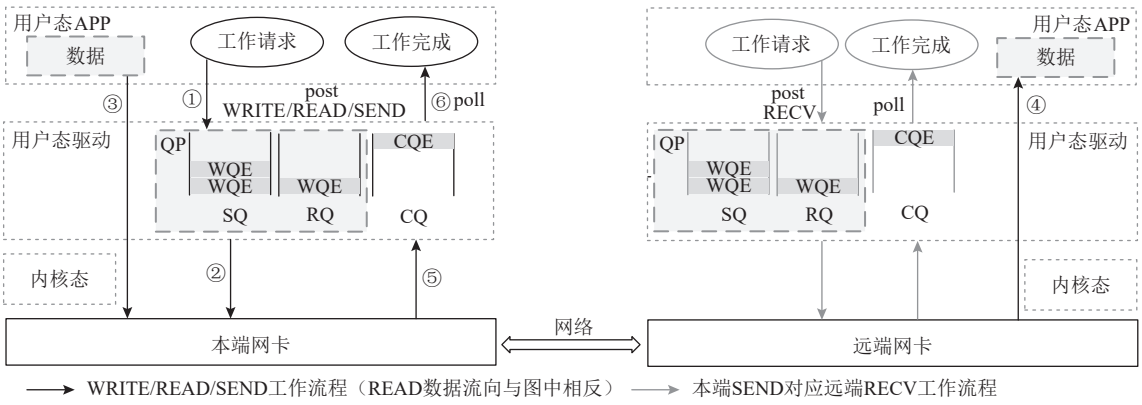


Fig. 2 RDMA communication flow

图 2 RDMA 通信流程

UD)[43]等传输方式. 其中 RC 和 UC 传输方式需要在通信双方之间建立 1 对 1 的 QP 连接, 且 RC 传输方式还需要网卡负责报文的可靠传输, UD 传输方式不保证可靠性但可支持 1 对 1 或 1 对多 UD QP 之间的通信[77]. 不同传输方式下支持的通信原语和消息大小等区别可见表 1.

Table 1 Comparison of RDMA Common Transmission Modes

表 1 RDMA 常用传输方式对比

传输方式	通信原语				可靠性	可扩展性	消息大小/KB
	①	②	③	④			
RC	√	√	√	√*	√	一般	$\leqslant 2^{21}$
UC	√	×	√	×	×	一般	$\leqslant 2^{21}$
UD	√	×	×	×	×	好	⩽4

注：①SEND (SEND with imm)/RECV；②READ；③WRITE (WRITE with imm)；④ATOMIC. “*”表示对 ATOMIC 操作，消息大小仅支持 8 B.

在数据传输完成后, 网卡生成完成队列元素(completion queue element, CQE)并将之放入与 QP 关联的完成队列(completion queue, CQ)中(图 2 中⑤)[57], 最终应用程序由网卡上送(poll)的工作完成(work completion, WC)获知 RDMA 传输完成(图 2 中⑥).

1.2 以太网 RDMA 网卡功能分析

标准以太网卡只具备物理层串并转换、数据链路层以太网帧头封装和解析、DMA 等功能[78], 无法支持 RDMA 的通信原语和传输方式等, 因此需要专用的以太网 RDMA 网卡在兼容标准以太网卡功能的基础上增加对 RDMA 功能的支持.

对于 RDMA 通信原语, 如应用程序下发的 WRITE 请求, 网卡需要 DMA 从用户空间中取出要写入远端的数据并在硬件中完成报文封装[36], 以实现零拷贝、内核旁路和 CPU 卸载的通信特性. 由于用户态应用程序使用虚拟地址, 网卡使用物理地址, 因此需要网卡进行虚实地址转换(根据由内核态提前配置好的内存转换表(memory translation table, MTT)等信息)以实现直接访问主机内存[21]. 此外, 对于 RC 传输方式, 需要由网卡通过确认应答和重传等机制保证传输的可靠性[41].

综合兼容以太网的特性和 RDMA 的功能需求, 本文将以太网 RDMA 网卡的基础功能定义为:

1)兼容标准以太网卡具备的功能;

2)支持 RDMA 报文封装与解析;

3)支持 RDMA 传输方式和通信原语;

4)支持虚实地址转换、DMA;

5)对于 RC 传输方式, 支持可靠传输.

很多工作在基础功能上进行优化, 如商用 RoCEv2 网卡[16]普遍支持 PFC[10], ECN[11]等功能构建无损网络环境以保证高带宽低延迟可靠传输, 但学术界提出的适应有损网络的 RoCEv2 网卡则通过选择重传等功能[7]消除对 PFC 的依赖. 此外还有多租户性能隔离[55]、安全性[62]等功能的增强, 详见第 2, 3 节.

2 以太网 RDMA 网卡架构

本节根据目前工业界和学术界的研究, 在明确以太网 RDMA 网卡功能的基础上, 总结其通用架构, 并介绍其报文处理逻辑、DMA、可靠传输等功能部件.

目前商用网卡内部通常是黑盒状态[14], 开源 RDMA 网卡实现较少[36,58]. 在工业界, Xilinx 于 2018 年推出的嵌入式 RoCEv2 网卡 ETRNIC(embedded target RDMA enabled NIC)[79]及后续的 ERNIC(embedded RDMA enabled NIC)[80]相关资料中描述了其内部部件及功能; 在学术界, 文献 [12, 14, 36, 39, 55, 57−59, 81−82] 等对 RoCEv2 网卡内部架构进行了介绍. 网卡供应商可以决定微体系结构级别的实现细节, 不同的网卡架构会由于面向的场景、支持的功能、针对性的优化等有差异, 但由于 IBTA(InfiniBand Trade Association)标准已经对 RDMA 分层架构、处理流程等进行了详细的定义[83], 因此 RDMA 网卡的基本架构是相似的[82].

根据 1.2 节定义的网卡基础功能和本节提到的代表性网卡架构, 以太网 RDMA 网卡通常包含 7 个部件:

1)用于连接主机端 CPU 和内存的 PCIe 部件;

2)用于连接网络端的 MAC 部件和 PHY 部件;

3)用于处理本端或远端不同通信原语对应的 RDMA 请求和响应, 进行报文封装和解析的报文处理逻辑部件;

4)用于直接访问主机内存的虚实地址转换、DMA 部件;

5)用于支持 RC 传输的可靠传输部件;

6)用于存储数据和元数据[82]信息的网卡 SRAM(static random access memory)缓存;

7)不同网卡架构中还可能有其他可选部件, 以在存储资源、可靠传输、应用相关方面进行优化.

据此, 以太网 RDMA 网卡架构如图 3 所示. 网卡

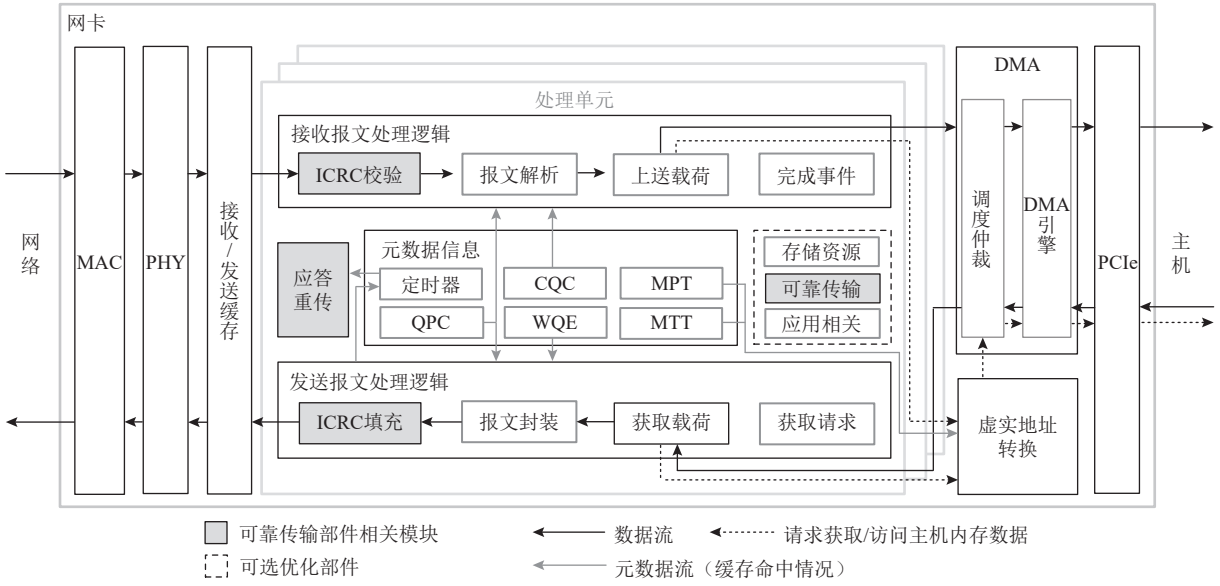


Fig. 3 Ethernet RDMA NIC module architecture (taking transmit-receive as an example)

图 3 以太网 RDMA 网卡模块架构图（以发送-接收为例）

通常具有多个处理单元(processing unit, PU)用以并行处理 QP[13,49], 每个处理单元包含报文处理逻辑、可靠传输、其他可选部件等. 以下依据代表性网卡架构对网卡各关键部件进行阐述.

1)PCIe 部件[12,14,36,39,57-58]、MAC 部件、PHY 部件[39,58]. 兼容标准以太网卡具备的连接主机端和网络端进行以太网帧头封装和解析功能.

2)报文处理逻辑. 该部件对应 1.2 节以太网 RDMA 网卡基础功能中对报文封装解析、不同传输方式和通信原语的支持. RDMA 网卡需处理 4 类报文, 分别为本端请求、本端响应、远端请求、远端响应, 对此存在依据发送接收或请求响应 2 种处理逻辑划分方式.

在基于发送接收的报文处理逻辑中, 需要发送的报文都经过发送报文处理逻辑进入网络, 包括本端请求、对远端请求的响应及重传报文等. 同理, 接收到的报文处理都在接收报文处理逻辑中完成, 包括对本端请求的响应、远端请求等. 该方式在收发路径上进行了较为清晰的处理逻辑分离, 2 条路径各自流水处理并通过专用功能部件[80]、共享数据结构信息[36]等方式进行交互以在收到远端请求后转入发送路径回送响应报文和实现可靠传输. 大多数网卡架构依据发送接收报文处理逻辑进行整体架构布局[12,36,39,55,59,79-82].

在基于请求响应的报文处理逻辑中, 则以发送本端请求和接收相应的响应(请求模块)、接收远端请求和发送相应的响应(响应模块)进行报文处理逻辑的划分. 由于不以收发进行逻辑划分, 因此需仲裁模块汇聚发送方向上的本端请求(来自请求模块)和远端响应(来自响应模块). 同理需分配模块将接收方向上的远端请求、本端响应分别送入相应模块. 中国科学院计算技术研究所先后提出的 csRNA[58], ScalaRNIC[57] 采用该处理逻辑. 相比于基于发送接收的报文处理逻辑, 该方式不需收发路径之间的交互模块, 但由于请求响应模块都需要收发报文而可能存在功能冗余.

无论划分方式如何, 相同类型报文的处理流程是相似的. 如对于本端发送 WRITE 请求报文, 网卡通常通过 Doorbell 机制(即驱动程序通知网卡 WQE 已放入 SQ, 通常通过更新 SQ 的写入指针值到网卡寄存器实现[14])获取 RDMA 传输请求[12,14,57-58,60,82], 通过解析 WQE 获取有效载荷描述符信息, 并发起 DMA 操作获取有效载荷, 之后依次进行 RETH(RDMA extended transport header), BTH(base transport header), UDP, IP 报文封装以及 ICRC(invariant cyclic redundancy check)填充[81], 并通过 MAC 部件和 PHY 部件发送至网络. 之后若网卡收到远端 ACK, 则构造 CQE 上送 CQ 告知应用程序传输完成[14,59], 否则进行报文重传(见可靠传输部分). 对于接收的远端 WRITE 请求报文, 则依次进行 ICRC 校验、各层报文头解析, 并根据可靠传输应答机制回送 ACK 或 NAK(见可靠传输部分). 若 WRITE 请求报文按序接收且校验无误, 则根据解析报文头获取的 DMA 长度等信息发起 DMA 操作并上送有效载荷.

3)虚实地址转换、DMA. 该部件对应 1.2 节以太

网 RDMA 网卡基础功能中对虚实地址转换的支持，并兼容标准以太网卡 DMA 功能. 在正式通信之前注册 MR 时会在内核态将虚实地址映射关系记录在 MTT 中. 当网卡进行内存访问时，由地址转换部件根据 MTT 等信息将应用程序下发的虚拟地址转换为物理地址后通过 DMA 部件[14,39,57-59,79-80,82] 获取主存数据.

在网卡上通常不会只有 1 个模块需要通过虚实地址转换和 DMA 部件与主机内存交互，如获取传输请求模块需从主存获取相应的 SQ WQE，获取和上送有效载荷模块需存取主存数据（如发送和接收 WRITE 请求时）等. 对此，通过将所有 DMA 操作请求集中汇聚到 DMA 部件中，由 DMA 调度仲裁模块对 DMA 请求进行有效检查、调度仲裁和缓冲后由 DMA 引擎执行 DMA 操作[14,39,57-59,82].

4）可靠传输. 该部件对应 1.2 节以太网 RDMA 网卡基础功能中对 RC 传输方式下可靠传输的支持，即需要保证数据的完整性和按序接收 2 方面. 对此，以太网 RDMA 网卡提供硬件卸载的可靠传输机制，即图 3 中 ICRC 校验和填充模块（数据完整性保证）、应答和重传模块（数据按序接收保证）. ICRC 校验通常在发送报文时进行 ICRC 计算和封装；在接收报文时重新计算和比对 ICRC 值，以检测报文在传输过程中是否发生了错误或被篡改.

应答模块[79-81] 负责可靠传输中的响应处理. 响应（接收请求）端应答模块会根据 BTH 报头（见图 2）中的报文序列号（packet sequence number, PSN）与网卡缓存的 ePSN（expected PSN）是否匹配判断报文是否按序接收[36]，并对 RETH 报头中的虚拟地址、DMA 长度、访问权限等进行校验，若报文按序接收且校验无误，则回送 ACK，否则丢弃数据包并回送 NAK 或静默丢弃. 响应端应对收到的每个请求包进行 ACK 或 NAK 确认. 对此，响应端通常会进行确认合并，为避免其无限期合并确认，请求端可以使用 BTH 报头中的 AckReq 位强制要求响应端回送确认（如对于每个请求 WQE 的唯一或最后 1 个数据包应设置该位）. 相应的[39,80]，请求端在收到 NAK 或超时未收到响应时触发重传，重传机制包括 go-back-0, go-back-N[84] 以及选择重传[7]. 重传机制可以在单独划分模块[14,79-80] 或嵌入在报文收发处理过程[36,39,58] 中实现.

ICRC 校验和应答重传保证了最基本的可靠传输，在大规模部署时，商用网卡普遍实现拥塞控制和流量控制以尽可能避免丢包重传，但部分学术界网卡实现仅做 RDMA 功能验证，未提及拥塞控制[36,55,57-59,81] 或流量控制[55,57-59,79,81]，因此本文将拥塞控制和流量控制归属为面向可靠传输的可选优化部件（详见 3.2 节）.

5）网卡 SRAM 缓存. 网卡功能部件在执行时往往需要使用网卡 SRAM 缓存. 网卡缓存信息主要包括数据和元数据信息. 数据缓存用于在收发方向上平衡网络传输带宽和网卡处理速率之间的性能差距. 元数据信息是支持 RDMA 传输所需的数据结构信息，通常包括直接访问主机内存所需的 MTT, MPT（memory protection table），以及 QP 相关的 QPC（QP context），WQE 等[57]. 其中，MTT 用于虚实地址转换，MPT 用于检查地址访问权限. QPC 则为每个 QP 维护 DMA 状态和连接状态等连接上下文信息，DMA 状态信息包括 DMA 地址、SQ 和 RQ 读写指针等，连接状态信息包括 ePSN, MSN（message sequence number）、重传定时器时间间隔、拥塞控制窗口或速率等. WQE 包括从 SQ 预取的发送 WQE 和从 RQ 预取的接收 WQE[14,36]. 各模块与元数据信息之间的交互关系如图 3 所示，以发送报文处理逻辑为例，获取请求模块根据缓存命中时预取的 WQE 进行请求解析，获取载荷模块根据请求解析信息和对应 QPC 通过虚实地址转换（根据 MTT 和 MPT）和 DMA 部件访问主存获取数据，报文封装模块也需根据对应 QPC 中的 PSN 等信息进行报文封装.

对于上述 QPC, MTT, WQE 等信息，是否使用网卡缓存[14]，在网卡中进行部分缓存[58] 或全部存储[36]，以及具体使用的网卡缓存级别[57] 等取决于设计目标. 通常情况下，以上信息会在主机内存中全部存储，同时在 RDMA 网卡中部分缓存. 但具体的设计选择需要根据网卡缓存容量和缓存命中率等进行权衡[82]，如 Wang 等人[14] 提出的 SRNIC 通过 SQ 调度等方式消除网卡中的 WQE 缓存；SRNIC 和 ETRNIC[79] 将 QPC 信息全部存储在网卡上以避免 QPC 缓存未命中带来的性能损失；中国科学院计算技术研究所提出的 ScalaRNIC[57] 则对网卡缓存进行了优先级划分并尽可能提升高优先级缓存信息的命中率（详见 3.1 节）.

