计 算 机 研 究 与 发 展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(4): 1039−1062, 2026

新一代互连技术 CXL 及其应用综述

李仁刚[1,2] 李 霞[3] 刘 俊[2] 侯鹏宇[2] 王彦伟[2] 杨广文[1]
[1](清华大学计算机科学与技术系 北京 100084)
[2](浪潮(北京)电子信息产业有限公司 北京 100085)
[3](浪潮电子信息产业股份有限公司 济南 250101)
(lirg@ieisystem.com)

Survey of New Generation Interconnect Technology CXL and Its Applications

Li Rengang[1,2], Li Xia[3], Liu Jun[2], Hou Pengyu[2], Wang Yanwei[2], and Yang Guangwen[1]
[1] (*Department of Computer Science and Technology*, *Tsinghua University*, *Beijing* 100084)
[2] (*IEIT SYSTEMS* (*Beijing*) *Co.*, *Ltd.*, *Beijing* 100085)
[3] (*IEIT SYSTEMS Co.*, *Ltd.*, *Jinan* 250101)

Abstract Driven by emerging high-performance computing applications, the demand for high-speed interconnects across heterogeneous computing system components has surged dramatically, making interconnect technologies a key research focus in the computer architecture field in recent years. The open standard CXL (compute express link), with its advantages of high scalability, high bandwidth, and low latency, has become the mainstream of interconnect technology in heterogeneous computing systems, which has gained much attention from both academia and industry. This survey comprehensively analyzes and summarizes CXL technology from three aspects, including CXL protocol architecture, device types and its performance characteristics, as well as CXL memory management and application research. Firstly, the overall architecture of the CXL protocol is introduced. The cache coherence mechanism is analyzed based on typical protocol transaction flows. Then, we provide an overview of CXL device types and development progress in industrial products domestically and abroad, followed by systematical analysis of real CXL device's performance characteristics. Finally, we summarize current academic application research, scenarios and challenges on the CXL technology from the perspective of CXL device types. On this basis, future research directions are discussed to promote its application research and industrial implementation in various fields, such as memory pooling, CXL-based processing-in-memory architecture and heterogeneous collaborative computing.

Key words CXL interconnect technology; cache coherency protocol; heterogeneous computing systems; memory expansion; memory disaggregation and pooling; memory sharing; heterogeneous collaborative computing

摘 要 在新型高性能计算应用的驱动下,异构计算系统对设备间高速互连的需求呈现激增态势,相关互连技术已成为体系结构领域近年来的研究热点。CXL(compute express link)以高扩展性、高带宽和低时延的优势,成为新一代互连技术的主流发展趋势,受到了学术界和产业界的大量关注。从 CXL 互连协议、CXL 设备产品及性能特点、CXL 内存管理与应用研究 3 个方面进行分析总结:首先,介绍 CXL 互连协议的总体架构,基于 CXL 协议的典型事务流,分析了其缓存一致性维护机制。然后,概述 CXL 设备类型及产品研发的国内外进展,并系统深入地分析了 CXL 设备的性能特点。最后,根据 CXL 的不同设备类型总

收稿日期:2025-03-18;**修回日期**:2025-11-03
基金项目:山东省自然科学基金项目(ZR2024QF124)
This work was supported by Shandong Provincial Natural Science Foundation (ZR2024QF124).
通信作者:刘俊(liujun12@ieisystem.com)

结梳理了当前学术界对CXL互连技术的应用研究、案例与挑战,并据此就未来的研究方向进行了展望,以促进其在内存扩展与池化、新型存算架构、加速器协同计算等领域的应用研究及产业界落地。

关键词 CXL互连技术;缓存一致性协议;异构计算系统;内存扩展;内存分离与池化;内存共享;异构协同计算

中图法分类号 TP334.7; TP303

DOI: 10.7544/issn1000-1239.202550211 **CSTR:** 32373.14.issn1000-1239.202550211

互连技术连接了计算、内存和存储等硬件资源,是计算系统的核心组成部分。随着人工智能应用的不断发展,算力需求急剧增加,新型加速器架构不断涌现,对异构互连提出了更高的要求。然而,过去20年间,计算硬件的峰值算力增加了60 000倍,内存带宽提升100倍,但互连带宽却只提高了30倍[1]。计算、内存、互连等技术的性能发展速度不匹配,导致“内存墙”与“IO墙”问题日趋严重,传统的计算架构逐渐成为新型应用发展的瓶颈。

为突破计算系统中硬件资源互连的性能瓶颈,近年来涌现了Gen-Z[2], OpenCAPI[3], CXL(compute express link), NVLink-C2C[4], CCIX[5]等多种新型互连协议,它们提供了计算单元间的缓存一致性维护功能,使得CPU与加速器间的数据交互更为高效。其中,CXL, OpenCAPI等协议通过维护CPU与加速器间的缓存一致性,显著优化了数据交换效率;NVLink实现了统一内存编址与高带宽点对点互连,并在NVlink-C2C中支持NVIDIA CPU与GPU间的缓存一致性。这些互连技术避免了频繁数据拷贝带来的性能损耗,实现了高带宽、低延迟的直接数据访问,从而显著提升异构计算的系统性能。

CXL互连协议因其突破性的设计备受瞩目。它基于PCIe物理层构建,凭借更高的引脚效率(pin-efficiency),相比传统DDR接口,可实现更高效的内存扩展;同时,CXL在PCIe协议的基础上引入了硬件维护的缓存一致性功能,支持CPU与加速器之间细粒度的数据交互,加速器和CPU可以缓存对方内存中的数据,并在需要时直接从本地缓存中读取或写入数据,显著减少数据迁移开销。此外,在CXL 2.0中通过单级交换机实现了内存池化,CXL 3.0进一步扩展为多层交换架构,提供了内存池化与共享的功能,实现CPU与内存资源的解耦,有效缓解云计算资源闲置问题,提升系统灵活性与利用率。凭借技术优势与开放生态,CXL联盟成员数6年内激增至超250家[6],并通过吸收OpenCAPI和Gen-Z协议技术,成为主导行业的互连标准,持续推动异构计算架构革新。

鉴于CXL高速互连技术的广阔前景,国内外已有大量研究成果。但是据我们调研,当前对CXL技术研究进行系统性梳理的工作仍较为有限,Sharma等人[7]对CXL互连技术进行了详细的解读,包括CXL发展历史与协议设计细节,但未对目前基于CXL的应用研究进行综述。文献[8]主要聚焦于基于CXL构建的单机系统应用与分布式计算系统应用,未深入探讨CXL协议机制及其硬件性能特点。为帮助后续研究者深入理解与应用CXL协议技术,本文主要围绕协议架构、设备性能特点和应用研究3个维度展开综述,系统梳理了CXL的技术特点以及在产业界和学术界的研究现状,旨在推动CXL的应用与发展。

本文整体框架如图1所示,主要贡献包括以下3个方面:

1)介绍了CXL互连协议及其缓存一致性机制、CXL设备类型及相关产品的研发进展,系统分析了CXL真实设备的性能特点。

2)总结了CXL Type 3设备的应用研究进展,梳理了CXL内存的模拟方法并对比了现有研究的实验方案。归纳了CXL Type 1/2设备的应用场景,并深入分析了其硬件设备研发的挑战。

3)基于当前研究进展,展望了CXL互连技术未来研究方向及面临的国产化挑战。

1 CXL互连协议介绍

CXL是基于PCIe总线5.0版本之上的新型总线协议(CXL 1.0与CXL 2.0基于PCIe 5.0物理层、CXL 3.0基于PCIe 6.0),延续了PCIe协议的分层结构,其整体架构如图2所示。弹性总线(flex bus)端口在链路训练期间通过协议协商机制决定选择原始的PCIe协议接口或者CXL协议接口。物理层负责物理信息交换、链路初始化与维护;数据链路层负责可靠的数据传输服务,包括流量控制和信用等机制;事务层处理与每个协议相关联的事务,进行协议消息的封装