6）可选部件. 在实现 RDMA 网卡基础功能之上，现有的研究通常面向存储资源、可靠传输、应用相关等方面进行优化，通过改进上述基础功能部件或引入新的部件以提升网卡性能或扩展网卡功能. 如在存储资源方面，对于网卡缓存资源，csRNA[58] 设计 QPC 连接调度器和管理模块实现非阻塞连接调度等. 对于主机内存资源，RIMA[67] 设计尾指针查找和更新模块对共享接收内存区域进行注册管理优化等. 在可靠传输方面，对于拥塞控制，HPCC[39] 在网卡上设计实现拥塞控制模块；对于流量控制，商用网卡通常支

持 PFC 功能; 对于重传机制, MELO[30] 在网卡中实现选择重传等. 在应用相关方面, 对于分布式存储系统, D-RDMA[85] 对 DMA 部件进行优化改进以提升数据库系统中数据混洗等操作性能; 对于数据中心应用, Optimize TX[55] 设计性能隔离模块以减少多租户环境流间的性能干扰等(详见第 3 节).

表 2 对以上在描述网卡架构和功能部件时参考的代表性网卡架构进行了统计对比. 不同网卡架构普遍支持 MAC、PHY、可靠传输、报文处理功能, 但在各功能部件的具体实现可能存在差异, 如对于接收报文模块处理逻辑, Xilinx ETRNIC[79] 由于其仅支持发送 READ/WRITE 请求(不支持接收), 在接收路径上主要完成响应报文的处理; 对于从发送报文处理逻辑中获取传输请求部分, Schelten 等人[81] 提出的 Efficient 以低资源开销实现 RoCEv2, 对于 READ 请求, 本端通过自定义预 RDMA 协议(基于 RC SEND)将待读数据大小和注册内存区域信息发送至远端, 由远端通过 Doorbell 机制下发 WRITE 请求实现本端 READ 请求功能. 此外, ETRNIC[79], ERNIC[80], Efficient[81] 作为嵌入式 RDMA 解决方案, 未通过 PCIe 连接标准主机 CPU 端, 而是通过 AXI 总线连接 MicroBlaze 处理器和 DDR 等完成双端建链和数据存取等操作, 因此不具备标准 RDMA 网卡中的 PCIe、虚实地址转换和 DMA 部件.

Table 2 Comparison of Typical RoCEv2 NIC Architectures

表 2 代表性 RoCEv2 网卡架构对比

发表或公布年份	会议/期刊/厂商	网卡	功能部件				通信原语				QP 数量
			报文处理逻辑	PCIe、DMA、虚实地址转换	可选部件		①	②	③	④	
					拥塞控制	流量控制					
2018	Xilinx	ETRNIC*[79]	发送/接收	×	√	×	√	√	√	×	127
2019	SIGCOMM	HPCC[39]		√	√	√	×	√	√	×	300
2020	EuroSys	StRoM[36]		√	—	√	√	√	√	√	16 000
2021	ICNP	StaR[59]		√	—	—	√	√	√	×	120
2021	Xilinx	ERNIC[80]		×	√	√	√	√	√	√	2 047
2022	TRETS	Efficient[81]		×	—	—	√	√	√	×	1
2022	ICCD	csRNA[58]	请求/响应	√	—	—	×	√	√	×	15 000
2023	ISCAS	ScalaRNIC**[57]		√	—	—	—	—	√	—	320
2023	NSDI	SRNIC[14]	发送/接收	√	√	√	√	√	√	×	10 000
2023	GLSVLSI	Optimize TX[55]		√	—	—	√	√	√	×	16

注: 以上网卡均支持 MAC、PHY、可靠传输基础功能. ①SEND/RECV; ②READ; ③WRITE; ④SEND with imm|SEND with invalidate|WRITE with imm. "—"表示原文献未说明. "*"表示 ETRNIC 只支持发送 READ 和 WRITE 请求(不支持接收). "**"表示 ScalaRNIC 高优先级 QP 数量支持 320 个, 一般优先级可支持大于 25 600 个.

3 以太网 RDMA 网卡优化

RDMA 允许直接访问远端内存以提供低 CPU 开销、低延迟和高带宽的网络传输, 可以满足当前数据中心应用及云存储等交互式系统的需求. 但在现有系统中部署以太网 RDMA 网卡仍然面临很多问题和挑战, 需要综合考虑存储容量、部署规模、网络环境、现有应用程序的兼容性和可移植性等多方面的因素. 目前已有较多的研究工作基于以太网 RDMA 网卡进行改进和优化, 以充分发挥其性能优势. 本节将主要从存储资源、可靠传输、应用相关 3 个方面分类阐述现有的研究如何针对以太网 RDMA 网卡进行性能优化或功能增强. 图 4 统计了本文以太网 RDMA 网卡优化方向的相关论文.

3.1 面向存储资源的优化

RDMA 硬件卸载和内核旁路等特性要求在网卡中缓存 QPC, MTT 等元数据信息以维护 QP 连接状态和进行直接内存访问等, 在主机内存中存储 QP, QPC, MTT 等并预先注册内存区域 MR, 以支持网卡接收应用程序下发的 RDMA 操作和进行直接内存访问等, 因此存储空间的使用策略和注册访问机制是否高效将直接影响系统的整体性能[107]. 本节主要介绍面向网卡缓存资源的连接可扩展性优化和面向主机内存资源的注册访问优化相关的研究工作.

3.1.1 网卡缓存资源

对于构建在多台机器之上的大规模分布式应用程序, 往往需要使用大量并发连接进行频繁的网络

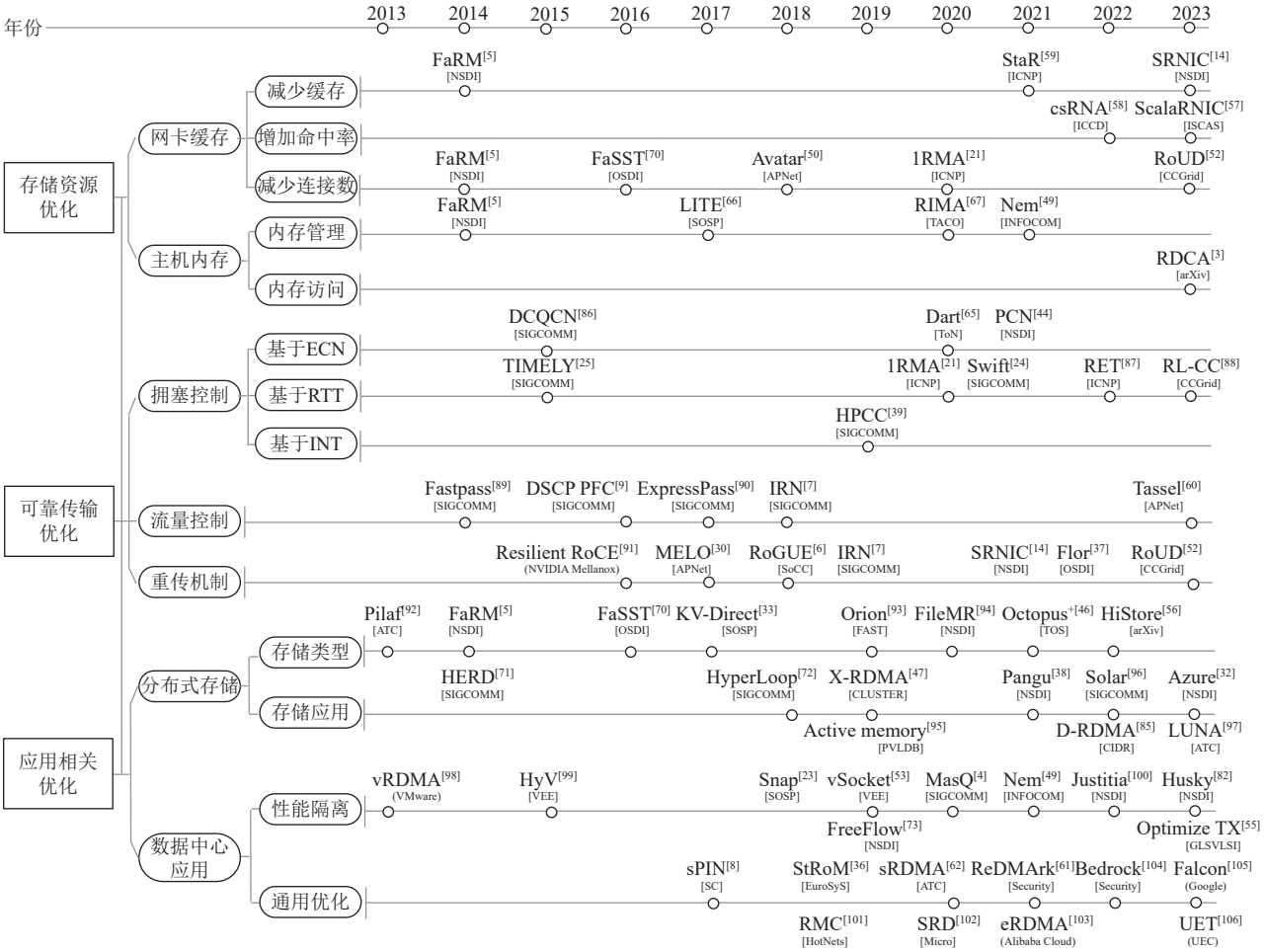


Fig. 4 Overall frame diagram of Ethernet RDMA NIC optimization

图 4 以太网 RDMA 网卡优化总体框架图

通信[59]. 然而由于网卡片上缓存容量非常有限(如 Mellanox ConnectX-5 的片上缓存容量为 2 MB)[14], 只能支持有限数量的 QP 连接, 当连接数量增加至一定值时系统性能显著下降, 即存在连接可扩展性问题. 商用网卡普遍面临该问题, 如 ETRNIC[79] 将所有的连接元数据信息都存储在网卡中, 因此最多只支持 127 条连接; NVIDIA Mellanox ConnectX 系列则在网卡上缓存部分元数据信息, 这使其可以支持更多的连接, 但当活跃的 QP 连接数超过网卡缓存大小, 会由于缓存未命中而需要在主机内存和网卡之间进行频繁的数据交换导致网卡性能急剧下降. 如对于 ConnectX-6 网卡, 当网卡需要维护的 QP 数量从 128 增长到 16 384 时, 吞吐量下降了 46%, 甚至在 QP 连接数量超过 256 时吞吐量就开始下降[58].

针对网卡有限缓存资源带来的连接可扩展性问题, 目前的研究工作主要从减少网卡缓存信息、增加网卡缓存命中率以及减少可靠连接数 3 个方面进行优化.

1)减少网卡缓存信息. 通过扩展报文头、上送主机内存、转移至远端等方式减少网卡存储的元数据信息.

Wang 等人[14] 在 NSDI 2023 提出的 SRNIC 针对网卡上的通用数据结构和选择重传相关的数据结构(沿用 IRN[7](见 3.2.2 节): 未完成请求表、重排缓冲区、位图)进行优化. 对于通用数据结构, SRNIC 采用全部存储(对 QPC)或部分缓存(对 MTT)的方式减少缓存未命中率, 并设计 SQ 调度器, 对预取但未使用的 WQE 通过丢弃而非缓存的策略实现无 WQE 缓存的 SQ 调度. 对于选择重传相关的数据结构, SRNIC 通过报文头扩展消除了未完成请求表和重排缓冲区; 通过上送主机内存消除了位图部分, 并将乱序报文的元数据信息送入软件重传模块进一步处理. SRNIC 使用 4.4 MB 缓存可支持 10 000 条 QP 连接, 但其片上缓存需求与 QP 连接数仍呈线性关系, 且上载软件重传的方式在丢包率较高时会引入较大 CPU 开销.

Wang 等人[59] 在 ICNP 2021 提出的 StaR 则直接

将服务器端(高并发连接)的连接状态信息完全转移到客户端(低并发连接),在服务器端中只基于客户端发送的报文中携带的连接状态信息进行报文处理和 DMA 等操作. 对于服务器端而言,StaR 解决了连接扩展性问题,但其局限在于只适用于通信双方并发连接数不对称的客户端服务器场景.

2)增加网卡缓存命中率. 通过高效的缓存管理、预取、连接调度等方式增加缓存命中率及减少缓存未命率对性能的影响.

中国科学院计算技术研究所在 ICCD 2022 提出的 csRNA[58] 和在 ISCAS 2023 提出的 ScalaRNIC[57] 采用增加网卡缓存命中率的思路. SRNIC[14] 在网卡中存储全部 QPC 的方式虽可以避免出现 QPC 缓存未命中的情况而影响网卡性能,但也使得网卡可支持连接数受网卡 QPC 存储容量的限制,csRNA 则采用在网卡中部分缓存 QPC 的方式以支持更多连接数(可达 15 000). 对于使用 QPC 缓存带来的头阻塞问题(缓存未命中会阻塞对后续 QPC 访问请求的处理),csRNA 设计 QPC 调度器和管理模块实现非阻塞连接调度. 先由调度器调度准备好的连接,再由管理模块维护 QPC 缓存和处理 QPC 访问请求,并在缓存未命中时一方面暂存该请求并发起 DMA 操作获取相应 QPC,另一方面直接处理后续命中的访问请求. 为避免语义错误,QPC 管理模块还解决不同 QPC 请求间的依赖关系以保证同一 QP 连接上的顺序处理. csRNA 可以在活跃连接数超过 15 000 的情况下实现接近峰值的消息速率,但 QPC 调度和管理模块处理逻辑复杂且引入了较多新的片上数据结构. 此外,csRNA 只针对 QPC 进行优化,未考虑其他连接上下文信息.

以上 csRNA 仅针对网卡中的 QPC 缓存进行非阻塞连接调度和缓存管理优化,ScalaRNIC[57] 则针对网卡中的 QPC, MTT, MPT, CQC(CQ context)四类缓存对其访问模式和管理机制进行优化. 在缓存访问模式方面,ScalaRNIC 通过 QPC 转发机制减少处理过程中访问 QPC 缓存的次数,通过 MTT/MPT/CQC 预取机制减少访问等待时间并在缓存未命中时实现延迟隐藏. 在缓存管理机制方面,ScalaRNIC 使用细粒度(8B 为单位)缓存分区,并将其划分为高优先级部分和普通部分,这 2 部分缓存占比和优先级软件可配置,同时不同类型缓存容量也可根据其访问特性(如时间和空间局部性特征)进行配置. 通过上述机制,ScalaRNIC 对于高优先级部分的连接上下文信息访问可保持近 100% 的缓存命中率,但对于普通部分的访存,其缓存命中率与缓存容量密切相关,当连接数超过容量限制,系统性能仍会下降.

3)减少可靠连接数. 通过使用 UD、连接共享等方式避免或减少建立可靠连接数,从而减少网卡需缓存的连接状态信息.

①使用 UD 而非 RC 传输方式可避免建立 1 对 1 的 QP 连接. Kalia 等人[70] 提出的 FaSST 使用 UD 提升网卡的可扩展性,并使用双边原语实现快速远程过程调用(remote procedure call, RPC),但并不保证可靠传输. He 等人[52] 提出的 RoUD 和 Singhvi 等人[21] 提出的 1RMA 则在使用 UD 的基础上将拥塞控制、重传和流控机制等可靠传输功能上载至软件传输层. UD 相比 RC 更简单且扩展性更好,但 UD 不支持单边原语,无法传输大于 4 KB 的数据,且在传输层实现可靠传输重新引入了 CPU 开销.

②使用动态连接传输(dynamically connected transport, DCT)[108]、扩展可靠连接(extended reliable connection, XRC)[109]、跨进程/线程的连接共享等方式减少连接数. DCT[108] 传输方式可以通过为请求建立动态临时连接的方式减少连接所需资源,但可能需要频繁地建立拆除连接,存在性能损失和复杂性问题. XRC[109] 传输方式允许请求端以单个连接与同一远端主机的所有进程通信,Dragojevic 等人[5] 在 NSDI 2014 提出的 FaRM 采用了类似的思路,通过远端多线程使用本端单连接、本端跨线程连接共享 2 种方式,将所需连接数从 $2\times m\times t^2$ 减至 $2\times m\times t/q$(其中 m 为机器数,t 为每台机器线程数,q 为共享连接的线程数),但需要根据不同集群规模分别确定最优 q 值,且当集群规模较大时该方式仍无法有效减少连接数.

FaRM 支持连接共享,但由于需要使用互斥锁来串行化对共享连接的访问,其同步 QP 共享模式在连接争用的情况下可能导致 80% 的性能损失[70]. 对此 Qiu 等人[50] 在 APNet 2018 提出的 Avatar(及后续进展[13,49])实现了跨应用程序的无锁 QP 共享,应用程序生成的 WR 并不直接下发至 QP,而是通过 Avatar 复用层中的 worker 线程(到同一远程节点的连接共享该 worker),根据 WR 将 WQE 下发到共享 QP,从而消除对 QP 的锁争用,实现了 1 024 条连接共享 1 个 QP. 连接共享的方式有效减少了对网卡缓存资源的使用,但由于需内核态参与,考虑到系统调用开销、用户空间和内核空间频繁的转换可能带来较大的性能损失.

此外,还可以通过分时复用来限制网卡需要同时处理的连接数. Chen 等人[41] 在 EuroSys 2019 提出的 ScaleRPC 对连接进行分组,通过 1 次只为有限数量(根据分组优先级、缓存大小、网卡和 CPU 处理能

力动态调整，在 20~70 个 QP 之间）连接提供服务实现网卡的分时复用，从而避免了网卡上的高并发连接数. 但该优化方式同样会增加 CPU 开销，甚至会破坏不同连接之间的依赖关系[59].

表 3 对上述网卡缓存资源优化的相关工作进行了总结. 以上扩展报文头的优化方式不兼容标准以太网 RDMA 报文格式，转移至远端的方式仅适用于客户端-服务器场景，通用性不高. 使用 UD 传输方式虽扩展性好，但不能发挥 RDMA 传输优势且 CPU 资源开销高. 表 3 所示的其他优化方式虽可缓解网卡连接可扩展性限制，但尚无方案完全解决该问题.

Table 3 Summary of NIC Cache Resource Optimization Schemes

表 3 网卡缓存资源优化方案总结

优化方向	优化方式	典型方案
减少缓存	扩展报文头	SRNIC[14]
	上送主机内存	
	转移至远端	StaR[59]
增加命中率	优化缓存管理	csRNA[58]
	优化连接调度	
	预取转发	ScalaRNIC[57]
减少连接数	使用 UD	FaSST[70], RoUD[52], 1RMA[21]
	连接共享	DCT[108], XRC[109], FaRM[5], Avatar[50]
	分时复用	ScaleRPC[41]

3.1.2 主机内存资源

RDMA 直接读写远端内存建立在应用程序预先在主机内存中注册内存区域（MR）的基础上. 此外，在内存中还需预先创建和维护 QP, CQ, QPC, MTT, MPT 等数据结构用于应用程序和网卡间交互、QP 连接状态维护、直接内存访问等[4]. 对于主机内存中 RDMA 相关的存储区域和各类数据结构，目前的研究工作主要从内存注册管理的易用性和高效性、内存访问争用 2 方面进行优化.

1）内存注册管理. 针对内存注册管理，RDMA 允许应用程序直接访问硬件网卡，即在硬件级别注册管理和保护内存资源，该低级抽象与建立在高级抽象之上的应用程序并不匹配，导致应用程序难以充分发挥 RDMA 性能优势，目前的研究工作主要通过添加软件中间层、修改内核驱动程序、改进硬件网卡架构等方式对内存注册管理的易用性和高效性等方面进行优化.

①添加软件中间层. Tsai 等人[66] 在 SOSP 2017 提出的 LITE 在 Linux 内核空间构建 LITE 层，将 MR, QP 等资源的管理和保护功能转移到 LITE，并为应用程序提供用于内存操作等的编程接口. 如对于 MR, LITE 仅为网卡注册 1 个全局 MR 并向应用程序提供虚拟内存区域 LMR，由 LITE 进行 LMR 的地址映射和权限检查从而隐藏 LMR 底层细节，为上层提供一个易使用的高级 RDMA 资源抽象. 针对添加中间层的系统调用和内存复制开销，LITE 采用部分功能上载、物理地址直接寻址等技术进行优化.

Wang 等人[49] 在 INFOCOM 2021 提出的 Nem 则对内存注册管理的高效性进行优化，通过在软件中间层中进行混合内存管理减少频繁 MR 注册和取消注册的开销. RDMA 对于大消息通常使用 memreg 注册 MR 后进行数据传输，对于小消息则提前注册一个空间较大的 MR，将要传递的消息 memcpy 复制至该 MR 以避免频繁注册 MR 阻塞程序运行. Nem 在此基础上，对大消息采用商用网卡（如 ConnectX-5）提供的按需分页（on demanding paging, ODP）功能消除 memreg 开销，对小消息采用流水 memcpy（消息分段与预取）机制减少传输延迟. LITE 和 Nem 可以更安全高效地管理和使用内存资源，但使用软件中间层不可避免存在性能损失并引入一定的 CPU 开销.

②修改内核驱动程序. FaRM[5] 针对在内存中注册 MR 较大时 MTT 条目数（以 4 KB 为单位进行映射）过多导致网卡频繁缓存未命中的问题，修改内核驱动程序实现以大页（物理连续且对齐的 2 GB 内存区域）为单位的 MR 注册管理，将 MTT 地址映射条目数从 50 多万条减少为 1 条，从而提升了注册 MR 空间大小的可扩展性，但使用大页也会导致内存占用增加以及物理内存碎片等问题[66].

③改进硬件网卡架构. 文献 [5,49,66] 的研究工作主要在软件层面进行 MR 等资源的注册管理优化，Xue 等人[67] 在 TACO 2020 提出的 RIMA 则针对共享接收队列（shared receive queue, SRQ），通过改进硬件网卡架构对 SRQ 对应共享接收内存区域的注册管理机制进行优化. SEND/RECV 双边操作需要响应端预先注册分配接收内存区域和下发 RQ WQE, SRQ 通过多 QP 共享 1 个 RQ 来减少需预分配的内存量. 但由于 SRQ 限制对不同操作请求均按照固定最大值（1MB）进行内存预分配，因此使用 SRQ 仍存在较显著的内存浪费和内存碎片等问题.

对此，RIMA 在 SRQ 基础上进行改进，如图 5 所示. 在网卡中，RIMA 根据接收数据包中携带的 QP 标识和消息大小进行尾指针查找（增加尾指针查找表）和更新（增加加法器）；在主机内存中，RIMA 将共享

接收内存区域在逻辑上作为一个连续区域的循环队列进行管理,通过头指针更新定位最早未完成请求对应数据位置并进行空间回收,通过尾指针更新定位当前接收数据包有效载荷应放入的内存空间,并在尾指针越过可用空间警戒线时提前注册分配内存空间以隐藏延迟. 在网卡侧,RIMA 通过较小的硬件开销实现了共享接收内存区域的灵活高效管理,但其仅针对使用双边原语时的接收端内存进行优化.

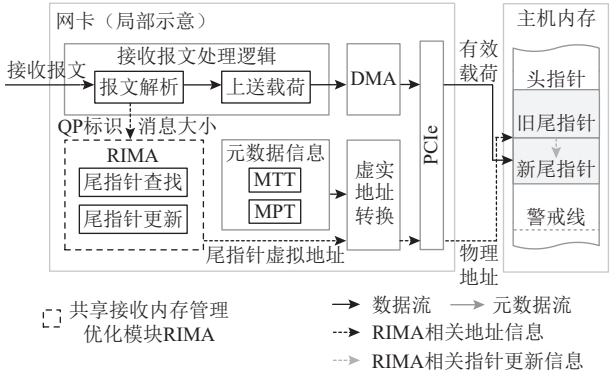


Fig. 5 Shared receive memory management optimization module: RIMA[67]

图 5 共享接收内存管理优化模块：RIMA[67]

2)内存访问争用. 针对内存访问争用,网卡和 CPU 在内存访问上的高争用可能导致网卡不能获得足够的存储器带宽资源,使得数据包在网卡缓冲区排队时溢出丢失,进而导致网卡在网络带宽尚未被充分利用时就触发拥塞控制机制,影响网络性能.

Li 等人[3]在 arXiv 2023 提出在最后一级缓存(last level cache, LLC)中为网卡保留一块区域,直接将 QP 等数据结构放入 LLC 中,实现远程直接缓存访问(remote direct cache access, RDCA). 通过 RDCA,网卡可以直接利用缓存的高带宽支持线速操作,同时避免了内存访问争用导致的网卡丢包问题. 但尽管 RDCA 使用了多种优化技术来减少对 LLC 的消耗,其 RDMA 资源仍需占用 20% 的 LLC,因此该方式的主机缓存开销较大,且其系统性能和可扩展性会受到缓存大小的限制.

表 4 对上述主机内存资源优化的相关工作进行了总结. 对于内存注册管理,LITE[66], Nem[49], FaRM[5]主要针对 MR 等资源的注册和管理,在软件层面简化用户编程和提升资源管理效率;RIMA[67]则针对使用 SRQ 存在的内存碎片等问题,在硬件层面改进网卡架构实现了 SRQ 对应共享接收内存区域灵活高效的注册管理. 对于内存访问争用,RDCA[3]采用了一种较为新颖的做法,不使用主机内存而是直接在主机缓存中管理 QP 等资源,但该方式显然对主机缓存的要求较高.

Table 4 Summary of Host Memory Resource Optimization Schemes

表 4 主机内存资源优化方案总结

优化方向	优化方式	典型方案
内存注册管理优化	添加软件中间层	LITE[66], Nem[49]
	修改内核驱动	FaRM[5]
	改进网卡架构	RIMA[67]
内存访问争用优化	上送主机缓存	RDCA[3]

3.2 面向可靠传输的优化

在 RDMA 应用于高性能计算(high performance computing, HPC)时,IB 网络中使用专用网卡和交换机并通过基于信用的流控策略构建无损网络环境. 当 RDMA 迁移到基于以太网构建的数据中心网络时,由于普通以太网是不可靠的传输媒介,在传输过程中可能会由于网络拥塞、服务质量(quality of service, QoS)策略等导致丢包. 为了实现类似于 IB 的高性能 RDMA 操作并满足 RC 传输方式下报文可靠传输的需求,在 RDMA 兼容以太网形式的协议实现中,iWARP 直接在其网卡中卸载 TCP/IP 实现可靠传输;对于 RoCE,现有的研究工作主要采用拥塞控制、流量控制并结合重传机制实现可靠传输以最大程度发挥 RDMA 的性能优势. 本节主要围绕 RoCE 介绍基于 ECN 等的拥塞控制、基于优先级等的流量控制,以及重传机制相关的 go-back-*N*、选择重传等工作.

3.2.1 拥塞控制

拥塞控制的主要目标是防止过多数据注入到网络中,避免路由器或链路过载导致的传输延迟增加、丢包率上升等问题,以最大限度保证网络稳定性和传输性能. 现有的拥塞控制有基于丢包或 ECN 等的拥塞感知方式,基于速率或窗口等控制方式,基于软件、FPGA、ASIC 等实现方式以及基于网卡或交换机等实现平台. 对于拥塞控制机制,更多介绍可见文献 [110−111],相比这些文献,本节主要基于不同的拥塞感知方式阐述网卡对拥塞控制的支持.

传统的 TCP 及改进的 TCP(如 BIC-TCP[112], TCP CUBIC[113])等使用丢包作为拥塞感知信号,在未出现丢包时不断增加拥塞窗口大小. 该方式可以尽可能提升网络利用率,但其并非针对数据中心网络环境专门设计,且使用丢包作为感知信号无法及时响应网络拥塞. 现有针对数据中心 RDMA 网络设计的拥塞控制中普遍采用的拥塞感知信号包括 ECN, RTT(round trip time)和 INT(inband network telemetry).

1)基于 ECN 的拥塞控制. 基于 ECN 的拥塞控制需要网卡和交换机均支持 ECN 功能. 在网络拥塞时，由交换机为数据包添加标记，接收端网卡需根据 ECN 标记构造和发送拥塞通知包(congestion notification packet, CNP)，并完成接收速率计算等工作，发送端网卡需根据收到的 CNP 包依据流级拥塞控制算法进行数据包级发送速率或拥塞窗口调整. Zhu 等人[86]在 SIGCOMM 2015 提出的 DCQCN 在发送端网卡根据接收数据包是否携带 ECN 标记分别采用类似 DCTCP[114] 和 QCN[115] 的发送速率调整策略，DCQCN 在 Mellanox ConnectX-3 Pro 硬件网卡中实现发送端和接收端的流级拥塞控制状态机. Xue 等人[65] 在 ToN 2020 提出的 Dart 则对拥塞采用分而治之的思想，针对接收端拥塞，通过发送速率直接分配实现快速收敛，针对网络拥塞，在空间局部网络拥塞时通过有序流偏转实现快速收敛，并在其他情况下回退到迭代收敛速度较慢的 DCQCN. 由于网卡固件修改需供应商支持，Dart 在软件应用层提供算法实现所需交互参数. Cheng 等人[44] 在 NSDI 2020 提出的 PCN 同样采用接收端驱动的发送速率调整以加快收敛，设计 NP-ECN 在接收端网卡进行拥塞识别以区分真正的拥塞流，并使用 DPDK(data plane development kit)网卡实现 PCN. 以上基于 ECN 的拥塞控制算法通常有较多(如 DCQCN 有 15 个)需调整的控制参数[39]，如速率降低因子、目标速率、CNP 发送周期等，因此其实现效果依赖于控制参数的选定.

2)基于 RTT 的拥塞控制. 以上基于 ECN 的拥塞控制依赖于交换机标记数据包，基于 RTT 的拥塞控制则直接根据端到端延迟对网络拥塞状态的反应进行数据包级发送速率或窗口的调整，不需要交换机支持，但如何获得精确测量的 RTT 值是关键. 目前，Mellanox, Chelsio 等厂商已在硬件网卡提供微秒级高精度定时器和数据包时间戳，Linux 内核也为应用程序提供了访问网卡硬件时间戳的接口. Mittal 等人[25] 在 SIGCOMM 2015 提出的 TIMELY 要求在接收端网卡中保证硬件级的 ACK 生成和在高优先级队列中发送，在发送端网卡中精确记录 RDMA 操作完成时间，在主机软件中，TIMELY 流级拥塞控制根据 RTT 梯度进行数据包发送速率的调整. 其后续在 SIGCOMM 2020 提出的工作 Swift[24] 同样使用网卡精确时间戳，在软件使用拥塞控制算法，Swift 将拥塞区分为网络拥塞和主机拥塞，对二者分别根据固定目标延迟基于 AIMD(additive increase multiplicative decrease)调整拥塞窗口. Tang 等人[87] 在 ICNP 2022 提出的 RET 在 Swift 使用固定目标延迟的基础上进行改进，结合 ECN 标记频率估计网络队列状态信息以获得准确的目标延迟值. Singhvi 等人[21] 提出的 1RMA 采用类似的实现思路，在硬件网卡进行延迟测量，在软件实现拥塞控制算法进行发送速率调整. 相比于 ECN，基于 RTT 的拥塞控制简单高效且可部署性强，但需要硬件精确测量 RTT，并在其不失效的情况下尽快调整发送速率或窗口.

此外，Fuhrer 等人[88] 基于 RTT 信息使用机器学习(machine learning, ML)进行拥塞控制，他们将基于 RL(reinforcement learning)拥塞控制算法[116] 的复杂神经网络提炼为决策树，从而将推理时间从 450 μs 减少为 0.9 μs，并部署在 ConnectX-6 Dx 网卡可编程拥塞控制引擎上实时运行. 该方式可以较好地适应负载动态变化的网络环境但实现复杂度和部署的难度较大.

3)基于 INT 的拥塞控制. 基于 INT 的拥塞控制可以根据获得的链路负载信息实现精确的流级拥塞控制. Li 等人[39] 在 SIGCOMM 2019 提出的 HPCC 中采用由交换机向数据包中插入 INT 元数据(时间戳、队列长度、链路带宽容量等)，接收端网卡将 INT 信息插入 ACK 响应包. 发送方网卡收到 ACK 后根据 INT 信息来细粒度感知网络状态并进行基于未确认字节数的发送窗口调整，从而可以快速提高流量并控制其低于链路容量从而获得高网络利用率，或适当降低流量以避免拥塞. HPCC 基于商用 FPGA 可编程网卡和 P4 可编程交换机实现了算法原型. 相比于 ECN 和 RTT，基于 INT 的 HPCC 只有 3 个需调整的参数并可以根据网络状况进行快速精确控制，但显然对交换机的依赖性很高且开销较大. 表 5 对上述以太网 RDMA 拥塞控制相关工作进行了总结.

Table 5 Summary of Congestion Control Optimization Schemes

表 5 拥塞控制优化方案总结

拥塞感知	感知控制网卡支持	典型方案	算法实现
ECN	CNP 报文封装解析、速率限制器	DCQCN[92]	基于硬件网卡
		Dart[65]	软件实现（协同硬件网卡 DCQCN）
		PCN[44]	基于 DPDK 网卡
RTT	数据包时间戳、速率限制器	TIMELY[25]	软件实现
		1RMA[21]	软件实现
		RL-CC[88]	基于可编程网卡
	数据包时间戳、软件拥塞窗口	Swift[24]	软件实现
		RET[87]	软件实现
INT	INT 报文封装解析、硬件拥塞窗口	HPCC[39]	基于可编程网卡

3.2.2 流量控制

拥塞控制的主要目的是使网络负载不超过网络的传送能力,而流量控制则关注接收方的接收和处理能力,主要目的是使发送方的发送速率不超过接收方的接收能力,以解决点对点或端对端的收发速率不匹配的问题[117]. 目前数据中心网络中主要采用基于优先级的流量控制,此外学术界相关研究中还有基于网络带宽延迟积的流量控制、基于信用的流量控制以及流量控制动态规划等.

1)基于优先级的流量控制(priority based flow control, PFC)[9]. 对于RC传输方式而言,拥塞控制可以提高网络传输质量,但并不能保证可靠传输,在突发流量、非连续流量等模式下仍易出现缓冲区溢出导致数据包丢失,进而影响RDMA传输性能,因此通常依赖PFC构建无损网络来保证可靠传输. PFC在网卡中实现基于优先级的队列管理,并在网络拥塞时通过向上游交换机或网卡发送PFC暂停帧来避免缓冲区溢出,从而提供点对点逐跳的基于优先级的流量控制.然而原始基于2层VLAN的PFC将数据包优先级携带在VLAN Tag的PCP(priority code point)中,在数据中心大规模部署(交换机运行在3层IP网络环境)时,跨越子网边界的通信无法保留VLAN中的PCP,其根本问题在于数据包优先级与2层VLAN不必要的耦合. 对此,Guo等人[9]在SIGCOMM 2016提出基于3层IP报文头中差分服务代码点(differentiated services code point, DSCP)字段的PFC机制. DSCP字段用于承载QoS优先级标签以便在网络拥塞时对不同优先级网络流量进行分类处理,基于DSCP的PFC只需将IP报文头中的DSCP值映射为PFC对应的优先级即可,该改进方式已经广泛得到商用网卡和交换机的支持.