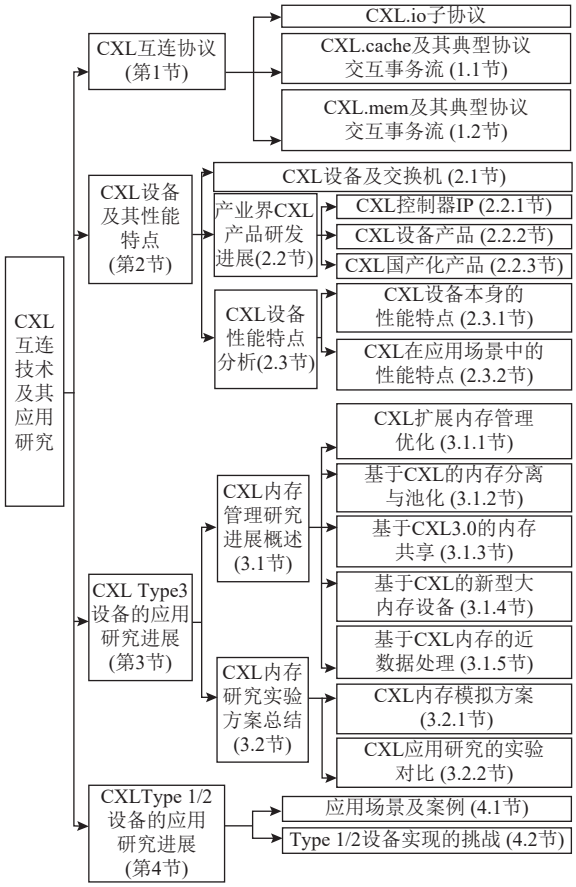


Fig. 1 Overall architecture of CXL interconnect technology and application research

图 1 CXL 互连技术与应用研究整体架构

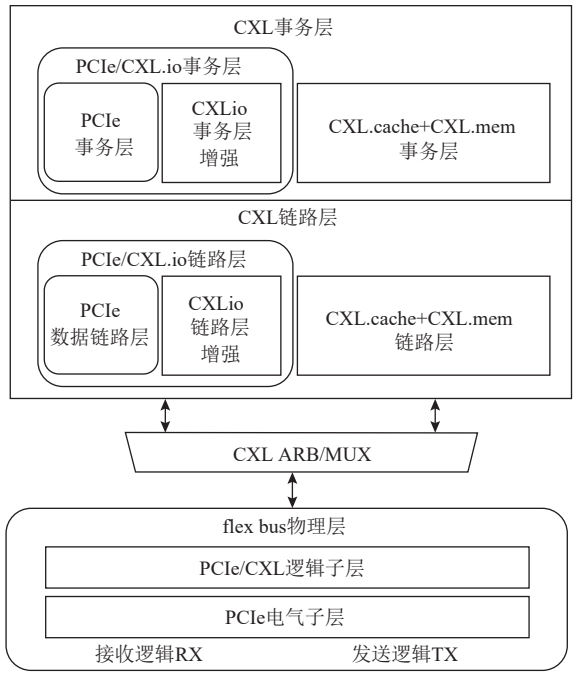


Fig. 2 Overall architecture of CXL protocol

图 2 CXL 协议总体架构

和解析。软件可以通过访问各层的寄存器进行配置、链路状态获取和控制管理。

CXL 是一种多协议架构，共包含 3 个子协议：CXL.io, CXL.cache, CXL.mem, 其中 CXL.io 基于 PCIe 协议负责设备发现、配置地址空间、主机物理地址查找和中断 DMA 操作等，并为 IO 设备提供非一致性的 load/store 访问。CXL.cache 和 CXL.mem 是新增的协议类型，可以实现主机与设备之间基于缓存一致性的内存访问，两者共享链路层与事务层，并借助动态仲裁复用机制（ARB/MUX）在 PCIe 物理层上实现协议融合传输。

1.1 CXL.cache 协议

CXL.cache 使 CXL 设备能够低延迟高效地缓存主机内存中的数据，并通过通道中传输的事务消息来维护设备缓存与主机的一致性。CXL.cache 定义了主机（host, H）与设备（device, D）间的 6 种消息通道：D2H Req, D2H Resp, D2H Data, H2D Req, H2D Resp, H2D Data（Req: 请求，Resp: 响应，Data: 数据），其中每条消息报文都与一致性数据请求或状态转换有关。其缓存一致性维护机制可通过形式化方法进行精确建模与验证[9]。CXL.cache 协议将缓存的一致性维护拓展到了主机之外，主机代理（home agent, HA）使用监听消息的方式管理缓存间的一致性。

图 3 中展示了主机内存被 CXL 设备 1 和 CXL 设备 2 共享，由 CXL.cache 协议维护主机与设备缓存一致性的一个典型事务交互流程[10]。

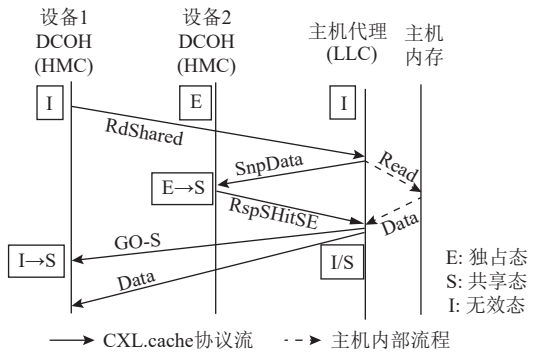


Fig. 3 Flowchart of CXL.cache protocol interaction under D2H shared read request

图 3 设备对主机共享读请求下 CXL.cache 协议交互流程图

图 3 中，设备 1 想要以共享方式（S）缓存主机内存地址 64 B 数据。初始时设备 1 的 HMC（host memory cache）中没有该主机数据的副本（处于无效态（I）），而设备 2 缓存中存在该数据的独占态副本（E）。设备 1 首先向主机代理发送 RdShared 请求（D2H Req），

主机代理接收到该请求后，对设备 2 缓存行发起 SnpData 监听（H2D Req），请求其缓存中数据副本由 E 态修改为 I 态或 S 态。设备 2 的设备一致性引擎（device coherency engine, DCOH）处理主机监听请求并把缓存行状态修改为 S 后发送 RspSHitSE 响应（D2H Resp）给主机，主机对设备 2 的监听结束。在主机代理发送对设备 2 监听消息的同时，也对主机内存发送了读数据请求。收到数据和 RspSHitSE 响应消息后，主机发送 GO-S 响应（H2D Resp）给设备 1，然后返回请求数据（H2D Data），主机 LLC（last level cache）从状态 I 变为 S 或者仍为 I（由主机代理决定）。从而设备 1 获得该数据的共享态副本，完成交互流程。

1.2 CXL.mem 协议

CXL.mem 允许主机处理器将连接的 CXL 设备内存视为其内存地址空间的一部分，可以使用 load/store 命令访问设备连接的内存，并维护主机缓存和设备内存数据的一致性。这使得系统可以通过附加设备（如内存扩展卡）来扩展物理内存容量，而不受传统架构 DDR 内存通道数量的限制。

在 CXL 3.0 版本中，CXL.mem 定义了主设备（master, M）到从设备（subordinate, S）间的 6 种消息通道：M2S Req, M2S RwD, M2S BIRsp, S2M NDR, S2M DRS, S2M BISnp（Req: 未携带数据的请求，RwD: 携带数据的请求，NDR: 未携带数据的应答，DRS: 携带数据的应答，BISnp: 反向无效监听，BIRsp: 反向无效应答）。其中，反向无效监听通道可以用来实现设备 HDM（host-managed device memory）的一致性[10]。

图 4 展示了设备利用反向无效监听机制维护主机缓存与设备一致性的交互流程。设备发送 RdAny 请求，旨在对任何状态下的缓存行进行完整读取，设备 DCOH 中监听过滤器（snoop filter, SF）命中，表示在主机缓存中存在有设备要读取数据的副本，随后向主机发送 BISnpData 请求（S2M BISnp 消息），请求反向无效监听该共享或独占态的副本数据，主机返回 BIRspI 响应（M2S BIRsp 消息）给设备的 DCOH，表示完成了相应缓存行的反向无效监听，且主机缓存行处于无效态。然后 DCOH 中的监听过滤器释放对主机中该缓存行状态的记录，再向设备内存请求读取数据，最终通过 DCOH 返回数据给设备缓存。

由此可见，CXL.mem 引入的反向无效监听机制赋予了设备主动维护主机缓存一致性的能力，弥补了 CXL 3.0 之前仅依赖 CXL.cache 处理一致性的局限。这使得设备能够更高效地映射与管理更大规模的内存资源。尤其在同一设备内存被多主机共享的场景下，该机制可有效管理多主机与设备间的缓存一致性。

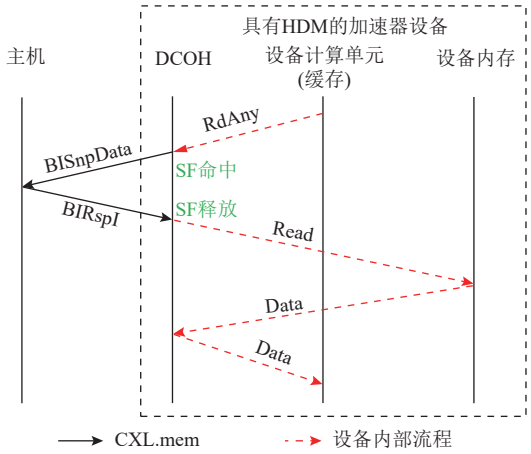


Fig. 4 Flowchart of back-invalidate snoop transaction interaction under the read request of device to local memory

图 4 设备对本地内存读请求下反向无效监听事务交互流程图

2 CXL 设备及性能特点

2.1 CXL 设备类型

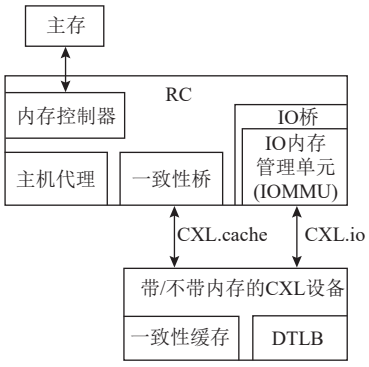
为了满足不同应用场景的需求，CXL 提供了多种不同子协议的灵活组合实现，共支持 Type 1 设备、Type 2 设备和 Type 3 设备 3 种设备类型，其典型架构如图 5 所示。主机侧根复合体（root complex, RC）负责管理 CPU、主机内存子系统和设备子系统之间的交互，其中主机代理负责管理主机缓存与设备的一致性，并发起主机对设备内存的一致性读写请求，IO 桥负责处理 CXL.io 事务，如寄存器访问、配置、初始化、中断和 DMA 等操作。

2.1.1 CXL Type 1 设备

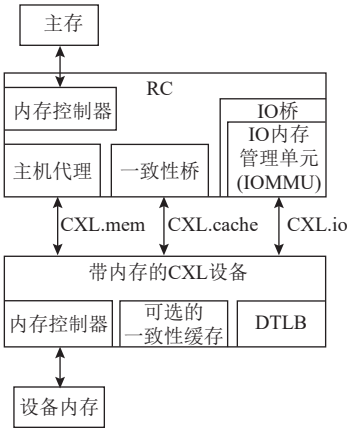
CXL Type 1 设备支持 CXL.io 和 CXL.cache 协议，具备可选的私有内存，其典型应用场景是智能网卡。此类设备可通过 CXL.cache 协议处理与主机的缓存一致性事务，包括一致性缓存和设备端 DTLB（device translation lookaside buffer）等组件，以支持不限数量的原子操作以及设备端的快速地址转换。

2.1.2 CXL Type 2 设备

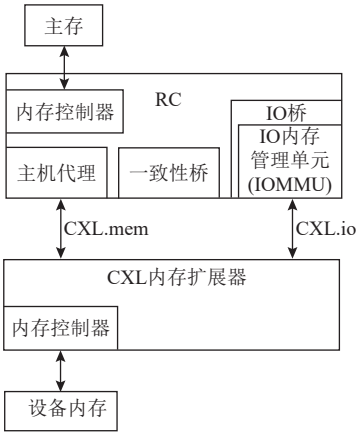
CXL Type 2 设备完整支持 CXL.io, CXL.cache, CXL.mem 协议，基于 CXL 协议的缓存一致性维护，主机能够将操作数写入设备内存，并在设备完成计算后从设备内存中快速读取结果，与此同时，由于设备和主机可以缓存对方内存中的数据，支持主机与



(a) CXL Type 1设备



(b) CXL Type 2设备



(c) CXL Type 3设备

Fig. 5 Typical architecture of CXL device

图 5 CXL 设备的典型架构

设备间的频繁数据同步，并在需要时直接从本地缓存中读写数据，显著减少数据在主机和设备间的搬移开销。此类设备的典型应用是带有内存的加速器，如 GPU 和 FPGA。

CXL 将具备一致性的系统地址映射设备内存称为 HDM。HDM 与 PCIe 私有内存的核心区别在于访问机制：PCIe 私有内存需通过 MMIO（memory-mapped IO）方式访问，而 HDM 则可提供与主机内存一致的访问方式。

在 CXL 3.0 及以上版本中，CXL Type 2 设备的 DCOH 组件通过 2 种机制管理 HDM 的一致性。1）基于 CXL.cache 协议的交互。此模式下的 HDM 称为 HDM-D（device coherent）。该机制实现了基于偏置模式的一致性模型，确保设备内存映射至系统地址空间时的一致性，同时支持设备以高带宽直接访问本地内存而无需大量的一致性开销（例如主机端监听），并允许主机以统一缓存语义访问设备内存。2）基于 CXL.mem 的反向无效监听。设备通过专用 BISnp 通道直接发起对主机缓存的监听，并利用监听过滤器按缓存行粒度管理一致性。此机制会阻塞新的 M2S 请求直至 BISnp 消息被主机处理完毕，该模式下的 HDM 称为 HDM-DB（device coherent using back-invalidate）。

这 2 种方法为实现 HDM 的一致性提供了灵活的途径。然而，它们都要求在 CXL Type 2 设备上实现相应的监听及控制组件以确保一致性，因此使得该类型设备的研发变得相对复杂。

2.1.3 CXL Type 3 设备

CXL Type 3 设备支持 CXL.io 和 CXL.mem 协议，典型的应用是主机的内存扩展器。该设备主要通过 CXL.mem 运行，为主机发送的请求提供服务。CXL.io 协议主要用于设备发现、枚举、错误报告和管理。

若 CXL Type 3 设备的内存完全由主机管理，则称为 HDM-H（host-only coherent）模式。此类设备主要用于内存扩展，且无需与主机进行一致性交互管理，因此这类设备无需实现 DCOH 组件。而对于 CXL 3.0 及以上版本的 CXL Type 3 设备，若需支持 HDM-DB 模式的内存，则需在设备中集成 DCOH 组件，以负责一致性管理。

2.1.4 CXL 交换机

CXL 交换机通过扩展主机与设备的连接规模，充分发挥 CXL 协议的高带宽与低延迟优势。在 CXL 1.0/1.1 架构中，受限于直连拓扑，单个主机能连接的 CXL 设备数量有限；而 CXL 2.0 引入的单级交换机突破了这一瓶颈，不仅能够连接多个设备，还支持多个主机连接到同一台交换机，显著提升系统的扩展性。

通过支持连接多个主机的 CXL 交换机，CXL 2.0 实现了单逻辑设备（single logical device, SLD）和多逻辑设备（multi-logical device, MLD）的内存池化。该功能可将分散在不同物理设备上的内存资源整合为一个逻辑上统一的内存池。通过其集成的 Fabric 管理器，该池化内存能够供多个主机并根据工作负载需求进行灵活的分配与使用。如图 6 所示，主机和设

备侧同一个域的资源用相同的填充图案标识，CXL 2.0 交换机可以处理多个域，一个 CXL Type 3 的 MLD 设备在其 CXL 接口上最多可支持 16 个域。池化可以克服资源滞留和碎片化问题，提高资源利用效率。