PFC应用部署广泛且可构建无损网络,但当发送暂停帧时同一优先级队列的所有流量都会受到影响,而不区分具体的流,因此在大规模数据中心网络部署时存在头阻塞、拥塞扩散以及死锁等问题[10,25,29,118]. 目前的研究工作通常使用PFC进行快速响应来防止丢包,同时结合使用端到端流级拥塞控制,如DCQCN等以避免持续触发PFC,实现快速响应和长期稳定的平衡[29](见3.2.1节).

2)基于网络带宽延迟积的流量控制(bandwidth delay product based flow control, BDP-FC). 相比于RoCE依赖PFC构建无损网络,Mittal等人[7]在SIGCOMM 2018提出的IRN更认同iWARP的理念,即在网卡中有效处理丢包,而不是使网络无损. 但区别于iWARP在网卡中卸载整个TCP/IP协议栈,IRN以尽可能小的开销和网卡修改为出发点,在网卡中采用基于硬件的选择重传(见3.2.3节)和端到端数据包级的流控机制,即BDP-FC. BDP-FC由网卡依据BDP和QP MTU计算出限制流在网络中未确认数据包数量的固定上限值,并依据该值进行流量控制. 此外,为支持BDP-FC所需的逐包确认,网卡需额外增加对读响应报文的ACK/NAK. 相比于TCP字节流级别的流量控制,BDP-FC包级流量控制和固定上限值可以在减少网络中不必要的排队提升性能的同时减少网卡实现选择重传所需维护的状态开销;相比于PFC通过点到点逐跳反压以尽可能构建无损网络,BDP-FC采用的端到端包级流量控制不能保证无损(IRN认为设计好的丢失恢复机制比无损网络更重要(见3.2.3节)),但实现复杂度低,不需要交换机参与且不会带来死锁、拥塞扩散等问题.

3)基于信用的流量控制(credit-based flow control). 传统IB网络中基于信用的流量控制需要交换机逐跳参与支持,难以扩展到数据中心规模. Cho等人[90]在SIGCOMM 2017提出的ExpressPass则基于信用进行带宽分配,在交换机处只需速率限制功能. 接收端提前发送信用包,由交换机限制不同链路上的信用包数,使发送端根据接收的信用包发送数据,从而确保数据流不会超过链路容量. 对于信用包,由网卡构造以太网帧(84~92 B间随机)并引入发送时间随机抖动以避免信用包同步到达交换机,以便交换机执行公平的信用包丢弃. 此外,ExpressPass接收端网卡以信用包丢失率作为拥塞指标,并据此进行信用发送速率调整. ExpressPass在SoftNIC中实现其核心组件(基于信用的流量控制和速率限制器等),它有效解决了incast突发流问题,可以在发送数据前有效控制拥塞,但同时使用信用包会带来一定的带宽浪费(尤其是短流).

4)流量动态规划. Perry等人[89]在SIGCOMM 2014提出的Fastpass使用一种集中式控制架构进行流量动态规划,核心思想是对端节点的数据包发送进行集中式的严格控制来实现网络的分时复用以避免产生拥塞. Fastpass主要包括端节点和仲裁器2部分,端节点作为客户端将待发送数据包信息通过自定义FCP(Fastpass Control Protocol)发送至仲裁器,由仲裁器作为服务端确定其发送时间和传输路径并返回响应包至端节点. Fastpass可以用于支持数据中心RDMA网络的可靠传输且提供确定性的低传输延迟,但在大规模部署时集中式仲裁器的计算能力和网络

Table 6 Summary of Flow Control Optimization Schemes

表 6 流量控制优化方案总结

典型方案	核心思想	网卡支持
DSCP PFC[9]	点对点优先级流量控制	基于优先级的队列管理和流量控制、携带 DSCP 报文的封装解析、PFC 暂停帧构造响应.
BDP-FC[7]	端对端数据包级流量控制	基于 BDP 的固定窗口流量控制、增加对读响应的 ACK/NAK.
ExpressPass[90]	端对端信用流量控制	基于信用的流量控制、信用包封装解析、速率限制器.
Fastpass[89]	集中式流量控制	FCP 报文封装解析、时间同步.

吞吐量显然会成为瓶颈. 此外, 全网时间同步以及端节点和仲裁器之间的交互等会引入新的通信开销. 表 6 对上述以太网 RDMA 流量控制相关工作进行了总结.

在上述拥塞控制和流量控制中, 基于速率的控制都需使用网卡中的速率限制器来对每个流进行发送速率调节. 为满足硬件集成速率限制器的性能、准确性、可扩展性等需求, Wang 等人[14] 以 SRNIC 网卡架构为基础, 在 APNet 2023 中提出速率限制器 Tassel[60]. 如图 6 所示, Tassel 将经典的基于时间的加权共享算法 WF^{2}Q+扩展到以太网 RDMA 网卡场景下的速率限制. Tassel 通过事件综合调度活跃、准备就绪且有信用的有效 QP 以避免无效调度, 并设计从获取 WQE、计算数据包传输时间、将紧急数据包插入时间轴, 到参照全局计时器的系统时间对紧急数据包进行监控、排序和调度的整个速率限制算法处理流程. Tassel 可以提供数据包级精确可扩展的发送速率调节, 但增加了调度相关的数据结构, 且需要完成数据包传输时间计算、排序、调度等, 内部实现逻辑较为复杂.

3.2.3 重传机制

尽管结合 PFC, DCQCN 等机制已经可以改善网络传输环境, 实现一定程度的无损网络, 但设备或链路故障等原因导致的丢包、超时未送达等无可避免, 对此 RoCE 采用 go-back-0, go-back-N 或选择重传机制来配合流量控制和拥塞控制实现可靠传输.

1) go-back-0 重传方式要求接收方按序收到所有数据包, 否则将从对应消息的第 1 个数据包开始重传. 该重传机制对于网卡实现而言最为简单, 但同时在丢包时会占用大量网络带宽和系统资源. 即便丢包率极低(PFC 保证), RDMA 的性能也会急剧下降, 甚至在链路以线速充分利用的情况下(全部带宽都用于重传), 应用程序级别的有效吞吐量也为 0[9]. 因此, 该重传方式已普遍被 go-back-N 取代.

2) go-back-N 重传方式则根据 NAK 响应包携带的 PSN 从丢包处开始重传[1]. go-back-N 虽然同样存在带宽浪费(单个数据包丢失最多浪费 $RTT\times C$ 个字节(其中 C 为链路容量)), 但几乎和 go-back-0 一样简单, 对硬件网卡实现较为友好[9].

①硬件实现. go-back-N 已在 Mellanox, Intel 等商用网卡中得到广泛部署, 如 Mellanox 自 ConnectX-4 网卡起, 基于 go-back-N 实现弹性 RoCE(resilient RoCE)[91]. 早期经验显示, RoCE 需构造无损网络实现良好的传输性能, 因此数据中心通常依赖 PFC 进行流量控制. 但在大规模部署时 PFC 会带来死锁、管理复杂等一系列问题, 弹性 RoCE 使用 DCQCN 结合 go-back-N 以在有损以太网络上无缝运行(无 PFC), 简化了 RoCE 部署. 但由于 DCQCN 无法保证在所有动态流量模式下的可靠传输, 因此 Mellanox 仍建议启用 PFC.

②软硬件协同实现. Le 等人[6] 在 SoCC 2018 提出的 RoGUE 同样结合拥塞控制和 go-back-N/0 机制消除 RoCE 对 PFC 的依赖, 它在 verbs 编程接口之上软件实现拥塞控制相关逻辑, 同时利用硬件高精度时间戳和速率限制器等辅助软件实现基于 RTT 的拥塞控制. 对于重传, RoGUE 使用网卡 go-back-N 机制作为基础重传方案, 应用程序可设置网卡重传尝试次

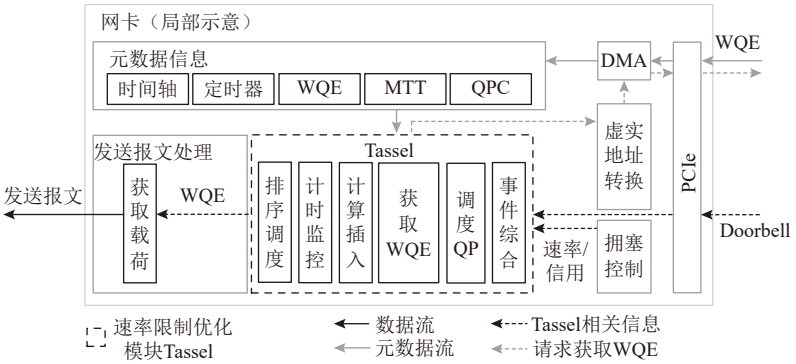


Fig. 6 Rate-limiting optimization module: Tassel[60]

图 6 速率限制优化模块：Tassel[60]

数(设置为 1),若硬件重传仍未成功,则由软件继续尝试重传. 由于硬件网卡并不告知软件已正确传输的数据包数量,因此软件采用 go-back-0 重传. 为隐藏软件重传延迟并保证应用程序,可及时下发新操作请求(重传期间 QP 置于错误状态无法接收新请求),RoGUE 设计影子 QP 机制,为每个活跃 QP 维护相应的影子 QP. 当活跃 QP 发生丢包重传时,RoGUE 会在影子 QP 中继续接收新请求(该 QP 转换为新的活跃 QP). 但由于无 PFC 流量控制,软件拥塞控制会引入处理延迟,同时使用的重传机制本身不够高效,因此必然影响 RDMA 传输性能.

3)选择重传.go-back-*N* 机制下丢包重传仍需占用较多带宽,如在原始链路吞吐量为 97 Gbps 的情况下,当丢包率超过 0.1% 时,Mellanox ConnectX-6 的有效吞吐量会迅速下降;当丢包率超出 1% 时,有效吞吐量会降至 25 Gbps[14]. 在不使用 PFC 的情况下,重传性能对于在有损以太网中部署 RDMA 更显重要[119]. 目前学术界已有相关研究通过选择重传增强网卡有效处理丢包的能力,并将其结合拥塞控制实现基于有损以太网的可靠传输.

① 硬件实现. Lu 等人[30]在 APNet 2017 提出的 MELO 较早尝试在 RoCE 中使用基于硬件的选择重传代替 go-back-*N* 机制,并通过数据和元数据分离存储、所有并发连接共享元数据结构 2 种优化技术尽可能减少选择重传引入的片上存储资源消耗. MELO 提升了网卡对有损网络的适应能力,但并未对 PFC 的必要性进行讨论,因此只在启用 PFC 的环境中实现. He 等人[52]提出的 RoUD 则结合选择重传和拥塞控制算法 LDCP 消除 RoCE 对 PFC 的依赖,并在可编程多核网卡 (uNIC) 和 ConnectX-6 网卡上进行了部署.

IRN[7]则结合选择重传和 BDP-FC 消除对 PFC 的依赖. 由于 BDP-FC(见 3.2.2 节)并不保证无损,IRN 通过选择重传适应有损网络环境. 为此,IRN 将原本请求端发送和接收采用的统一 PSN 空间划分为 sPSN 和 rPSN,分别用于跟踪发送的请求包和接收的响应包. 为支持乱序数据包处理,对于原本请求端只在消息首包中携带的信息,IRN 将其添加到每个数据包头;对于要求到达的数据包与其 WQE 匹配的应用,IRN 将 WQE 序列号携带在数据包头中进行标识. 此外,对于许多 RDMA 操作,请求端会在消息尾包中携带完成本次操作所需的元数据信息(如 SEND/WRITE with imm 的立即数据). 对于原本响应端收到尾包后回送 ACK 及上送 CQE(对 SEND/WRITE with imm 等)的操作,IRN 响应端使用位图记录乱序数据包到达情况,在尾包到达后预先生成 CQE 并存入主机内存,在所有数据包到达后回送 ACK 并将预先生成的 CQE 上送应用程序.

相比于 RoCE(带 PFC)和弹性 RoCE, IRN 可以在无 PFC 和显式拥塞控制的情况下达到较好的传输性能,但需要额外在网卡上维护选择重传相关的状态信息,如记录未收到确认数据包对应 PSN 等信息的未完成请求表(outstanding request table, ORT)以及记录数据包是否按序接收的位图(2-bitmap)等.

②软硬件协同实现. 该方面相关工作通过将部分功能上载至软件以实现性能与扩展性、灵活性之间的平衡. Wang 等人[14]在 NSDI 2023 提出的 SRNIC 在上述 IRN 基础之上,将数据包处理划分为快速处理路径和慢速处理路径,在硬件网卡中快速处理顺序数据包,并在软件中实现乱序数据包处理的选择重传逻辑,以一定的性能损失换取网卡的连接可扩展性(见 3.1.1 节). 此外,完全在硬件中实现拥塞控制和丢失重传等机制,不利于网络管理员进行自定义的硬件更改和灵活的拥塞控制算法部署, Li 等人[37]在 OSDI 2023 提出的 Flor 采用软硬件协同的实现方式,在不修改现有网卡的基础上将无 PFC 依赖的拥塞控制和选择重传上载软件实现. 软硬件协同的方式以较低的实现成本消除了对 PFC 的依赖,但同时软件实现的拥塞控制和重传以及软硬件之间的切换和交互也会引入一定的存储资源和 CPU 处理开销. 表 7 对上述重传机制相关优化工作进行了总结.

Table 7 Summary of Retransmission Mechanism Optimization Schemes

表 7 重传机制优化方案总结

重传机制	实现形式	优化方案	优化方式
go-back-*N*	硬件	弹性 RoCE[91]	结合 DCQCN 适应有损
	软硬协同	RoGUE[6]	结合软件实现 TIMELY 适应有损
选择重传	硬件	MELO[30]	优化选择重传网卡存储资源消耗
		RoUD[52]	结合 LDCP 适应有损
		IRN[7]	结合 BDP-FC 适应有损
	软硬协同	SRNIC[14]	基于 IRN 软件实现选择重传适应有损
		Flor[37]	软件实现选择重传和拥塞控制适应有损

3.3 面向应用相关的优化

面对数据中心多样化的部署需求和应用场景,如分布式存储、多租户性能隔离等,构建高性能的 RDMA 系统需要仔细选择传输模式和通信原语以充分发挥其性能优势,并调整软硬件结构进行适配[107]. 在软件层面结合 RDMA 硬件特性通过应用和扩展

RDMA 原语、添加软件中间层、虚拟化、加密验证等方式适配不同部署需求和场景特点. 在硬件层面通过改进网卡架构和添加功能模块来增强网卡功能. 本节主要介绍在分布式存储系统、数据中心应用中基于以太网 RDMA 网卡的优化研究工作.

3.3.1 分布式存储系统

作为数据中心网络中为上层应用提供存储服务的基础设施, 分布式存储系统将数据分散存储在多个设备上, 并通过网络连接进行管理和访问. 对于分布式存储, 更多介绍可见文献 [45,107]. 相比这些文献, 本节侧重于从分布式存储类型和应用 2 个方面阐述以太网 RDMA 网卡面向分布式存储系统的优化工作.

1)分布式存储类型. 分布式存储按照存储类型可分为块存储、对象存储、文件存储, 这 3 者以不同的方式和层次保存、整理和呈现数据. 其中块存储将数据拆分成块并单独存储各个块, 对象存储将管理数据链接至关联的元数据, 文件存储则采用文件和文件夹的层次结构存储数据[120].

在块存储方面, 陈思新[121] 在 2022 年提出的分布式块存储系统 Flame 结合 RDMA 技术从通信模型、传输策略和 RDMA 资源管理 3 个方面进行优化, 采用贯穿式通信模型、读写转换、动态技术轮询获取 CQE 等方式在块存储系统中充分发挥 RDMA 性能优势. 但由于块存储涉及底层裸磁盘, 需要考虑数据完整性、并发访问等问题, 目前研究较少, 以下主要介绍对象存储和文件存储的相关工作.

①对象存储. 对象存储将数据和元数据存储在一个对象中, 并通过标识符唯一标识, 即基于键值对进行存储以及基于键值操作进行访问和修改. 对象存储可用于存储机器学习模型参数、图计算的节点和边等非结构化数据[33]. 传统基于 TCP/IP 的对象存储键值操作难以满足数据中心应用通信带宽需求[5], 对此 RDMA 被广泛应用以提升其传输性能. 现有的键值操作相关研究主要在软件层面通过应用和扩展 RDMA 原语的方式进行性能优化.

其一, 使用现有网卡提供的 RDMA 原语执行键值操作. 应用不同通信原语(单边、双边、混合单双边)RDMA 的传输性能不同, Mitchell 等人[92] 在 ATC 2013 提出的 Pilaf 使用 READ 优化 GET 操作, 使用 SEND/RECV 优化 PUT 操作. FaRM[5] 同样使用 READ 完成 GET 操作, 并将集群中机器内存公开组成共享地址空间从而为上层应用提供简单的编程模型. Han 等人[56] 提出的 HiStore 则分别对 GET 和 SCAN 操作应用单边和双边原语以优化读写性能. Kalia 等人[71] 提出的 HERD 同样混合使用单双边原语, HERD 认为当单边操作需要多个 RTT 时, 只需单 RTT 的双边操作更具性能和可扩展性优势. 在 HERD 中客户端通过 UC WRITE 向服务器发送 GET 或 PUT 请求, 服务器在执行完请求后使用 UD Send 向客户端发送回复. 其后续提出的 FaSST[70] 则认为单边原语会增加软件复杂性, 且在网卡上存储连接状态会限制可扩展性, 因此完全使用 UD SEND/RECV 进行传输, 并通过集成协程、Doorbell 批处理等方式进行延迟隐藏和性能优化.

其二, 扩展现有网卡提供的 RDMA 原语以适配键值操作. 单边原语可以实现更好的 GET 性能, 但需要更高的通信和同步开销, 相应 PUT 性能会降低; 使用双边原语具有更好的可扩展性但同时会受到服务器 CPU 性能限制. Li 等人[33] 在 SOSP 2017 提出的 KV-Direct 使用基于 FPGA 的可编程网卡将 READ 和 WRITE 操作扩展为键值操作和矢量操作, 并支持用户定义的更新函数. KV-Direct 还通过设计的乱序执行引擎等减少和隐藏网卡的 PCIe 访问延迟, 实现了比基于 CPU 的分布式 KVS 高 3 倍的能效.

②文件存储. 文件存储将数据存储到文件, 将文件组织到文件夹, 将文件夹组织到目录和子目录的层次结构中, 并使用分布式文件系统对文件进行访问、管理和权限控制[122]. 非易失主存(non volatile main memory, NVMM)可以在持久存储数据的同时提供与动态随机存取存储(dynamic random access memory, DRAM)相当的高速读写性能, NVMM 的出现将分布式文件系统数据路径上的性能瓶颈从访存转移至网络堆栈处理开销和软件延迟[93], 对此, 现有的相关研究主要在软件层面通过应用 RDMA 原语和添加软件中间层的方式进行性能优化.

其一, 根据现有网卡提供的 RDMA 原语重新设计文件系统. Lu 等人[46] 提出的 Octopus+将 RDMA 和 NVMM 紧耦合以设计文件系统, 并分别针对数据和元数据访问进行优化, 对数据操作其直接访问共享持久内存, 对元数据则引入自识别的 RPC 并结合 WRITE 和 ATOMIC 操作等实现 RDMA 和文件系统的有效集成. Yang 等人[93] 提出的 Orion 将文件系统的功能和网络操作组合到 1 个层中, 同样将 RDMA 深度集成到基于 NVMM 的文件系统中以加速其元数据和数据访问.

其二, 添加软件中间层适配文件系统的内存管理特点. NVMM 文件系统以文件进行内存管理, RDMA 网卡则以内存区域(MR)进行远程访问, 二者

不同的内存抽象层次需要较大的转换开销, 对此, 上述 Orion[93] 在 NSDI 2020 的后续工作 FileMR[94] 的基础上创建一个结合 MR 和文件的抽象层, 客户端可通过 RDMA 直接访问文件, 并通过文件偏移量进行内容寻址. 在网卡缓存有限的情况下, NVMM 较大的容量需要网卡频繁与主存通信以获取虚实地址转换信息, 对此, FileMR 采用基于范围的转换, 将可变大小的虚拟地址映射为物理地址, 从而减少 MTT 所需空间和注册时间.

2)分布式存储应用. 基于分布式存储的应用, 如数据库系统和云存储系统等使用分布式存储作为底层的存储支持, 并向上层用户提供高效、可靠的数据管理或存储服务. 以下主要介绍基于分布式存储的数据库系统、云存储系统相关工作.

①基于分布式存储的数据库系统. 分布式数据库系统主要用于存储和管理结构化数据, 并提供数据查询、事务处理等功能. 面向数据库系统, 现有的研究工作通过应用和扩展 RDMA 原语、添加软件中间层、改进硬件网卡架构等方式保证数据库系统的高可用性、高性能和可维护性.

其一, 应用和扩展现有网卡提供的 RDMA 原语以提升数据库的可用性. 分布式数据库系统通常通过数据复制以适当的冗余备份来容忍单机故障, 实现高可用性. 传统的数据复制算法以处理冗余为代价尽可能减少副本之间的网络通信, 然而基于 RDMA 的高性能网络将瓶颈转移至 CPU, 对此 Zamanian 等人[95] 提出的 Active Memory 使用 WRITE 操作直接对远端备份节点进行更新, 从而以增加一定网络流量为代价消除了复制过程中备份节点进行冗余计算的 CPU 开销. Kim 等人[72] 提出的 HyperLoop 则扩展 RDMA 原语(gWRITE 等), 由网卡直接访问和修改数据库内容, 从而实现旁路 CPU 的复制事务处理并提供可预测的低处理延迟.

其二, 添加软件中间层以实现数据库系统的高可维护性. 对于大规模生产集群的数据库系统, 提升系统的开发和维护效率十分必要. 对此, Ma 等人[47] 在 CLUSTER 2019 中开发用户空间中间层 X-RDMA, 通过提供高度抽象的数据结构(context, channel, msg)和 API 来简化 RDMA 编程, 并通过 seq ack 窗口和 keep-Alive 选项扩展 RDMA 协议以提升发送端对接收端处理进度和活动状态的感知. 此外, X-RDMA 通过流量控制、资源管理、性能分析和调试等组件, 进一步提升系统的健壮性、可扩展性以及可维护性.

其三, 改进硬件网卡架构以提升数据库系统传输性能. 数据库系统中常见的数据混洗操作等可能产生大量非连续数据, 需要频繁发起 DMA 操作, 占用的通信和处理开销较大. Ryser 等人[85] 在 CIDR 2022 提出的 D-RDMA 通过改进硬件网卡解决该问题, 其核心思路为让网卡接管 DMA 调度, 从而允许主机应用程序更紧凑地描述要传输的区域. 如图 7 所示, D-RDMA 允许应用程序使用非连续区域(non-contiguous regions, NCR)向网卡发起请求, NCR 可包含比单个消息和数据块更大的区域, 并同时包含数据和数据间隙的信息. 对于 NCR 请求, 由优化器决定 DMA 传输范围, 由分段存储器暂存 NCR 数据, 由装配器根据优化器提供的掩码信息组装有效载荷. D-RDMA 可将网络吞吐量从 18 Gbps 提高到 98 Gbps (基于 100 Gbps ConnectX-5), 并保持接近于零的 CPU 开销, 但对网卡与上层应用的交互方式改动较大, 不易兼容现有 RDMA 应用.

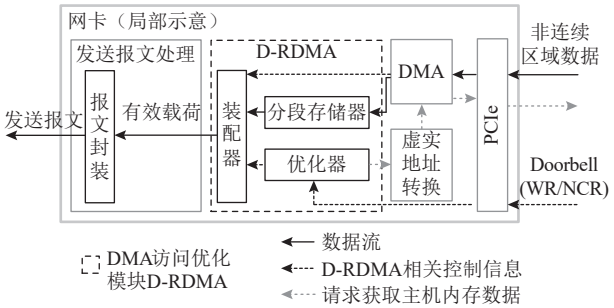


Fig. 7 DMA access optimization module: D-RDMA[85]

图 7 DMA 访问优化模块: D-RDMA[85]

②基于分布式存储的云存储系统. 云存储通过互联网对存储资源进行抽象化、汇集和共享[120]. 目前领先的云存储系统, 如阿里巴巴公司的 Pangu[38]、微软公司的 Azure[32,123] 均采用“计算存储分解”的理念, 并可根据使用场景提供块存储、对象存储、文件存储等不同的分布式存储类型.

Pangu 在 NSDI 2021[38] 中指出在存储后端(存储集群内)部署 RoCEv2 以提供微秒级访问延迟和近 100 Gbps 吞吐量的网络服务, 通过应用层共享链路的模式减少 QP 数量, 并通过 RoCEv2 和 TCP 混合部署保持系统的高可用性. 在存储前端(计算和存储集群之间), 使用用户空间 TCP 软件堆栈 LUNA[97] 而非 RDMA 以在降低延迟和 CPU 开销的同时保证可扩展性和互操作性, 提供跨数据中心通信的支持, 并进一步在 SIGCOMM 2022 中提出可靠 UDP 堆栈 Solar[96] 以实现网络存储的联合硬件加速. 相比于 RDMA 只卸载网络堆栈, Solar 将网络数据包和存储数据块的概念统一, 使弹性块存储(elastic block storage, EBS)的

数据路径(包括存储虚拟化部分)可完全卸载到硬件.

Pangu 在存储后端应用 RDMA, Azure[32] 在 NSDI 2023 中则提出同时在存储前端和后端应用 RDMA, 并同样提供 RoCEv2 和 TCP 混合部署. 在存储前端, RoCEv2 可实现从单个数据中心扩展到跨数据中心的数据传输. 对此, 微软公司开发新的基于 RDMA 的存储协议分别用于存储集群内(sU-RDMA)及集群间通信(sK-RDMA), 并针对集群间异构网卡的互操作性问题, 更新网卡固件统一 DCQCN 的行为, 以避免 DCQCN 实现差异(生成和响应拥塞通知包(CNP)的粒度等)引起的性能损失. 表 8 对上述分布式存储相关优化工作进行了总结.

Table 8 Summary of Distributed Storage Optimization Schemes

表 8 分布式存储优化方案总结

优化方向		优化方式	典型方案
存储类型	对象存储	重构键值操作	Pilaf[92], FaRM[5], HiStore[56], HERD[71], FaSST[70]
		扩展 RDMA	KV-Direct[33]
	文件存储	重构文件系统	Octopus+[46], Orion[93]
		添加软件中间层	FileMR[94]
存储应用	数据库系统	应用扩展 RDMA	Active Memory[95], HyperLoop[72]
		添加软件中间层	X-RDMA[47]
		改进硬件网卡	D-RDMA[85]
	云存储系统	重构存储系统	Pangu[38], LUNA[97], Solar[96], Azure[32]

3.3.2 数据中心应用

面对数据中心中承载的大数据处理[90,124-125]、机器学习[126-127]、搜索、流媒体服务等各种应用, 完全使用原生 RDMA 构建多应用程序共享的数据中心网络难以满足其性能、安全性等方面的需求. 本节主要阐述以太网 RDMA 网卡面向数据中心应用需求的多租户性能隔离、安全性及可编程性相关的优化工作.

1)多租户性能隔离. 使用多租户技术可使多个延迟敏感(latency sensitive, LS)或带宽敏感(bandwidth sensitive, BS)应用程序共享单个硬件网卡, 提升网卡资源利用率. 但 RDMA 在将数据平面卸载到网卡以减少 CPU 开销的同时, 也无法使用基于 CPU 的资源分配和流仲裁机制等. 在多租户环境下, 这会带来不可预测的消息尾部延迟、不公平的带宽共享等性能干扰问题. Zhang 等人[77] 在 KBNets 2017 中基于 ConnectX-4 网卡对 LS 流和 BS 流共享网卡进行了性能测试, 发现 BS 流间、LS 流间、BS 与 LS 流间均存在性能干扰. Kong 等人[82] 在 NSDI 2023 中构建的 Husky 则综合网卡带宽、处理单元、缓存以及 PCIe 带宽 4 类网卡资源对多租户性能隔离解决方案进行测试评估, 并指出目前的商用 RDMA 网卡均无法提供性能隔离. 现有学术界的相关工作主要通过使用虚拟化技术、添加软件中间层、改进硬件网卡架构 3 种方式实现一定程度的性能隔离.

①使用虚拟化技术. 虚拟化技术将单个硬件网卡资源虚拟化为多个网卡资源供虚拟机或容器使用. 相比于在裸机环境下执行, 虚拟化可以为不同租户的应用提供性能隔离保护, 并通过硬件复用实现更高的网卡资源利用率.

对于虚拟机的虚拟化, 目前相关研究主要集中在软件半虚拟化、硬件虚拟化和混合虚拟化. 在软件半虚拟化方面, VMware 提出的 vRDMA[98] 通过客户端前端驱动和主机端后端驱动配合实现半虚拟化. Wang 等人[53] 提出的 vSocket 针对云环境中基于 Socket 的应用程序, 通过内核连接重用和 RDMA 连接的虚拟化而非硬件网卡虚拟化, 消除了半虚拟化 I/O 方案的前后端数据复制开销. 在硬件虚拟化方面, 由 Intel 提出的单根 I/O 虚拟化(single root I/O virtualization, SR-IOV)[128] 已经得到了广泛部署和应用, 在 SR-IOV 中虚拟机可以直接访问硬件网卡以提供接近裸机的性能, 但由于需要硬件和平台支持, 灵活性和可维护性不足. 在混合虚拟化方面, Pfefferle 等人[99] 提出的 HyV 基于 RDMA 控制和数据操作路径分离的概念, 对于控制操作由虚拟机管理程序参与进行虚拟化, 对于数据操作则完全绕过虚拟机在网卡中执行. 相比于 SR-IOV, HyV 在高性能数据操作和灵活资源管理之间取得了一定的平衡. He 等人[4] 提出的 MasQ 延续了 HyV 的混合虚拟化思路, 在虚拟专用云(virtual private cloud, VPC)中部署了虚拟化 RDMA, 并在控制路径上进行租户隔离及安全等方面的增强.

对于容器的虚拟化, HyV 等虚拟化技术虽可用于容器, 但仍存在一定限制(如混合虚拟化不能支持灵活的数据平面策略等). Kim 等人[73] 提出的 FreeFlow 使用服务器上运行的软件虚拟交换机对容器间通信的控制路径和数据路径进行管理, 实现了专用于容器化云的半虚拟化 RDMA, 并可在一定 CPU 开销下实现与原生 RDMA 相当的吞吐量和延迟.

②添加软件中间层. Marty 等人[23] 在 SOSP 2019 提出的 Snap 在应用程序和网卡之间添加软件中间层, 并根据不同处理需求(调度延迟、性能隔离、CPU 效率)提供专用、共享和最少处理单元 3 种调度模式从而实现不同强度的性能隔离, 如在使用专用处理单

元时, 应用程序不与其他应用程序共享 CPU 和调度决策, 虽然可以获得更强的性能隔离, 但同时也会带来更高的 CPU 和内存成本. Zhang 等人[100] 在 NSDI 2022 提出的 Justitia 则通过软件中间层实现租户级连接管理. Justitia 添加整形器通过连接拆分和消息整形动态调整消息大小、速率等, 并根据接收端反馈信息调整资源分配, 实现应用程序级的连接管理和可控的带宽共享. 此外, Justitia 使用守护进程依据到不同接收方的参考流延迟测量值检测资源争用, 进而根据尾部延迟目标为延迟敏感流提供可预测延迟. Nem[49] 同样在软件中间层实现性能隔离, Nem 通过硬件 QoS 将 QP 分为 8 个优先级来隔离 LS 流和 BS 流. 对于不同连接共享的同一 QP, 将 BS 流工作请求分段成几个消息长度相对较小(4KB)的工作请求来减轻其对 LS 流工作请求的影响.

③改进硬件网卡架构. 同样针对性能隔离, Liao 等人[55] 在 GLSVLSI 2023 提出的 Optimize TX 则直接改进接收端硬件网卡架构, 如图 8 所示, 对于 LS 流和 BS 流间的性能隔离, Optimize TX 对其进行分离缓存和 2 倍加权轮询调度(2-weighted round-robin, 2-WRR), 并通过 BS 流切片执行(在 1 次数据传输后将未完成处理的 WQE 信息写回切片状态等待再次调度)解决 BS 流对 LS 流的头阻塞问题. 针对 BS 流间的性能隔离, Optimize TX 使用切片状态隔离背压机制避免单 BS 流 WQE 缓存队列满信号与状态写入模块允许读信号之间的耦合, 并根据工作负载特性对不同的 BS 流进行自适应 WRR 调度. Optimize TX 在增加一定调度处理逻辑和缓存开销的情况下, 以较低的 CPU 开销实现了较优的性能隔离. 表 9 对上述性能隔离相关优化工作进行了总结.

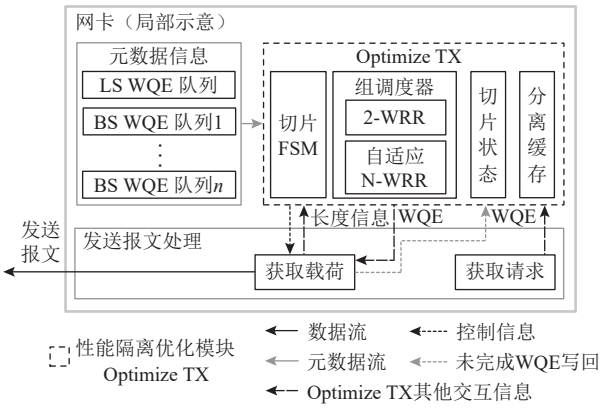


Fig. 8 Performance isolation optimization module: Optimize TX[55]

图 8 性能隔离优化模块：Optimize TX[55]

Table 9 Summary of Performance Isolation Optimization Schemes

表 9 性能隔离优化方案总结

优化方式		典型方案	方案特点
虚拟化	软件半虚拟化	vRDMA[98], vSocket[53]	客户端前端驱动和主机端后端驱动配合
	硬件虚拟化	SR-IOV[128]	可提供接近裸机的性能
	混合虚拟化	HyV[99], MasQ[4]	仅 RDMA 控制操作参与虚拟化
	容器虚拟化	FreeFlow[73]	可满足容器化云对可移植性迁移等的要求
添加软件中间层		Snap[23], Justitia[100], Nem[49]	可实现较为灵活的流调度和资源分配
改进硬件网卡架构		Optimize TX[55]	较好解决 BS 流对 LS 流的头阻塞问题

2)安全性. RDMA 体系结构的设计主要关注性能而非安全性, 其在旁路 CPU 的情况下直接将内存暴露给远端, 并仅根据收到的数据包是否与其状态信息匹配(如 MPT, MR, rkey 等)来识别恶意内存访问[61]. 网络窃听者可以轻易获取应用程序的敏感信息甚至篡改数据包内容, 这会损害应用程序的机密性和完整性. 目前的相关工作对 RDMA 的安全漏洞进行了探索, 并通过加密验证等方式增强其安全性.