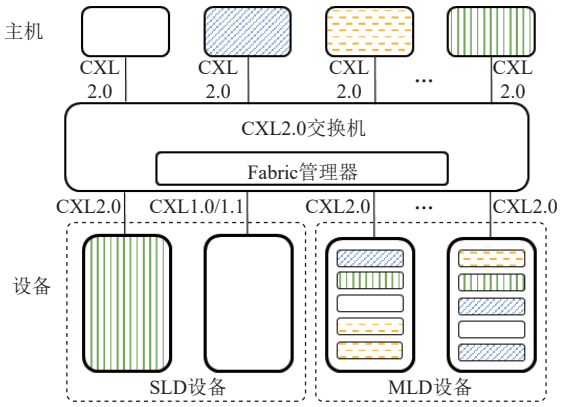


Fig. 6 Pooling of SLD and MLD based on CXL 2.0 switch

图 6 基于 CXL 2.0 交换机的 SLD 和 MLD 池化

进一步地，CXL 3.0 实现了多级交换机，并引入内存共享的概念，即允许多个主机同时访问 CXL 设备的同一块内存区域，同时保证每个主机缓存的都是共享区域数据的一致性副本。并且 CXL 3.0 实现的是硬件维护的缓存一致性，无需软件的管理协调，可以提高数据流效率和内存利用率。

图 7 展示了一个基于 CXL 3.0 交换机进行内存池化和共享的拓扑示例，其中 CXL 设备中的 S1 和 S2 是共享区域，S1 由主机 1, 3 共享，S2 由主机 2, 3, *N* 共享。

CXL 2.0 引入的单级交换机和 CXL 3.0 引入的多级交换机都可以实现主机和设备的树状拓扑结构。而 CXL 3.0 增加了一种可选的基于端口路由(port based routing, PBR)的 Fabric 功能，不同于基于层次结构路由方式(hierarchy based routing, HBR)的 PCIe 和 CXL 2.0 交换机，它可以实现非树状拓扑扩展，为构建更大规模的可扩展系统提供了可能。在 CXL 3.1 规范[10]中，进一步增强 Fabric 的功能，通过使用全局

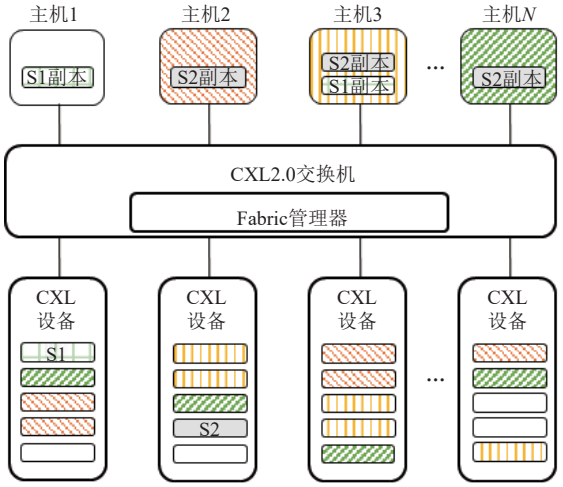


Fig. 7 Memory pooling and sharing of CXL 3.0

图 7 CXL 3.0 的内存池化和共享

集成内存(global integrated memory, GIM)支持 CXL 的 Fabric 进行主机与主机间的通信。

在设备间 P2P(peer to peer)访问方面，CXL 3.0 在 CXL.io 的基础上引入了无序 IO(unordered input/output, UIO) P2P 直接内存访问(direct memory access, DMA)功能，提升了 DMA 性能。且采用基于 PBR 的路由方式，设备能够以 P2P 方式访问网络拓扑中的任意其他设备。进一步地，CXL 3.1 的 mem 协议通道和事务中新增了对直接 P2P 的支持，使得加速器能够通过交换机以 CXL.mem 语义方式直接访问内存扩展设备。表 1 总结对比了不同版本 CXL 交换机的关键特性。

2.2 CXL 产品研发进展

在产业界，随着越来越多技术储备成熟的控制器、存储器厂商加入 CXL 联盟，近年来涌现出众多 CXL 产品，其硬件生态系统迅速发展。

2.2.1 CXL 控制器 IP

CXL 控制器 IP 实现了完整的 CXL 协议栈，支持 CXL.io, CXL.cache, CXL.mem 这 3 个子协议，负责封装、解析和传输设备间的请求、响应事务和数据。同时，它保障物理链路稳定性(如链路训练)与数据链路可靠性(如 CRC 校验、Flit 重传)，并提供设备

Table 1 Comparison of CXL 2.0, CXL 3.0 and CXL 3.1 Switches

表 1 CXL 2.0，CXL 3.0，CXL 3.1 交换机对比

特性	CXL 2.0 交换机	CXL 3.0 HBR 交换机	CXL 3.0 PBR 交换机	CXL 3.1 PBR 交换机
拓扑	单级树状	多级树状	树状/非树状/网状等	树状/非树状/网状等
连接的节点类型及个数	支持 1 个 Type 1/2 设备，多个 Type 3 设备	支持多个 Type 1/2/3 设备	支持连接 4 096 个节点	支持连接 4 096 个节点
路由方式	基于层次路由	基于层次路由	基于端口路由	基于端口路由
P2P 方式	PCIe P2P	PCIe P2P	UIO P2P	P2P CXL.mem
功能特点	支持内存池化	支持内存池化、内存共享	支持内存池化、内存共享	支持内存池化、内存共享

枚举、配置和管理等关键控制功能。表 2 中统计了目前各主要控制器厂商的 CXL 控制器 IP。其中，仅 Rambus 具备 CXL 3.1 控制器 IP，适用于 ASIC 和 FPGA 实现；具备 CXL 3.0 控制器 IP 的厂商 Cadence, Synopsys, Rapid Silicon, Alphawave Semi 实现了构建 CXL 设备、主机及交换机所需的端口逻辑，并支持 EP/RP/DM 多种模式。

2.2.2 CXL Type 3 设备产品

CXL 内存扩展控制器在单个芯片中集成了 CXL 控制器、DDR 控制器和多种外围接口，利用 CXL.mem 协议实现低延迟内存扩展，利用 CXL.io 协议实现设备管理，是内存扩展和池化的关键组件。目前最高版本有支持 CXL 2.0 的 Type 3 内存扩展控制器，其中包含产业界首款量产的澜起科技 CXL MXC（memory expander controller）M88MX5891、Microchip 的 SMC 2000 系列（16×32G, 8×32G）[20]、AsteraLabs 的 Leo 智能内存扩展控制器（最大扩展容量 2 TB）[21-22]。

CXL Type 3 设备可以为异构计算提供更高内存容量和带宽，同时具有低延迟等性能。目前产业界的 CXL Type 3 设备主要有 CXL-SSD、内存扩展器等产品，主要存储器厂商及产品特点见表 3。其中，三星电子搭载澜起科技 CXL MXC 芯片[30]，先后推出支持 CXL 1.1 的 128 GB/526 GB DRAM 内存扩展模块、2 TB 内存语义 SSD、支持 CXL 2.0 的 128 GB DRAM 内存扩展器样品。美光发布 128 GB/256 GB CZ120 CXL 2.0 内存扩展模块，已在搭载 AMD 霄龙处理器的多个平台上完成了性能验证[31]。海力士推出 96 GB DDR5 DRAM CXL 内存样品，并专门为其开发了异构存储器软件开发套件（HMSDK）。

目前产业界尚未有 CXL Type 1 或 CXL Type 2 产品，主要原因是 CXL.cache 协议以及设备 DCOH 支持逻辑的实现相对较为复杂，这点在 4.2 节 Type 1/2 硬件实现挑战中进行了详细阐述。

2.2.3 CXL 国产化产品

国产 x86 处理器厂商中科海光（Hygon）于 2025 年 5 月推出旗舰级 CPU 海光 C86-5G，实现对 CXL 2.0 协议的支持，使其在高速互连领域的技术竞争力达到国际先列，比肩同期支持 CXL 2.0 的 AMD 第 5 代 EPYC 9005 CPU（代号 Turin）及 Intel 第 6 代 GNR Xeon 6900P CPU（代号 GNR）。