Rothenberger 等人[61] 在 ReDMArk 中通过对各种威胁模型下 RDMA 安全问题的研究, 发现原生 RDMA 缺乏安全的身份验证机制, 且现有网卡(Mellanox, Broadcom 等)存在多个漏洞, 如攻击者可以发起拒绝服务攻击、注入欺骗性 RDMA 数据包等. 针对安全的身份验证机制, 其后续工作 sRDMA[62] 为 RDMA 设计 SRC(secure reliable connection) QP, 该 QP 使用对称密码(基于 OpenSSL)进行加密和身份验证, 并通过保护域(protection domain, PD)级密钥取代 QP 级密钥来减少引入的网卡缓存开销. 针对 ReDMArk 发现的漏洞, Xing 等人[104] 提出的 Bedrock 在可编程交换机(也可部署在可编程网卡上)实现了一套互补的防御措施.

3)可编程性. RDMA 可以提供高带宽和低延迟, 但其仅完成从内存中存取数据的搬移工作, 不对数据有效载荷进行处理, 也并不为数据中心应用程序提供自定义处理能力. 目前的相关工作主要通过改进硬件网卡架构或使用智能网卡的方式增强其可编程能力.

Hoefler 等人[8] 提出的 sPIN 基于网内计算的思想, 在 RDMA 的基础上更进一步, 其采用基于并行多核 HPU(handler processing units)形式的网卡架构, 直接将用户定义的简单的数据包处理操作卸载到网卡上来增强其可编程能力. Sidler 等人[36] 提出的 StRoM 在 sPIN 数据包级处理的基础上, 为用户提供更高级别

的数据抽象(关注有效载荷)和编程抽象(从硬件描述语言提升到 C/C++). 在网卡架构上, StRoM 在 RoCEv2 协议栈和 DMA 引擎之间的数据路径上放置 Kernel 可编程组件以支持从网卡直接执行遍历远程数据结构等存储访问和处理操作. sPIN 和 StRoM 直接部署在网卡数据路径上(on-path), Amaro 等人[101]在 HotNets 2020 提出的 RMC 则基于智能网卡设计 off-path 的远程内存操作. RMC 允许应用程序自定义扩展的单边操作, 以满足实际应用中复杂的访问模式(如扫描远程数据结构), 但 RMC 尚未实现, 文献[101]仅对其部署挑战和可行性进行了论证.

除上述面向数据中心应用的通用性优化, 针对 AI/HPC 等新兴数据中心工作负载, 阿里云公司在 2021 年提出的弹性 RDMA(elastice RDMA, eRDMA)[103]基于 iWARP 支持 AI/HPC 应用程序, 并通过自研拥塞控制算法实现有损网络中的良好性能, 通过大规模 RDMA 加速显著提高分布式计算通信效率. 谷歌公司在 2023 年公布的 Falcon[105]同样针对 AI/HPC 等场景, 在 UDP 和 RDMA 协议间增加 Falcon 层, 由 Falcon 事务处理层将上层 RDMA verbs 操作映射为请求-响应事务并保证 RDMA 顺序交付语义, 由 Falcon 数据包转发层实现基于延迟的拥塞控制和多路径负载平衡等.

eRDMA 和 Falcon 基于 RDMA 进行扩展改进, Shalev 等人[102]在 Micro 2020 中提出的可扩展可靠数据报(scalable reliable datagram, SRD)和 UEC 联盟[129]在 2023 年提出的超以太网传输(Ultra Ethernet Transport, UET)协议[106]则认为 RoCEv2 在拥塞控制算法、丢包处理、依赖 PFC, RC 可扩展性等方面存在各种问题, 无法满足 AI 和 HPC 对端到端数量、高带宽和分布式计算性能(如最小化尾部延迟)的需求, 并重新设计传输协议以取代 RoCEv2.

4 以太网 RDMA 网卡实现评估

4.1 网卡实现方式

目前, 以太网 RDMA 网卡主要基于 ASIC, FPGA, SoC 和软件实现 4 种方式, 表 10 对工业界实现的商用以太网 RDMA 网卡进行了统计汇总. 除了表 10 中所示的基于 RoCEv2 或 iWARP 的网卡, 阿里云公司自研 eRDMA 部署在其神龙架构 MOC 卡上[132], Shalev 等人[102]自研的 SRD 协议部署在 AWS Nitro 网卡上[133-134]. 以太网 RDMA 也存在软件形式的实现, 如 SRoCE[135]在用户空间实现 RDMA; Linux 内核中也实现 softRoCE[136]和 softiWARP[137]. 此外, 学术界常采用 FPGA 对设计网卡进行原型验证, 如 SRNIC[14], StaR[59]等.

基于 ASIC 实现的 RDMA 网卡可以最大程度发挥 RDMA 高带宽、低延迟的性能优势, 如 Mellanox 最新的 ConnectX-7 产品速度高达 400 Gbps, 可用于 AI、科学计算及大规模云数据中心工作负载, 在工业界批量生产最具性价比. 但基于 ASIC 实现的 RDMA

Table 10 Implementation of Commercial Ethernet RDMA NIC

表 10 商用以太网 RDMA 网卡实现

发布年份	实现方式	厂商	代表产品	支持协议	应用场景
2020	ASIC	Intel	100GbE E810[15]	RoCEv2, iWARP	HPC-AI、NFV、存储和混合云等高性能服务器工作负载、带宽密集型工作负载
2021	ASIC	NVIDIA Mellanox	400GbE ConnectX-7[16]	InfiniBand, RoCEv2	超大规模数据中心、HPC、人工智能、科学计算
2021	ASIC	MARVELL	FastLinQ 100GbE QL45000 系列[17]	RoCEv1, RoCEv2, iWARP	企业级数据中心、共有和私有云、托管服务提供商、电信
2021	FPGA	Xilinx	100GbE ERNIC[80]	RoCEv2	传感器数据采集、视频和图像捕获和传输、运行 RoCEv2 的远程存储节点
2022	ASIC	Chelsio	400GbE Terminator7[19]	RoCEv2, iWARP	数据中心网络、HPC、云计算、网络存储、流媒体应用、边缘产品
2022	NP-SoC	Huawei, Hisilicon	基于 Hi1822 的 200GbE SP670[18]	RoCEv2	OVS 加速、DPI 加速、虚拟化加速
2023	ASIC	Broadcom	200GbE BCM957508-P1200G[20]	RoCEv2	云和 Web2. 0 数据中心服务器、企业级数据中心、HPC 集群、机器学习集群、私有云、多节点容器平台、NVMe 存储分解(NVMe oF)数据库服务器
2023	SoC	AMD	200GbE Pensando DSC2-200[130]	RoCEv2	为大型云服务提供商和超大规模服务系统(如 Microsoft Azure, NetApp)提供计算、网络、存储和安全服务
2023	SoC	NVIDIA Mellanox	400GbE BlueField-3DPU[131]	RoCEv2	云网络、存储、安全、HPC/AI、电信与边缘计算

网卡在应用于复杂多变或新兴场景时存在功能固定、使用灵活性较差的问题. 此外, ASIC 的开发设计和验证成本高昂、周期较长, 尤其是对于需要硬件卸载 TCP/IP 的 iWARP 网卡复杂度更甚, 如 Chelsio T7[19] 与上一代 T6[138] 产品发布时间间隔长达 6 年.

基于 FPGA 实现的 RDMA 网卡可以在满足兼容性能较好的同时保证一定的灵活性和可编程性, 且处理资源丰富、开发周期较短, 因此常用于学术界进行 RDMA 功能原型验证. 在工业界, Xilinx ERNIC[80] 基于 FPGA 实现嵌入式 RDMA, 可用于边缘端传感器视频图像的数据传输. 但 FPGA 编程难度仍较高, 且在大规模部署时成本昂贵, 在数据中心领域 FPGA 实现的 RDMA 网卡相关产品较少.

基于 SoC 的网卡则往往集成包括 ASIC、FPGA、软核等众多逻辑资源, 可用于集成 RDMA 功能, 如使用 NP(network processor)-SoC 的华为 Hi1822[18] 在其 NP 流处理核心上实现 RDMA 逻辑, 相对于 FPGA 在灵活度和编程复杂度上更具优势, NVIDIA BlueField-3[131] 则直接集成 ConnectX-7 控制器以支持 RDMA 功能. 相比基于 ASIC 和 FPGA 实现的 RDMA 网卡, 基于 SoC 的网卡不仅仅实现 RDMA, 而且往往集成硬件加速器以支持安全、存储等更丰富的功能.

软件形式的 RDMA 实现性能最差, 但不需要网卡、交换机等硬件支持, 从而可以以最低成本部署并兼容互通其他形式实现的 RDMA 网卡. 选择合适的网卡实现形式需要根据具体应用场景、性能和成本等综合考量.

4.2 网卡评估方式

在商用网卡普遍黑盒的情况下, 对网卡进行测试和评估十分必要. 一方面, 数据中心运营商需要使用网卡供应商提供的无法访问的高度定制的硬件以及特定的系统配置, 因此其在实际部署 RDMA 网卡之前需要进行充分的测试以避免暂停帧风暴等异常情况出现[12]. 另一方面, 使用不同的 RDMA 选项在吞吐量方面最高会产生 70 倍的差异, 在占用主机 CPU 方面可相差 3.2 倍[74], 针对不同的应用场景和功能需求选择合适的原语、请求速度、消息大小、完成通知频率等以充分发挥硬件网卡的性能同样需要大量实际测试评估.

RDMA 网卡评估指标主要包括带宽、延迟、每秒消息数(messages per second, MPS)、CPU 利用率等, 对其评估主要有 2 种方式[12].

1)运行一组代表性应用程序工作负载. Wei 等人[76] 基于 ConnectX-4 网卡搭建实际本地机架级 RDMA 集群, 并运行流行的大规模在线事务处理(online transaction processing, OLTP)工作负载 TPC-C[139](模拟订单处理应用程序, CPU 密集型)和 SmallBank[140](模拟银行应用程序, 网络密集型), 对不同 RDMA 原语和传输方式的带宽和延迟进行对比评估. Huang 等人[141] 将 RDMA 中工作负载分为多请求单响应(如负载平衡算法等[142])、单请求多响应(如分布式键值存储等[143])、多对多(如分离式操作系统等[66])三类, 并基于 ConnectX-3 网卡搭建 CloudLab 集群[144], 对不同 RDMA 原语的带宽进行测试评估. Le 等人[69] 同样基于 CloudLab 搭建集群并测试评估集群负载和配置对网卡带宽、延迟、CPU 利用率的影响.

此外, Yu 等人[119] 在 SIGCOMM 2023 提出的 Lumina 为用户提供流量生成器以生成代表性的 RDMA 工作负载, 并结合可编程交换机注入各种网络事件, 从而线速模拟不同的网络场景. Lumina 将镜像数据包转储到专用服务器上以离线方式针对性分析 RDMA 的传输行为, 对网卡正确性和性能进行测试评估.

2)使用性能评估工具. 在目前的性能评估工具中, NVIDIA Mellanox 提供的 perftest[145] 开源测试工具集广泛用于 IB 和 RoCE 的软硬件调优、性能和功能(建链等)测试评估, 且提供的 NEO-Host[146] 管理软件向上提供大量诊断计数器等, 用于数据中心运营商监控或配置网卡. 俄亥俄州超算中心开源的 qperf 工具[147] 及俄亥俄州立大学开源的 OSU(Ohio State University)微基准测试套件[148] 同样支持对 IB 和 RoCE 进行带宽、延迟等性能测试. Kalia 等人[74] 提供的 RDMA-bench 开源软件[149] 可用于测评 RDMA 操作性能及 CPU 和 PCIe 开销.

以应用广泛的 perftest 为例, perftest 是基于 uverbs 编写的基准测试程序, 使用客户端-服务器(client-server)架构. 在客户端提供 *ib_write_bw*() 等函数用于 RDMA 操作的带宽、延迟测试, 并由 perftest 进一步调用 RDMA core 核心库函数 *ibv_reg_mr*(), *ibv_post_send*() 等与内核态或直接与硬件交互, 完成从设备初始化、资源注册、连接建立、下发 WR、获取 WC 等整个 RDMA 通信流程. 用户只需选择适当的测试参数(消息传输数据量、连接类型等)和测试命令, 由 perftest 使用 CPU 周期计数器获取时间戳并进行带宽、延迟的计算, 如 Flor[37] 用其对数据中心异构网卡互通进行性能评估, RoGUE[6] 用其对网卡 CPU 利用率进行评估等. 但 perftest 并不模拟实际流量, 无法预测实际应用负载的工作情况.

不同于上述工具单独测试硬件网卡, Kong 等人[12]

在 NSDI 2022 提出的 Collie 和 Liu 等人[40] 在 NSDI 2023 提出的 Hostping 诊断工具则综合考虑整个 RDMA 子系统进行性能瓶颈诊断. Collie 联合端系统上的网卡、内存以及 PCIe 等硬件设备，构建了一个包含主机各种流量、消息大小、连接数以及内存区域配置的全面搜索空间，并基于硬件网卡提供的性能、诊断计数器使用模拟退火算法和触发异常条件的最小特征集（minimal feature set, MFS）算法来寻找触发性能异常（异常低吞吐量、PFC 暂停帧风暴）的工作负载. 其后续工作 Hostping 则关注主机内网络瓶颈的监测和诊断. Hostping 通过网卡和主机内端点间的环回测试测量主机内延迟和带宽作为瓶颈诊断指标，并利用硬件监视器和数据分析器快速诊断、识别和响应主机内网络瓶颈. 此外，很多研究工作使用 Gem5 模拟器[8,57-58,60] 等模拟仿真网卡架构，并联合小规模环境（testbed）实际测试和大规模 NS-3[6,13-14,21,29,39,44,51,65,150] 或 OMNeT++[7,68,151] 网络模拟器仿真模拟的方式对网卡性能、拥塞控制算法等进行评估.

5 总结与展望

本文首先从以太网 RDMA 网卡的功能出发，对其通用模块架构进行了总结，并对比分析了其关键部件的功能实现方式；然后将以太网 RDMA 网卡的优化分为面向存储资源、可靠传输和应用相关 3 方面，并结合现有研究工作对相关优化技术进行详细的分类与剖析；最后对网卡的典型实现、性能评估方法和评估工具进行了汇总. 虽然现有的研究工作已采用多种优化方案来增强以太网 RDMA 网卡的扩展能力及其对网络和上层应用的适配能力，但在数据中心网络规模不断扩大、应用负载流量特性快速变化的情况下，以下 3 方面问题仍有待进一步研究.

1）软硬件协同方面. 当前优化方式中，网卡架构在硬件层面作改进虽可以实现较好的性能，但需增加新的网卡处理模块，难以实际应用部署，且不利于异构网卡互联互通. 在软件层面上载硬件功能、添加软件中间层等可降低硬件复杂度或简化用户编程，但会额外引入 CPU 开销，降低性能，因此如何进行高效的软硬协同和跨层设计仍是在硬件高性能和软件高可用性之间折中权衡的关键问题.

2）网络传输方面. 对于拥塞控制，无论 RDMA 运行于无损网络或有损网络，高效的拥塞控制对于数据中心网络传输性能都十分重要. 虽然已有较多相关研究，但尚无一种灵活、健壮且实施良好的拥塞控制机制可适配各种场景[38]. 在没有规范性拥塞管理的情况下，不同供应商甚至同一供应商的不同 RDMA 网卡间很难实现互操作性. 对于 PFC 和重传机制，基于 PFC 构造的无损网络往往采用 go-back-*N* 重传，但 PFC 会带来死锁、队头阻塞、拥塞扩散等一系列问题，阻碍了大规模数据中心网络的构建；除此之外，go-back-*N* 不支持多路径或无序传输，且重传代价过于昂贵[1]. IRN[7] 等采用的选择重传（ConnectX-6 中已支持选择重传）或许是未来消除以太网 RDMA 对 PFC 依赖的有效途径.

3）应用适配方面. 目前 RDMA 提供的单边操作类型难以满足众多应用需求，通过在网卡上卸载更多的数据操作[8,36] 和为用户提供扩展的单边原语或新型队列[85] 等以满足应用需要或简化 RDMA 编程等是研究趋势. 同时，随着数据中心规模的扩大和云化，以太网 RDMA 如何适配 AI/HPC 等新兴应用、如何实施合理的多租户性能隔离方案以及保障 RDMA 远程访问的安全性等问题变得十分重要. 此外，将 RDMA 扩展应用于区块链[152]、分布式机器学习[153]、数字超声数据采集系统[154]、射电天文系统[155] 等数据中心之外的场景也值得探索.

作者贡献声明：黄曼蒂和李韬共同负责学术界和工业界相关工作的搜集整理，论文整体架构的设计、撰写和修改；杨惠指导了以太网 RDMA 网卡优化部分并参与论文的完善和修改；李成龙指导了以太网 RDMA 网卡架构部分并参与论文整体架构的设计；张毓涛负责参考文献的梳理，以及面向可靠传输优化中拥塞控制部分的补充；孙志刚负责论文审阅并提出指导意见. 黄曼蒂和李韬为共同第一作者.

参 考 文 献

[1] Hoefler T, Roweth D, Underwood K, et al. Datacenter Ethernet and RDMA: Issues at hyperscale [J]. arXiv preprint, arXiv: 2302.03337, 2023

[2] Ma Xiaoxiao, Yang Fan, Wang Zhan, et al. Survey on smart network interface card[J]. Journal of Computer Research and Development, 2022, 59(1): 1−21(in Chinese)
(马潇潇，杨帆，王展，等. 智能网卡综述[J]. 计算机研究与发展，2022, 59(1): 1−21)

[3] Li Qiang, Xiang Qiao, Liu Derui, et al. From RDMA to RDCA: Toward high-speed last mile of data center networks using remote direct cache access [J]. arXiv preprint, arXiv: 2211.05975v2, 2023

[4] He Zhiqiang, Wang Dongyang, Fu Binzhang, et al. MasQ: RDMA for virtual private cloud [C/OL]//Proc of the 2020 ACM SIGCOMM

Conf. New York: ACM, 2020 [2024-07-06]. https://dl.acm.org/doi/abs/10.1145/3387514.3405849

[5] Dragojevic A, Narayanan D, Hodson O, et al. FaRM: Fast remote memory [C]//Proc of the 11th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2014: 401−414

[6] Le Yanfang, Stephens B, Singhvi A, et al. RoGUE: RDMA over generic unconverged Ethernet [C]//Proc of the ACM Symp on Cloud Computing. New York: ACM, 2018: 225−236

[7] Mittal R, Shpiner A, Panda A, et al. Revisiting network support for RDMA [C]//Proc of the 2018 ACM SIGCOMM Conf. New York: ACM, 2018: 313−326

[8] Hoefler T, Girolamo S D, Taranov K, et al. sPIN: High-performance streaming processing in the network [C/OL]//Proc of the Int Conf for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2017 [2024-07-06]. https://ieeexplore.ieee.org/document/9926279

[9] Guo Chuanxiong, Wu Haitao, Deng Zhong, et al. RDMA over commodity Ethernet at scale [C]//Proc of the 2016 ACM SIGCOMM Conf. New York: ACM, 2016: 202−215

[10] IEEE 802.1 Working Group. 802.1Qbb – priority-based flow control [EB/OL]. [2024-07-06]. https://1.ieee802.org/dcb/802-1qbb/

[11] Internet Engineering Task Force. RFC3168: The addition of explicit congestion notification (ECN) to IP [EB/OL]. [2024-07-06]. https://dl.acm.org/doi/book/10.17487/RFC3168

[12] Kong Xinhao, Zhu Yibo, Zhou Huaping, et al. Collie: Finding performance anomalies in RDMA subsystems [C]//Proc of the 19th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2022: 287−305

[13] Tang Jian, Wang Xiaoliang, Dai Huichen. Scalable RDMA transport with efficient connection sharing [C/OL]//Proc of the IEEE Int Conf on Computer Communications. Piscataway, NJ: IEEE, 2023 [2024-07-06]. https://ieeexplore.ieee.org/document/10228968

[14] Wang Zilong, Luo Layong, Ning Qingsong, et al. SRNIC: A scalable architecture for RDMA NICs [C/OL]//Proc of the 20th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2023 [2024-07-06]. https://www.usenix.org/ conference/nsdi 23/presentation/wang-zilong

[15] Intel. Intel E810 Ethernet network adapter [EB/OL]. [2024-07-06]. https://www.intel.cn/content/www/cn/zh/products/details/ethernet/800-network-adapters/e810-network-adapters/products.html

[16] NVIDIA. NVIDIA ConnectX−7 network adapter datasheet [EB/OL].[2024-07-06] https://nvdam.widen.net/s/csf8rmnqwl/infiniband-ethernet-datasheet-connectx-7-ds-nv-us-2544471

[17] MARVELL. MARVELL converged Ethernet network adapter [EB/OL]. [2024-07-06]. https://cn.marvell.com/products/ethernet-adapters-and-controllers/fastlinq-cna-adapters/documents.html

[18] Huawei Hisilicon. Huawei SP600 NIC [EB/OL]. [2024-07-06]. https://support.huawei.com/enterprise/zh/doc/EDOC1100309168?idPath=23710424%7C251364409%7C21782478%7C15791

[19] Chelsio. Chelsio T7 DPU products [EB/OL]. [2024-07-06]. https://www.chelsio.com/wp-content/uploads/resources/t7-dpu-asic.pdf

[20] Broadcom. Broadcom BCM957508-P1200G NIC datasheet [EB/OL]. [2024-07-06]. https://docs.broadcom.com/doc/957508-P1200G-DS1XX

[21] Singhvi A, Akella A, Gibson D, et al. 1RMA: Re-envisioning remote memory access for multi-tenant datacenters [C]//Proc of the 2020 ACM SIGCOMM Conf. New York: ACM, 2020: 708−721

[22] Vamanan B, Hasan J, Vijaykumar T N. Deadline-aware datacenter tcp (D2TCP)[J]. ACM SIGCOMM Computer Communication Review, 2012, 42(4): 115−126

[23] Marty M, Kruijf M D, Adriaens J, et al. Snap: A microkernel approach to host networking [C]//Proc of the 27th ACM Symp on Operating Systems Principles. New York: ACM, 2019: 399−413

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

[25] Mittal R, Lam V T, Dukkipati N, et al. TIMELY: RTT-based congestion control for the datacenter [C]//Proc of the 2015 ACM SIGCOMM Conf. New York: ACM, 2015: 537−550

[26] Zhang Qizhen, Bernstein P A, Berger D S, et al. Redy: Remote dynamic memory cache [J]. Very Large Data Base Endowment, 2021, 15(4): 766−779

[27] Firestone D, Putnam A, Mundkur S, et al. Azure accelerated networking: SmartNICs in the public cloud [C]//Proc of the 15th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2018: 51−66

[28] Kandula S, Sengupta S, Greenberg A, et al. The nature of data center traffic: Measurements & analysis [C]//Proc of the 9th ACM SIGCOMM Conf on Internet Measurement. New York: ACM, 2009: 202−208

[29] Zhu Yibo, Ghobadi M, Misra V, et al. ECN or delay: Lessons learnt from analysis of DCQCN and TIMELY [C]//Proc of the 12th Int Conf on Emerging Networking Experiments and Technologies. New York: ACM, 2016: 313−327

[30] Lu Yuanwei, Chen Guo, Ruan Zhenyuan, et al. Memory efficient loss recovery for hardware-based transport in datacenter [C]//Proc of the 1st Asia-Pacific Workshop on Networking. New York: ACM, 2017: 22−28

[31] Dragojevic A, Narayanan D, Castro M. RDMA reads: To use or not to use[J]. IEEE Data Engineering Bulletin, 2017, 40(1): 3−14

[32] Bai Wei, Abdeen S S, Agrawal, et al. Empowering Azure storage with RDMA [C]//Proc of the 20th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2023: 49−67

[33] Li Bojie, Ruan Zhenyuan, Xiao Wencong, et al. KV-Direct: High-performance in-memory key-value store with programmable NIC [C]//Proc of the 26th Symp on Operating Systems Principles. New York: ACM, 2017: 137−152

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

[35] Li Bojie, Cui Tianyi, Wang Zibo, et al. Socksdirect: Datacenter

sockets can be fast and compatible [C]//Proc of the 2019 ACM SIGCOMM Conf. New York: ACM, 2019: 90−103

[36] Sidler D, Wang Z, Chiosa M, et al. StRoM: Smart remote memory[C/OL]//Proc of the 15th European Conf on Computer Systems. New York: ACM, 2020 [2024-07-06]. https://dl.acm.org/doi/abs/10.1145/3342195.3387519

[37] Li Qiang, Gao Yixiao, Wang Xiaoliang, et al. Flor: An open high performance RDMA framework over heterogeneous RNICs [C]// Proc of the 17th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2023: 931−948

[38] Gao Yixiao, Li Qiang, Tang Lingbo, et al. When cloud storage meets RDMA [C]//Proc of the 18th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2021: 519−533

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

[40] Liu Kefei, Jiang Zhuo, Zhang Jiao, et al. Hostping: Diagnosing intra-host network bottlenecks in RDMA servers [C]//Proc of the 20th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2023: 15−29

[41] Chen Youmin, Lu Youyou, Shu Jiwu. Scalable RDMA RPC on reliable connection with efficient resource sharing [C/OL]//Proc of the 14th EuroSys Conf. New York: ACM, 2019 [2024-07-06]. https://dl.acm.org/do i/10.1145/3302424.3303968

[42] Miao Mao, Ren Fengyuan, Luo Xiaohui, et al. SoftRDMA: Rekindling high performance software RDMA over commodity ethernet [C]//Proc of the 1st Asia-Pacific Workshop on Networking. New York: ACM, 2017: 43−49

[43] Ma Teng, Chen Kang, Ma Shaonan, et al. Thinking more about RDMA memory semantics [C]//Proc of the IEEE Int Conf on Cluster Computing. Piscataway, NJ: IEEE, 2021: 456−467

[44] Cheng Wenxue, Qian Kun, Jiang Wanchun, et al. Re-architecting congestion management in lossless Ethernet [C]//Proc of the 17th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2020: 19−36

[45] Ma Shaonan, Ma Teng, Chen Kang, et al. A survey of storage systems in the RDMA era[J]. IEEE Transactions on Parallel Distributed Systems, 2022, 33(12): 4395−4409

[46] Lu Youyou, Shu Jiwu, Chen Youmin, et al. Octopus^{+}: An RDMA-enabled distributed persistent memory file system[J]. ACM Transactions Storage, 2017, 17(3): 1−25

[47] Ma Tengyu, Ma Tao, Song Zhuo, et al. X-RDMA: Effective RDMA middleware in large-scale production environments [C/OL]//Proc of the IEEE Int Conf on Cluster Computing. Piscataway, NJ: IEEE, 2019 [2024-07-06]. https://ieeexplore.ieee.org/document/8891004

[48] Shen Dian, Luo Junzhou, Dong Fang, et al. Distributed and optimal RDMA resource scheduling in shared data center networks [C]//Proc of the IEEE Int Conf on Computer Communications. Piscataway, NJ: IEEE, 2020: 606−615

[49] Wang Xiaoliang, Song Hexiang, Nguyen C, et al. Maximizing the benefit of RDMA at end hosts [C/OL]//Proc of the IEEE Int Conf on Computer Communications. Piscataway, NJ: IEEE, 2021 [2024-07-06]. https://ieeexplore.ieee.org/document/9488875

[50] Qiu Haonan, Wang Xiaoliang, Jin Tianchen, et al. Toward effective and fair RDMA resource sharing [C]//Proc of the 2nd Asia-Pacific Workshop on Networking. New York: ACM, 2018: 8−14

[51] Yu Peiwen, Xue Feiyang, Tian Chen, et al. Bifrost: Extending RoCE for long distance inter-DC links[C/OL] //Proc of the 31st IEEE Int Conf on Network Protocols. Piscataway, NJ: IEEE, 2023 [2024-07-06]. https://ieeexplore.ieee.org/document/10355634

[52] He Zhiqiang, Chen Yuxin, Hua Bei. RoUD: Scalable RDMA over UD in lossy data center networks [C]//Proc of the 23rd IEEE/ACM Int Symp on Cluster, Cloud Internet Computing. Piscataway, NJ: IEEE, 2023: 36−46

[53] Wang Dongyang, Fu Binzhang, Lu Gang, et al. vSocket: Virtual socket interface for RDMA in public clouds [C]//Proc of the 15th ACM SIGPLAN/SIGOPS Int Conf on Virtual Execution Environments. New York: ACM, 2019: 179−192

[54] Zang Dawei, Cao Zheng, Liu Xiaoli, et al. PROP: Using PCIe-based RDMA to accelerate rack-scale communications in data centers [C]//Proc of the 21st IEEE Int Conf on Parallel Distributed Systems. Piscataway, NJ: IEEE, 2015: 465−472

[55] Liao Yunkun, Wu Jingya, Lu Wenyan, et al. Optimize the TX architecture of RDMA NIC for performance isolation in the cloud environment [C]// Proc of the Great Lakes Symp on VLSI. New York: ACM, 2023: 29−35

[56] Han Shukai, Zhang Mi, Jiang Dejun, et al. HiStore: Rethinking hybrid index in RDMA-based key-value store [J]. arXiv preprint, arXiv: 2208. 12987, 2022

[57] Ma Xiaoxiao, Yang Fan, Wang Zhan, et al. A scalable RDMA network interface card with efficient cache management [C/OL]//Proc of the IEEE Int Symp on Circuits Systems. Piscataway, NJ: IEEE, 2023 [2024-07-06]. https://ieeexplore.ieee.org/abstract/document/10181426

[58] Kang Ning, Wang Zhan, Yang Fan, et al. csRNA: Connection-scalable RDMA NIC architecture in datacenter environment [C]//Proc of the 40th IEEE Int Conf on Computer Design. Piscataway, NJ: IEEE, 2022: 398−406

[59] Wang Xizheng, Chen Guo, Yin Xijin, et al. StaR: Breaking the scalability limit for RDMA [C/OL]//Proc of the 29th IEEE Int Conf on Network Protocols. Piscataway, NJ: IEEE, 2021 [2024-07-06]. https://ieeexplore.ieee.org/document/9651935

[60] Wang Zilong, Wan Xinchen, Zeng Chaoliang, et al. Accurate and scalable rate limiter for RDMA NICs [C]//Proc of the 7th Asia-Pacific Workshop on Networking. New York: ACM, 2023: 15−20

[61] Rothenberger B, Taranov K, Perrig A, et al. ReDMArk: Bypassing RDMA security mechanisms [C]//Proc of the 30th USENIX Security Symp. Berkeley, CA: USENIX Association, 2021: 4277−4292

[62] Taranov K, Rothenberger B, Perrig A, et al. sRDMA-Efficient NIC-based authentication and encryption for remote direct memory access [C]//Proc of the USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2020: 691−704

[63] Barthels C, Alonso G, Hoefler T. Designing databases for future high-

performance networks[J]. IEEE Data Engineering Bulletin, 2017, 40(1): 15−26

[64] Taranov K, Girolamo S D, Hoefler T. CoRM: Compactable remote memory over RDMA [C]//Proc of the Int Conf on Management of Data. New York: ACM, 2021: 1811−1824

[65] Xue Jiachen, Chaudhry M U, Vamanan B, et al. Dart: Divide and specialize for fast response to congestion in RDMA-based datacenter networks[J]. IEEE/ACM Transactions on Networking, 2018, 28(1): 322−335

[66] Tsai S Y, Zhang Yiying. LITE kernel RDMA support for datacenter applications [C]//Proc of the 26th Symp on Operating Systems Principles. New York: ACM, 2017: 306−324

[67] Xue Jiachen, Vijaykumar T N, Thottethodi M. Network interface architecture for remote indirect memory access (RIMA) in datacenters[J]. ACM Transactions on Architecture Code Optimization, 2020, 17(2): 1−22

[68] Taheri P, Menikkumbura D, Vanini E, et al. RoCC: Robust congestion control for RDMA [C]//Proc of the 16th Int Conf on Emerging Networking EXperiments Technologies. New York: ACM, 2020: 17−30

[69] Le Yanfang, Malekpourshahraki M, Stephens B, et al. On the impact of cluster configuration on RoCE application design [C]//Proc of the 3rd Asia-Pacific Workshop on Networking. New York: ACM, 2019: 64−70

[70] Kalia A, Kaminsky M, Andersen D G. FaSST: Fast, scalable and simple distributed transactions with two-sided (RDMA) datagram RPCs [C]//Proc of the 12th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2016: 185−201

[71] Kalia A, Kaminsky M, Andersen D G. Using RDMA efficiently for key-value services [C]//Proc of the 2014 ACM SIGCOMM Conf. New York: ACM, 2014: 295−306

[72] Kim D, Memaripour A, Badam A, et al. HyperLoop: Group-based NIC-offloading to accelerate replicated transactions in multi-tenant storage systems [C]//Proc of the 2018 ACM SIGCOMM Conf. New York: ACM, 2018: 297−312

[73] Kim D, Yu Tianlong, Liu H H, et al. FreeFlow: Software-based virtual RDMA networking for containerized clouds [C]//Proc of the 16th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2019: 113−126

[74] Kalia A, Kaminsky M, Andersen D G. Design guidelines for high performance RDMA systems [C]//Proc of the USENIX Conf on Usenix Annual Technical Conf. Berkeley, CA: USENIX Association, 2016: 437−450

[75] Openfabrics. Libibverbs release [EB/OL]. [2024-07-06]. https://www.openfabrics.org/downloads/libibverbs/

[76] Wei Xingda, Dong Zhiyuan, Chen Rong, et al. Deconstructing RDMA-enabled distributed transactions: Hybrid is better [C]//Proc of the 13th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2018: 233−251

[77] Zhang Yiwen, Gu Juncheng, Lee Youngmoon, et al. Performance isolation anomalies in RDMA [C]//Proc of the Workshop on Kernel-Bypass Networks. New York: ACM, 2017: 43−48