曦智科技（Lightelligence）于 2023 年全球首发基于自研片间光网络（optical inter-chip networking, oNET）技术的 CXL 光互连硬件 Photowave[32]，首次通过低延迟光互连技术实现 CXL 物理层信号传输通道，突破了传统方案中铜缆传输距离有限的瓶颈。曦智科技联合新华三集团推出的 CXL-O 光互连解决方案，进

Table 2 List of CXL Controller IP

表 2 CXL 控制器 IP 列表

厂商	控制器 IP	特点
Rambus[11-12]	CXL 2.0 和 3.1 控制器	向后兼容低版本 CXL，分别支持 PCIe 5.0（32 GTps）和 PCIe 6.1（64 GTps）；提供图形用户界面（GUI）向导，允许设计人员定制 IP
Synopsys[13]	CXL 2.0 和 3.0 控制器	支持 EP/RP/DM（endpoint/root port/dual mode）模式、行业标准 AMBA 接口，分别支持 PCIe 5.0 和 PCIe 6.0
Cadence	CXL 3.1 控制器[14]	向后兼容，提供将 EP/RP/DM 控制器集成到任何片上系统中所需的逻辑
Alphawave Semi	CXL 3.0 控制器[15]	向后兼容，支持配置为设备 RP/EP 模式、交换机 UP/ DN（up/down）模式
Rapid Silicon	CXL 3.0 控制器[16]	支持配置为设备 RP/EP 模式、交换机 UP/ DN 模式[17]。提供将控制器集成到任何片上系统、FPGA 或自定义加速器 IP 中所需的逻辑
Intel	CXL 2.0 控制器硬 IP[18]	FPGA R-Tile 中提供，搭载于 Intel Agilex 7 特定产品中。支持全带宽 PCIe 5.0×16 配置，适用于 CXL Type 1，CXL Type 2，CXL Type 3 设备[19]

Table 3 List of CXL Type 3 Device Products

表 3 CXL Type 3 设备产品列表

厂商	内存扩展产品	特点
三星电子（Samsung）	128 GB CXL 1.1 DRAM[23]	采用 FPGA CXL 控制器构建，业界首款基于 DRAM、支持 CXL 的内存扩展器，支持 DDR5，配备 FPGA 控制器
	512 GB CXL 1.1 DRAM[24]	采用 ASIC CXL 控制器构建，是首款配备 512 GB DDR5 DRAM、支持 PCIe 5.0 的内存扩展设备
	2 TB Memory-Semantic SSD[25]	支持 PCIe 5.0、CXL，结合 NAND 闪存的大容量、非易失的优势以及 DRAM 的高性能优势；设备可分为内存区域和 SSD 区域双模式操作，以提供更大的灵活性
	128 GB CXL 2.0 DRAM[26]	搭载澜起科技 CXL MXC[27]，首款支持 CXL 2.0 的 128 GB DRAM
美光（Micron）	128 GB/256 GB CZ120 CXL 2.0 DRAM [28]	搭载 MicroChip SMC 2000 CXL 内存控制器。支持 PCIe 5.0×8，内存读/写带宽高达 36 GBps
海力士（SK Hynix）	96 GB CXL 2.0 DDR5 DRAM[29]	搭载澜起科技 CXL MXC，支持 CXL 2.0 和 PCIe 5.0×8，使用 DDR5 标准 DRAM

一步为数据中心资源池化提供了可横向扩展的国产化基础架构。

澜起科技(Montage Technology)推出全球首款CXL内存扩展控制器芯片(MXC),在CXL内存控制芯片领域处于行业领先地位。MXC芯片支持CXL 1.1/2.0,兼容DDR4和DDR5标准,被三星电子和海力士成功用于DRAM产品研制中。2025年1月澜起持续推出了国产PCIe 6.x/CXL 3.x Retimer产品[33],传输速率达64 GTps,相较于PCIe 5.0提升1倍。

国数集联(Clussys)基于FPGA成功推出业界首款CXL多级网络交换机,可通过高速铜缆直接实现机柜内及相邻机柜间的网络互联,无需额外配置网卡或光模块,提供低延时、低成本的高性能网络,支持CPU, GPU, DDR, SSD, FPGA等资源扩展。

服务器层面,浪潮元脑NF5280G8 CXL服务器[34]主机侧支持CXL 2.0,配合AIC CXL内存扩展设备最高可额外扩展24条DDR4/DDR5内存,实现了内存池化和资源共享;联想ThinkSystem SR860 V3服务器[35]利用Astera Labs Leo CXL内存控制器实现内存扩展,可提供16 TB的DDR5内存。阿里云全球首创基于CXL交换机的分布式内存池架构PolarCXLMem[36],将跨机交互延迟缩短至百纳秒级别,并宣布将于2025年下半年发布全球首款基于CXL交换机的数据库专用服务器。

国内已形成从处理器、内存扩展设备、CXL交换机到系统解决方案的完整CXL产品研发链条,在硬件研制和系统集成上逐步展现出竞争力,为CXL技术演进和数据中心资源池化与高效利用奠定了坚实的国产化基础。

2.3 CXL设备的性能特点

2.3.1 CXL真实物理设备的性能特点

CXL官方Spec文档[10]的合规性测试中给出了CXL.mem及CXL.cache子协议中典型事务收发的推荐时延,即设备收到事务消息后经过处理再发出事务消息的时间间隔,如表4所示。由于CXL Type 1/2设备中需DCOH组件维护主机与设备间的缓存一致性,因此CXL.cache协议时延整体上比CXL.mem的协议时延高。

Table 4 Recommended Latency Targets for CXL Protocol Transactions[10]

表4 CXL协议事务的推荐时延目标[10]

场景	组件	协议	接收消息	发送消息	目标时延/ns
1	CXL Type 1/2	CXL.cache	H2D Req Snoop(未命中情况)	D2H Resp Snoop Response	90~150
2			H2D Resp WritePull	D2H Data	65
3	CXL Type 3(DDR)	CXL.mem	M2S Req MemRd(读内存)	S2M DRS MemData	80
4			M2S RwD MemWr(写内存)	S2M NDR Cmp	40
5	Host	CXL.mem	S2M BISnp	M2S BIRsp	90

在CXL设备上实际运行程序时,真实性能与规范中的参考时延存在差异。当主机访问CXL Type 3设备内存时,除了CXL协议处理时延外,还包括主机侧LLC处理时延与内存控制器的时延。文献[37]中对CXL内存访问链路上的每个模块给出了理论上的时延估计,如图8所示,CXL接口协议处理时延通常为25 ns,内存控制器时延大约为45 ns,CPU侧LLC处理时延为40 ns,理想情况下整体的访问时延约为140 ns。

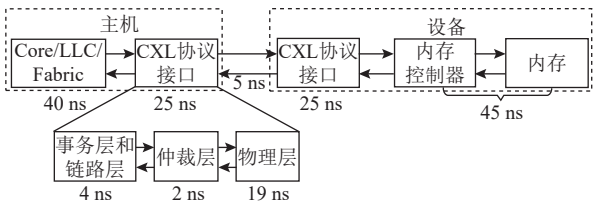


Fig. 8 Latency theory estimation of each module in CXL protocol[37]

图8 CXL协议各个模块的时延理论估计[37]

此外,实际应用中多项因素可能影响CXL设备内存访问性能。例如:1)设备通过CXL交换机连接至主机,或者在CXL设备侧部署访问控制模块,均会引入额外的通信跳数或处理开销,导致整体的访问时延增加。2)内存控制器设计也会影响CXL内存访问的性能,如文献[38]提到不同的内存读写比例下,内存控制器中读写交织(interleaving)功能的高效实现能够显著提升系统的内存带宽。3)不同内存访问操作类型,如顺序读写或者随机读写方式、对CXL内存不同读写比例,均会对实际性能产生不同程度的影响。

总体上,影响CXL内存访问性能的因素主要包括3点:1)CXL协议接口包括主机侧和设备侧的处理效率;2)内存控制器如引入读写交织、访问控制等的实现效率;3)缓存未命中的情况下,主机侧发起的

内存访问请求转化为 CXL 内存请求的效率。

相关研究[38-40]对真实 CXL 设备上内存的性能特点进行了测试和分析。文献 [38] 对 CXL 的真实系统性能进行了全面评估，测试平台包括支持 CXL 的 Intel 4th Xeon CPU 和 3 个来自不同制造商的 CXL IP（涵盖硬 IP 和软 IP）。研究针对基于 CXL IP 的扩展内存，分别测试了不同内存访问操作下的时延和带宽，结果如表 5 所示，测试结果表明，与基于 FPGA 的 CXL 软 IP 实现相比，ASIC 实现的硬 IP 具有更高的性能，但都比主机本地内存的性能低。值得注意的是，在读写操作比例为 2∶1 时，采用硬 IP+DDR5 的 CXL 设备带宽超过了访问远端 NUMA（non-uniform memory access）节点 DDR 内存，这是因为该设备的内存控制器可以更高效地处理交织内存的读写。

Table 5 Performance of Different CXL IP in CXL Devices[38]

表 5 CXL 设备不同 CXL IP 的性能[38]

IP 类型+内存介质	时延（随机访问）	带宽（全部为读操作）	带宽（读写比例为 2∶1）
硬 IP+DDR5	2	0.45	0.64
软 IP+DDR5	5	0.2	0.21
硬 IP+DDR4	3	0.47	0.4
远端内存 DDR5	1.5	0.7	0.41
本地内存 DDR5	1	1	1

注：表中数值为相对于本地 DDR5 内存性能的倍数。

同时，文献 [38] 给出了采用 4 种不同的读写访问下相对于本地 DDR 内存的时延性能，包括非瞬时读取旁路缓存、非瞬时写入旁路缓存。整体上，远端 DDR 内存的时延更低，但在非瞬时读取操作下，由于 CXL 内存相比远端 DDR 内存有更短时延的缓存一致性检查，远端内存访问的时延性能比硬 IP+DDR5 的时延高。

文献 [39] 使用基于 ASIC 实现的 CXL 内存设备，在多 Socket 系统中测试了本地与远端 CXL 内存的访问性能，并分析了不同读写比例下的时延带宽性能特点。结果如表 6 所示，通常情况下全部为读操作读写比例为 1∶0 的时延最低，跨 Socket 的远端内存访问由于需经过 CPU 间的 UPI（ultra path interconnect）链路，导致时延相对更高。CXL 内存的最大带宽并未出现在全部为读操作场景下，原因在于全部为读操作的场景未能充分利用 PCIe 的双向传输能力。整体上，跨 Socket 的远端 CXL 内存由于访问路径长，时延带宽性能均不如本地 CXL 内存，因此应尽量避免。

Table 6 Performance of CXL Memory in Multi-Socket System[39]

表 6 多 Socket 系统中的 CXL 内存性能[39]

内存类型	最小时延/ns	最大带宽/GBps
本地主存（MMEM）	97（读写比例为 1∶0）	67（读写比例为 1∶0）
远端主存（MMEM-r）	130（读写比例为 1∶0）	72（读写比例为 1∶0）
本地 CXL 内存（CXL）	250.42（读写比例为 1∶0）	56.7（读写比例为 2∶1）
远端 CXL 内存（CXL-r）	485（读写比例为 1∶0）	20.4（读写比例为 2∶1）

与 CXL Type 3 设备不同，CXL Type 2 设备可以访问主机，缓存一致性的维护同时涉及到主机与设备，因此其性能特点分析相对更加复杂。文献 [41] 给出了真实 CXL Type 2 设备的性能测试数据，不同偏置模式下设备访问 HDM-D 内存时延如表 7 所示。整体上，在设备偏置模式下，设备访问自身内存的时延均更低，原因在于设备偏置模式不会带来缓存一致性管理的性能开销。此外，在命中设备缓存的情况下，2 种偏置模式下的内存读操作性能相近，而设备偏置下的内存写时延比主机偏置下的时延低 60%。因为主机偏置模式下，内存写需要执行对主机 LLC 中的缓存一致性状态检查，并无效掉 LLC 中相应的缓存行，增加了相应的一致性维护开销。

Table 7 Latency Performance of Device Accessing It's Own Memory[41]

表 7 设备访问自身内存时延性能[41]

是否命中设备缓存	内存读（不缓存）	内存读（缓存）	内存写（不缓存）	内存写（缓存）
命中	1	1	0.4	0.4
未命中	0.6	0.6	0.4	0.4

注：表中数值为设备偏置相对主机偏置下的时延比值。

此外，在文献 [41] 的测试结果中，主机对设备内存的写请求下，命中设备缓存（且为独占态）情况下时延比未命中的时延增加了 10%。原因在于文献 [41] 中设备实现的 DCOH 缓存是设备私有缓存，只支持设备计算单元对其读写，不支持主机直接从其中读取数据，所以即便主机请求命中设备缓存，DCOH 仍然需要访问设备内存。而另一方面，当命中设备的缓存行状态为独占态时，需要将缓存行状态切换并维护一致性，导致此时时延更高。由此可以看出，增加的 10% 的时延正是主机与设备缓存一致性维护的开销。

文献 [40] 提出了一个 CXL 内存性能特征的分析框架，用于完整地分析、建模和优化 CXL 内存性能，

该框架基于大量的实验评估揭示了多项关键发现，涵盖工作负载性能特征分析、亚微秒级的内存时延、CXL 尾时延特征、CPU 对 CXL 时延的容忍度、CXL 性能的根因分析以及高精度的性能预测模型。基于这些分析对比，文献 [40] 进一步提出了更优的交织策略以及高效的内存分层数据迁移策略。

目前对 CXL 设备性能的关注点主要集中在设备侧，如设备中内存控制器和 CXL IP 的性能。实际上在主机侧，同样需要针对 CXL 内存这样的远端长延迟内存进行同步优化[42]。

2.3.2 CXL 设备在应用场景中的性能特点

CXL 设备内存可以扩展主机系统的内存容量和带宽，但其时延和带宽性能不及主存，也给实际的应用带来了影响。目前有很多工作研究分析了在一些应用场景中，如高性能计算[43-45]、内存数据库[46-51]、云计算[52-53]、AI 模型计算[39,54]，CXL 内存作为扩展内存的性能特点。

在高性能计算领域，文献 [44] 的研究结果表明，当 CXL 内存占比达 75% 时，对高性能计算任务（尤其带宽敏感型应用如 OpenFOAM，Hypre）的性能影响可忽略，其带宽表现与本地主存持平。Fridman 等人[45]对 CXL 内存作为持久内存解决方案的可行性进行了探索，在持久内存的基准测试中，CXL 内存表现出优异的性能。Ryu 等人[46]运行 Spark 计算任务时使用 CXL 内存替换现有硬盘存储，带来明显的性能提升。

在数据库领域，文献 [46,48] 在 CXL 设备原型的扩展内存上使用了通用数据库基准测试，结果展现了相比于服务器横向扩展，CXL 扩展内存应用于内存数据库场景时能够提升性能并降低成本。文献 [49] 探讨了低延时、高带宽的 CXL 内存给数据库系统带来的机遇与挑战。文献 [50] 详细地调研了 CXL 内存给数据库中的具体场景带来的性能影响，发现其性能取决于应用中数据结构的具体访问模式，比如在联机事务处理（on-line transaction processing，OLTP）任务中性能下降很小，而联机分析处理（on-line analytic processing，OLAP）任务中性能下降相对明显，但 CXL 内存对失效后的重启效率的提升有积极作用。文献 [55] 提出了一种针对事务数据的松散一致性的新型混合一致性原语，可以有选择地实现数据一致性，在数据库的 OLTP 工作负载中，CXL 相比当前基于网络的系统性能更优，并且在通用事务处理上实现了比普通 CXL 内存架构更高的吞吐量。