[78] Intel. Intel I350 Ethernet server adapter [EB/OL]. [2024-07-06]. https://www.intel.cn/content/www/cn/zh/products/details/ethernet/gigabit-network-adapters/i350-server-adapters.html

[79] Xilinx. Xilinx embedded target RDMA enabled v1.1 product guide [EB/OL]. [2024-07-06]. https://docs.xilinx.com/v/u/en-US/pg294-etrnic

[80] Xilinx. Xilinx embedded RDMA enabled NIC LogiCORE IP product guide [EB/OL]. [2024-07-06]. https://docs.xilinx.com/r/en-US/pg332-ernic

[81] Schelten N, Steinert F, Knapheide J, et al. A high-throughput, resource-efficient implementation of the RoCEv2 remote DMA protocol and its application[J]. ACM Transactions on Reconfigurable Technology, 2022, 16(1): 1−23

[82] Kong Xinhao, Chen Jingrong, Bai Wei, et al. Understanding RDMA microarchitecture resources for performance isolation [C]//Proc of the 20th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2023: 31−48

[83] IBTA. InfiniBand architecture specification [EB/OL]. [2024-07-06]. https://www.infinibandta.org/ibta-specification/

[84] Guo Zehua, Liu Sen, Zhang Zhili. Traffic control for RDMA-enabled data center networks: A survey[J]. IEEE Systems Journal, 2020, 14(1): 677−688

[85] Ryser A, Lerner A, Forencich A, et al. D-RDMA: Bringing zero-copy RDMA to database systems [C/OL]//Proc of the 12th Annual Conf on Innovative Data Systems Research. Chaminade, CA: CIDR, 2022 [2024-07-06]. https://www.cidrdb.org/cidr2022/papers/p77-ryser.pdf

[86] Zhu Yibo, Eran H, Firestone D, et al. Congestion control for large-scale RDMA deployments [C]//Proc of the 2015 ACM SIGCOMM Conf. New York: ACM, 2015: 523−536

[87] Tang Jian, Xu Tingting, Nguyen C, et al. Tuning target delay for RTT-based congestion control [C/OL]//Proc of the 30th IEEE Int Conf on Network Protocols. Piscataway, NJ: IEEE, 2022 [2024-07-06]. https://ieeexplore.ieee.org/document/9940420

[88] Fuhrer B, Shpigelman Y, Tessler C, et al. Implementing reinforcement learning datacenter congestion control in NVIDIA NICs [C]//Proc of the 23rd IEEE/ACM Int Symp on Cluster, Cloud and Internet Computing. Piscataway, NJ: IEEE, 2023: 331−343

[89] Perry J, Ousterhout A, Balakrishnan H, et al. Fastpass: A centralized "zero-queue" datacenter network[J]. ACM SIGCOMM Computer Communication Review, 2014, 44(4): 307−318

[90] Cho I, Jang K, Han Dongsu. Credit-scheduled delay-bounded congestion control for datacenters [C]//Proc of the 2017 ACM SIGCOMM Conf. New York: ACM, 2017: 239−252

[91] NVIDIA. Introduction to resilient RoCE [EB/OL]. [2024-07-06]. https://enterprise-support.nvidia.com/s/article/introduction-to-resilient-roce---faq#jive_content_id_What_is_Resilient_RoCE

[92] Mitchell C, Geng Yifeng, Jinyang Li. Using one-sided RDMA reads to build a fast, CPU-efficient key-value store [C]//Proc of the

USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2013: 103−114

[93] Yang Jian, Izraelevitz J, Swanson S. Orion: A distributed file system for non-volatile main memory and RDMA-capable networks [C]//Proc of the 17th USENIX Conf on File and Storage Technologies. Berkeley, CA: USENIX Association, 2019: 221−234

[94] Yang Jian, Izraelevitz J, Swanson S. FileMR: Rethinking RDMA networking for scalable persistent memory [C]//Proc of the 17th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2020: 111−125

[95] Zamanian E, Yu Xiangyao, Stonebraker M, et al. Rethinking database high availability with RDMA networks[J]. Very Large Data Base Endowment, 2019, 12(11): 1637−1650

[96] Miao Rui, Zhu Lingjun, Ma Shu, et al. From Luna to Solar: The evolutions of the compute-to-storage networks in Alibaba cloud [C]//Proc of the 2022 ACM SIGCOMM Conf. New York: ACM, 2022: 753−766

[97] Zhu Lingjun, Shen Yifan, Xu Erci, et al. Deploying user-space TCP at cloud scale with LUNA [C]//Proc of the USENIX Annual Technical Conf. Berkeley, CA: USENIX Association, 2023: 673−687

[98] Broadcom VMware. VMware technical journal: Toward a paravirtual vRDMA device for VMware ESXi guests [R]. Palo Alto, CA: VMware, 2012: 22−27

[99] Pfefferle J, Stuedi P, Trivedi A K, et al. A hybrid I/O virtualization framework for RDMA-capable network interfaces [C]//Proc of the 11th ACM SIGPLAN/SIGOPS Int Conf on Virtual Execution Environments. New York: ACM, 2015: 17−30

[100] Zhang Yiwen, Tan Yue, Stephens B E, et al. Justitia: Software multi-tenancy in hardware kernel-bypass networks [C]//Proc of the 19th USENIX Symp on Networked Systems Design and Implementation. Berkeley, CA: USENIX Association, 2022: 1307−1326

[101] Amaro E, Luo Zhihong, Ousterhout A, et al. Remote memory calls [C]//Proc of the 19th ACM Workshop on Hot Topics in Networks. New York: ACM, 2020: 38−44

[102] Shalev L, Ayoub H, Bshara N, et al. A cloud-optimized transport protocol for elastic and scalable HPC[J]. IEEE Micro, 2020, 40(6): 67−73

[103] Aliyun. eRDMA [EB/OL]. [2024-07-06]. https://github.com/alibaba/elastic-rdma-drivers

[104] Xing Jiarong, Hsu K, Qiu Yiming, et al. Bedrock: Programmable network support for secure RDMA systems [C]//Proc of the 31st USENIX Security Symp. Berkeley, CA: USENIX Association, 2022: 2585−2600

[105] Google. Falcon [EB/OL]. [2024-07-06]. https://cloud.google.com/blog/topics/systems/introducing-falcon-a-reliable-low-latency-hardware-transport

[106] Ultra Ethernet. UET [EB/OL]. [2024-07-06]. https://ultraethernet.org/wp-content/uploads/sites/20/2023/10/23.07.12-UEC-1.0-Overview-FINAL-WITH-LOGO.pdf

[107] Chen Youmin, Lu Youyou, Luo Shengmei, et al. Survey on RDMA-based distributed storage systems[J]. Journal of Computer Research and Development, 2019, 56(2): 227−239 (in Chinese)
(陈游旻, 陆游游, 罗圣美, 等. 基于 RDMA 的分布式存储系统研究综述[J]. 计算机研究与发展, 2019, 56(2): 227−239)

[108] Openfabrics. Dynamically-connected transport service [EB/OL]. [2024-07-06]. https://www.openfabrics.org/images/eventpresos/workshops2014/DevWorkshop/presos/Monday/pdf/05_DC_Verbs.pdf

[109] Matthew J K, Jaidev K S, Dhabaleswar K P. Scalable MPI design over InfiniBand using extended reliable connection [C]//Proc of the IEEE Int Conf on Cluster Computing. Piscataway, NJ: IEEE, 2008: 203−212

[110] Du Xinle, Xu Ke, Li Tong, et al. Traffic control for data center network: State of the art and future research[J]. Chinese Journal of Computers, 2021, 44(7): 1287−1309 (in Chinese)
(杜鑫乐, 徐恪, 李彤, 等. 数据中心网络的流量控制: 研究现状与趋势[J]. 计算机学报, 2021, 44(7): 1287−1309)

[111] Liu Jingling, Huang Jiawei, Jiang Wanchun, et al. Survey on load balancing mechanism in data center[J]. Journal of Software, 2021, 32(2): 300−326 (in Chinese)
(刘敬玲, 黄家玮, 蒋万春, 等. 数据中心负载均衡方法研究综述[J]. 软件学报, 2021, 32(2): 300−326)

[112] Xu Lisong, Harfoush K, Rhee Injong. Binary increase congestion control (BIC) for fast long-distance networks [C]//Proc of the IEEE Int Conf on Computer Communications. Piscataway, NJ: IEEE, 2004: 2514−2524

[113] Ha Sangtae, Rhee Injong, Xu Lisong. CUBIC: A new TCP-friendly high-speed TCP variant[J]. ACM Special Interest Group on Operating Systems, 2008, 42(5): 64−74

[114] Alizadeh M, Greenberg A G, Maltz D A, et al. Data center TCP (DCTCP)[J]. ACM SIGCOMM Computer Communication Review, 2010, 40(4): 63−74

[115] IEEE 802.1 Working Group. 802.1Qau-congestion notification [EB/OL]. [2024-07-06]. https://1.ieee802.org/dcb/802-1qau/

[116] Tessler C, Shpigelman Y, Dalal G, et al. Reinforcement learning for datacenter congestion control[J]. ACM SIGMETRICS Performance Evaluation Review, 2021, 49(2): 43−46

[117] Zhang Miao, Wu Jianping, Lin Chuang. A review of research on end-to-end congestion control for the Internet[J]. Journal of Software, 2002, 13(3): 354−363 (in Chinese)
(章淼, 吴建平, 林闯. 互联网端到端拥塞控制综述[J]. 软件学报, 2002, 13(3): 354−363)

[118] Dally W J, Seitz C L. Deadlock-free message routing in multiprocessor interconnection networks[J]. IEEE Transactions on Computers, 1987, 36(5): 547−553

[119] Yu Zhuolong, Su BoWei, Bai Wei, et al. Understanding the micro-behaviors of hardware offloaded network stacks with Lumina [C]//Proc of the 2023 ACM SIGCOMM Conf. New York: ACM, 2023: 1074−1087

[120] RedHat. What is cloud storage? [EB/OL]. [2024-07-06]. https://www.redhat.com/zh/topics/data-storage/what-is-cloud-storage

[121] Chen Sixin. Optimization study of RDMA communication in distributed block storage systems [D]. Wuhan: Huazhong University of Science and Technology, 2022 (in Chinese)
(陈思新. 分布式块存储系统中 RDMA 通信优化研究 [D]. 武汉: 华中科技大学, 2022)

[122] IBM. What is file storage? [EB/OL]. [2024-07-06]. https://www.ibm.com/cn-zh/topics/file-storage

[123] Azure. Azure blockchain service [EB/OL]. [2024-07-06]. https://azure.microsoft.com/en-us/services/blockchain-service/

[124] The Ohio State University. High-performance big data project [EB/OL]. [2024-07-06]. http://hibd:cse:ohio-state:edu/

[125] Carbone P, Katsifodimos A, Ewen S, et al. Apache Flink™: Stream and batch processing in a single engine[J]. IEEE Computer Society, 2015, 36(4): 28−33

[126] Li Hao, Kadav A, Kruus E, et al. Malt: Distributed data-parallelism for existing ml applications [C/OL]//Proc of the 10th European Conf on Computer Systems. New York: ACM, 2015 [2024-07-06]. https://dl.acm.org/doi/10.1145/2741948.2741965

[127] Abadi M, Barham P, Chen Jianmin, et al. Tensorflow: A system for large-scale machine learning [C]//Proc of the 12th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2016: 265−283

[128] Peripheral Component Interconnect Special Interest Group. SR-IOV [EB/OL]. [2024-07-06]. https://pcisig.com/specifications/iov/single_root/

[129] Ultra Ethernet. UEC [EB/OL]. [2024-07-06]. https://ultraethernet.org/

[130] AMD Pensando. DSC2−200 distributed services card [EB/OL]. [2024-07-06]. https://www.amd.com/system/files/documents/pensando-dsc-200-product-brief.pdf

[131] NVIDIA. BlueField−2DPU [EB/OL]. [2024-07-06]. https://www.nvidia.com/content/dam/en-zz/Solutions/Data-Center/documents/datasheet-nvidia-bluefield-3-dpu.pdf

[132] Aliyun. eRDMA [EB/OL]. [2024-07-06]. https://help.aliyun.com/zh/ecs/user-guide/erdma/?spm=a2c4g.11186623.0.0.52db44bbxNFCyy

[133] Amazon . ENA express: Improved network latency and per-flow performance on EC2 [EB/OL]. [2024-07-06]. https://aws.amazon.com/cn/blogs/aws/new-ena-express-improved-network-latency-and-per-flow-performance-on-ec2/

[134] Amazon. AWS Nitro system [EB/OL]. [2024-07-06]. https://aws.amazon.com/cn/ec2/nitro

[135] Pandey S M, Shashidhara R. SRoCE: Software RDMA over commodity Ethernet [EB/OL]. 2020 [2024-07-06]. https://homes.cs.washington.edu/~rajaths/sRoCE.pdf

[136] Liss L. The Linux softROCE driver [EB/OL]. [2024-07-06]. https://www.openfabrics.org/images/eventpresos/2017presentations/205_SoftRoCE_LLiss.pdf

[137] Linux. softiWARP [EB/OL]. [2024-07-06]. https://lxr.linux.no/linux+v5.15.91/drivers/infiniband/sw/siw/

[138] Chelsio.ChelsioT6[EB/OL].[2024-07-06].https://www.chelsio.com/wp-content/uploads/resources/T6-Architecture.pdf

[139] The Transaction Processing Council. TPC-C benchmark v5.11 [EB/OL]. [2024-07-06]. http://www.tpc.org/tpcc/

[140] The H-Store Team. SmallBank benchmark [EB/OL]. [2024-07-06]. http://hstore.cs.brown.edu/documentation/deployment/benchmarks/smallbank/

[141] Huang Haoju, Ghandeharizadeh S. An evaluation of RDMA-based message passing protocols [C]//Proc of the IEEE Int Conf on Big Data. Piscataway, NJ: IEEE, 2019: 3340−3349

[142] Hong Yuju, Thottethodi M. Understanding and mitigating the impact of load imbalance in the memory caching tier [C/OL]//Proc of the 4th Annual Symp on Cloud Computing. New York: ACM, 2013 [2024-07-06]. https://dl.acm.org/doi/10.1145/2523616.2525970

[143] Redis. Redis open source in-memory data store [EB/OL]. [2024-07-06]. https://redis.io/

[144] CloudLab. CloudLab infrastructure for cloud computing [EB/OL]. [2024-07-06]. https://cloudlab.us/

[145] OFED. Linux perftest [EB/OL]. [2024-07-06]. https://github.com/linux-rdma/perftest

[146] NVIDIA Mellanox. NEO-Host [EB/OL]. [2024-07-06]. https://www.nvidia.cn/networking/management-software/

[147] Ohio Supercomputer Center. Linux qperf [EB/OL]. [2024-07-06]. https://github.com/linux-rdma/qperf

[148] Ohio State University. OSU benchmarks [EB/OL]. [2024-07-06]. https://mvapich.cse.ohio-state.edu/benchmarks/

[149] Carnegie Mellon University. rdma_bench [EB/OL]. [2024-07-06]. https://github.com/efficient/rdma_bench

[150] NS−3. The NS−3 discrete-event network simulator [EB/OL]. [2024-07-06]. http://www.nsnam.org

[151] OMNeT++. OMNeT++ discrete event simulator [EB/OL]. [2024-07-06]. https://omnetpp.org/

[152] Huang Bo, Jin Li, Lu Zhihui, et al. BoR: Toward high-performance permissioned blockchain in RDMA-enabled network[J]. IEEE Transactions on Services Computing, 2020, 13(2): 301−313

[153] Ren Yufei, Wu Xingbo, Zhang Li, et al. iRDMA: Efficient use of RDMA in distributed deep learning systems [C]//Proc of the 19th IEEE Int Conf on High Performance Computing and Communications. Los Alamitos, CA: IEEE Computer Society, 2017: 231−238

[154] Cossettini A, Taranov K, Vogt C, et al. A RDMA interface for ultra-fast ultrasound data-streaming over an optical link [C]//Proc of the Design, Automation & Test in Europe Conf & Exhibition. Leuven, BEL: European Design and Automation Association, 2022: 80−83

[155] De Laat W. The application of RDMA over converged Ethernet data transport for radio-astronomy systems [D]. Delft, Netherlands: Delft University of Technology, 2022



Huang Mandi, born in 1999. Master. Her main research interest includes RDMA.

黄曼蒂, 1999 年生. 硕士. 主要研究方向为 RDMA.



Li Tao, born in 1983. PhD, associate professor. His main research interest includes high-performance network system and chip.

李　韬, 1983 年生. 博士, 副研究员. 主要研究方向为高性能网络系统及芯片.



Yang Hui, born in 1986. PhD, associate professor. Her main research interests include RDMA and network processor.

杨　惠, 1986 年生. 博士, 副研究员. 主要研究方向为 RDMA、网络处理器.



Li Chenglong, born in 1995. PhD, assistant professor. His main research interests include time-sensitive networking, in-network computing, and RISC-V.

李成龙, 1995 年生. 博士, 助理研究员. 主要研究方向为时间敏感网络、在网计算、RISC-V.



Zhang Yutao, born in 2000. Master. His main research interests include RDMA and data center congestion control.

张毓涛, 2000 年生. 硕士. 主要研究方向为 RDMA、数据中心拥塞控制.



Sun Zhigang, born in 1973. PhD, professor. His main research interests include software-defined network, time-sensitive networking, network architecture, and network security.

孙志刚, 1973 年生. 博士, 研究员. 主要研究方向为软件定义网络、时间敏感网络、网络体系结构、网络安全.