在云计算领域，数据中心的最大成本是 DRAM 内存，由于内存需求扩大，成本将继续增长[37,56]。Yang 等人[52]研究了将 CXL-DRAM 内存与成本更低的 SSD 硬盘结合的混合内存池架构对应用的性能影响，通过在混合内存池中运行深度学习训练、数据库、数据分析和视频处理等多种代表性的工作负载，验证了该方案在不牺牲应用程序性能的基础上可以降低云服务器的总体成本。由于 CXL 内存接口相对于 DDR5 接口的引脚有效性更高，文献 [53] 探索了在服务器内使用 CXL 内存全面代替 DDR 内存以解决多核处理器扩展带来的内存带宽不足的问题，并论证了尽管空载时延较高，但该方案在带负载的应用场景下依然能够有效降低时延。

在 AI 模型计算领域，深度学习往往需要大量内存来存储数据集。然而现有的内存系统成本高且基于低成本磁盘扩展内存的方案耗时较长。Arif 等人[54]基于 CXL 内存为深度学习任务提供额外的内存和快速暂存空间，来缓解在训练过程中输入训练数据和模型参数被交换到慢速存储层而造成总体训练时间长的情况。文献 [39] 通过实验验证了 CXL 内存对大模型推理服务的性能影响，结果表明在主存与 CXL 内存交织配置下运行的大语言模型（large language model，LLM）推理服务，其性能显著高于仅使用主存的方案；且随线程数增加，CXL 内存的高带宽优势带来的性能提升愈加显著。文献 [57] 提出使用基于 CXL 大内存的 KV cache 存储用于大模型的推理服务，在提升 30% 的 Batch Size 的同时降低 87% 的 GPU 成本。

此外，CXL 内存不仅可以扩展主机内存，也可以扩展 GPU 显存，有部分研究工作[58-61]尝试在 GPU 设备上支持 CXL 以缓解大模型计算中 GPU 显存不足的问题。文献 [58] 在 CXL-GPU 上设计了多个 CXL 的根端口以连接多种存储介质，并在 RTL 层实现了 CXL 控制器，其引脚间的时延为几十纳秒级，扩充了 GPU 上可高效访问的内存容量。文献 [59] 在多 GPU 系统内利用 CXL 内存扩充 GPU 内存容量的场景中，通过减少 CXL 内存的争用现象，提升了系统整体的吞吐率。文献 [60] 探讨了 NVIDIA GPU 的统一内存框架中引入 CXL 内存的可能性，在软件层的 GPU 内存管理上引入了 CXL 内存错误的处理机制。文献 [61] 在使用 GPU 处理内存密集型的图计算应用时，通过引入 CXL 外部内存提升了应用性能。

整体而言，CXL Type 3 设备内存的优势可以概括如下：

1）扩展主机系统或者加速器设备的内存容量和带宽。

2)相较对于跨网络访问的横向扩展(scale-out)远端内存或者本地硬盘性能更优。

3)通过替换云服务商的部分 DDR 内存，有效降低总体成本。

基于上述场景的应用，均可采用 CXL 内存设计性能优化方案。

3 CXL Type 3 设备的研究进展

CXL Type 3 设备内存(简称 CXL 内存)与主机内存不同，其通过 PCIe 外围总线连接到 CPU，相比传统 DDR 接口的紧耦合设计，其内存扩展更加灵活、成本更低，且通过 CXL 交换机可以实现更大规模扩展，显著提升系统整体的内存容量和带宽。然而，CXL 内存的时延高于主存，性能存在劣势，因此，需根据应用场景特点对 CXL 内存进行合理有效的利用。

3.1 节将首先介绍当前关于 CXL 内存管理的研究进展。鉴于 CXL 真实设备尚未普及，3.2 节将系统性地梳理该领域研究所采用的实验方案，并评估其成本与难度，为后续研究提供参考。

3.1 CXL 内存管理研究进展

围绕 CXL 内存带来的新特点，近几年来学术界已开展了一系列的相关研究，通过对这些研究的整理和分析，总结了如下 5 类应用研究方向：

1)CXL 扩展内存的管理优化[38,62-68]；

2)基于 CXL 的内存分离与内存池化[37,69-73]；

3)基于 CXL 3.0 的内存共享[74-79]；

4)基于 CXL 的大容量内存设备[80-85]；

5)基于 CXL 内存的近数据处理(near data processing, NDP)[65,86-93]。

图 9 更加详细地给出了应用研究方向之间的关系，本地主机侧可以直接通过 CXL 扩展内存，进一步，可以通过 CXL 交换机或者以太网方案连接到远端的 CXL 内存，远端 CXL 内存设备中可以支持多种存储介质以实现大内存设备，并集成计算单元以实现近数据处理。具体地：1)合理设计策略，既能利用 CXL 内存带来的容量与带宽扩展，又能避免 CXL 内存的性能劣势。2)利用 CXL 的独立性与可扩展性实现内存分离架构，替换现有基于网络的内存分离方案，以提升远端内存访问效率。进一步可构建更加高效的内存池化系统，提高云计算场景下的内存利用率，降低整体的总拥有成本(total cost of ownership, TCO)。3)利用 CXL 3.0 引入的多主机内存共享替换现有基于网络的分布式内存共享方案，提供更高效的共享内存访问方式。4)利用 CXL 协议接口可支持多种存储介质的特点引进新型大容量存储介质，比如持久内存、闪存(flash memory)等，在设备控制器层面进行优化设计，如内存寻址方式的转变、访存效率的优化等，以实现大容量内存设备的高效访问。5)CXL 设备端集成计算单元，支持近数据处理，可针对性地优化上层应用的性能，实现内存密集型任务的性能提升。

3.1.1 CXL 扩展内存管理优化

CPU 能够以相同的方式访问主存和 CXL 内存，但 CXL 内存与主存存在明显的差异，其容量更大，可达数百 GB 至 TB，但性能更差，访问时延是主存的 2~3 倍[38]，因此重构了存储层次结构，如图 10 所示。在 CXL 内存上运行应用需考虑其带来的性能影响，比如时延敏感型的应用直接在 CXL 内存上运行可能会导致性能下降。文献 [65] 中提到了基于冷热数据调度对 CXL 内存进行分层管理的必要性，其在 CXL 内存设备控制器中实现页面冷热程度的统计功能，

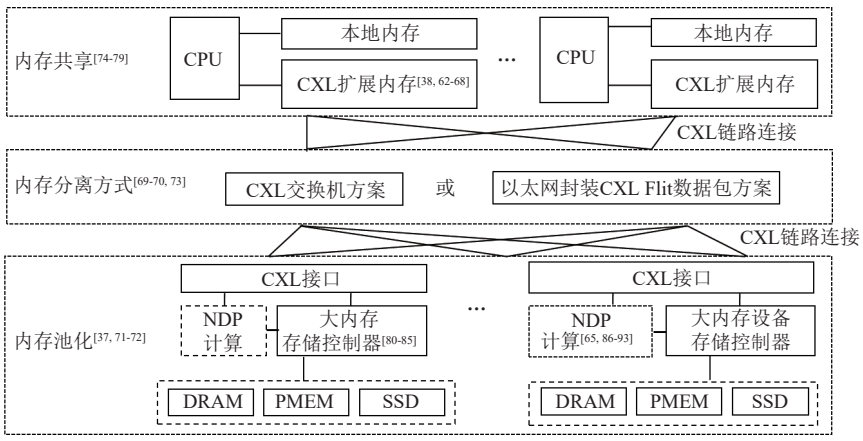


Fig. 9 Summarization of research progress of CXL Type 3 device applications

图 9 CXL Type 3 设备应用研究进展总结

可以更高效地完成页面统计与页面迁移。整体上，为优化 CXL 内存的管理与使用，需研究适配其特性的数据冷热感知及交换策略。

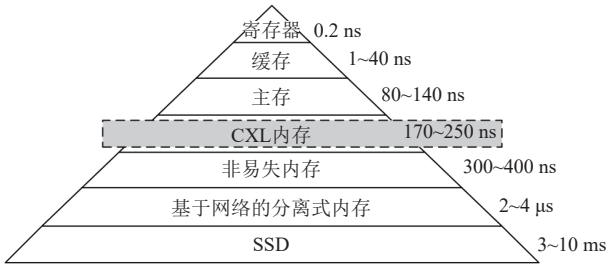


Fig. 10 Memory hierarchy structure based on CXL memory[62]

图 10 基于 CXL 内存的存储层次结构[62]

在 CXL 内存管理的软件运行环境上，三星的 Lee 等人[63]设计实现软件定义的内存分层系统，其在操作系统内核态和用户态分别定义一组基于分层内存的内存分配软件库，包括异构内存分层、管理和智能的内存分配，方便上层应用程序根据自身需求动态灵活地分配主机内存和 CXL 内存，使得应用程序的性能波动不受底层异构内存性能差异的影响。

文献 [38] 提出了一种 CXL 内存感知的页面分配策略 Caption，以更有效地利用 CXL 内存作为带宽扩展器。该方法考虑多种因素以决定分配在主机内存和 CXL 内存上的页面的最佳比例。在多种应用上的实验表明，Caption 可以将分配到 CXL 内存的页面比例自动收敛到一个经验上可提升应用带宽性能的值。

由于 CXL 内存时延比主存高 2~3 倍，传统多 NUMA 系统下的 NUMA-Balancing 与 Auto-NUMA 等机制无法有效适应 CXL 内存的长延迟特点。Maruf 等人[62]提出了基于 CXL 内存的透明页面交换策略 TPP，在程序运行过程中，通过及时精准地识别主机内存中的冷页面和 CXL 内存中的热页面，有效地将 CXL 内存中的热页面换入到主机内存中，同时将主机内存中的冷页面及时换出到 CXL 内存，在扩展了内存容量带宽的同时保持了与主存相当的性能，目前该策略已更新至 5.19 以上版本的 Linux 内核程序中。

CXL 内存在主机操作系统上通常被映射为无 CPU 的内存 NUMA 节点，而传统的 NUMA 节点间的内存交织方法可能会因 CXL 内存的长延迟导致性能下降。Lee 等人[66]提出了一种新型的内存交织策略 T-CAT，将 LLC 分成 2 块区域，分别对应近端内存和远端内存，通过动态缓存管理机制，减少近端内存和远端内存的平均访问时延上的差异，缓解访问远端 CXL 内存造成的性能下降，提高 NUMA 交织的内存层次系统的性能。

文献 [67] 在设备侧实现了 CXL 内存页面冷热程度的精准统计，相较于基于软件的感知方式，可以更好地支持 CXL 内存的冷热调度决策，实现 CXL 原生的分层内存的高效管理。文献 [68] 提出了一种 CXL 驱动的性能分析方案以准确和透明地统计 CXL 内存中 4 KB 页面与 64 B 字数据的访问次数，结果显示：CPU 驱动的页面感知策略容易把温页面识别成热页面；部分应用的热页面中，只有少量的 64 B 数据被频繁访问；同时，识别热页面的开销会影响应用的性能。因此，作者提出在 CXL 控制器中实现基于硬件的热页面和热字的追踪器，提高了 CXL 内存热页面识别准确率，基于此追踪器的页面迁移相比于 CPU 驱动的方案性能提升了 14%。

表 8 总结对比了现有的 CXL 扩展内存管理方案，根据管理策略的实现位置分为 2 种：第 1 种将 CXL 内存感知与调度在主机系统软件层实现，虽然针对 CXL 内存的特点进行了适配，但存在页面冷热程度统计不准确、开销大的问题，同时周期性地统计内存访问频次会影响 CPU 性能。第 2 种是将 CXL 内存页面冷热程度感知的策略卸载至设备侧，不影响 CPU 性能，同时更精准地感知 CXL 内存的页面冷热程度，可以支持更复杂的统计策略。

Table 8 Comparison of CXL Memory Management Schemes

表 8 CXL 内存管理方案对比

CXL 内存管理方案	文献	特点
操作系统软件侧	[62−63,66]	CXL 内存的页面冷热程度统计不准确、开销大，影响 CPU 性能
CXL 设备硬件侧	[65,67−68]	CXL 内存的页面冷热程度感知更精准，可以支持更复杂的策略，不影响 CPU 性能

3.1.2 基于 CXL 的内存分离与内存池化

内存分离[94-96]架构解耦内存与计算，使系统设计者能够独立于服务器，灵活扩展内存资源，同时精准适配应用负载需求。这种硬件分离方法既可以节省系统的管理成本，又可以提高内存资源的利用率。

传统内存分离通常使用 RDMA 访问远端内存，其通过内核旁路和零拷贝技术减少内存访问路径长度、降低软件开销以提供高吞吐量和低延迟的远端内存访问性能。但通过网络访问远端内存的延迟通常在微秒级，相对于本地内存来说，性能仍然存在较大差距，且当缓冲区满时，RDMA 的内存访问延迟更高[97]。

相较于 RDMA, CXL 总线互连具备更低延迟, 且提供了 load/store 指令提供缓存一致性的的远端内存访问能力, 有望将内存访问延迟降至纳秒级。当前基于 CXL 的内存分离架构解决方案主要包括 3 类: 基于 CXL 交换机的实现(DIRECTCXL)[69]、基于网络的实现(CXL over Ethernet)[70] 以及两者的结合方案(Rcmp)[73]。表 9 对比了现有内存分离方案的性能特征。

Table 9 Comparison of Different Memory Disaggregation Solutions

表 9 不同的内存分离方案对比

内存分离方案	内存访问性能	扩展能力	具备一致性
RDMA	微秒级	机架间	否
DIRECTCXL	纳秒级	机架内	是
CXL over Ethernet	微秒级，缓冲区命中时更低	机架间	是
Rcmp	机架内纳秒级，机架间微秒级	机架间	是

Gouk 等人[69] 提出了基于 CXL 的新型内存分离系统 DIRECTCXL, 首次实现了 CXL 2.0 交换机通过 CXL 内存协议(CXL.mem)直接连接主机处理器和远程内存资源。相较于基于 IB 网络的 RDMA 方案, CXL 分离架构因减少 1 次内存拷贝, 内存访问性能更高。当工作负载所需的数据可以命中主机处理器缓存时, DIRECTCXL 可实现近 DRAM 级别的性能。在实际的应用场景(如深度学习推荐模型和图分析任务), 其平均性能较 RDMA 方案提升显著。

由于 CXL 是总线级别的协议, 信号有效传输的距离有限, 分离扩展规模受限。文献 [70] 中提出 CXL over Ethernet 架构, 通过在以太网帧中封装 CXL Flit 数据包, 实现跨网络远端内存的一致性访问, 同时为避免频繁网络访问造成的性能瓶颈, 在主机侧设计了 SRAM 缓冲区以提升平均访问性能。文献 [73] 中提出了一种机架内使用 CXL、机架间使用 RDMA 的 Rcmp 内存分离架构, 通过利用 CXL 特性增强了以 RDMA 为基础的性能, 解决了软件层面 RDMA 与 CXL 在数据粒度、通信模式及性能等方面不匹配的问题。

CXL 2.0 及以上版本中引入了 CXL 交换机, 支持内存池化的功能, 同时也引入了 SLD 与 MLD 的逻辑设备概念, 支持内存资源细粒度的切分。现有的软件池化方案由于处理页面错误[78,98-99] 或更改虚拟机(virtual machine, VM)客户端[79,100], 池化内存访问时延在微秒级。通过基于 CXL 硬件级的内存分离来构建内存池, 可避免软件池化方案中的问题, 提升池化内存访问效率。

Pond[37] 率先系统探索了基于 CXL 的内存池化架构, 通过实验验证了拓扑结构、容量规模与主机数量对池化内存性能的影响。结果表明, 8~16 个主机构建的池化内存系统可满足多数任务的低延迟需求, 同时降低 7% 的内存成本。为评估生产环境效果, 其采用 zNUMA 节点模拟 CXL 内存, 并基于应用延迟敏感度建立机器学习模型, 精确预测虚拟机上本地内存和池化内存的分配比例, 实现与同一 NUMA 节点内存性能相近的表现。

内存池化系统的设计不仅可以通过 CXL 交换机来实现, 也可以通过多头(multi-headed)CXL 设备[10] 来实现, 多头 CXL 设备通常是指带有多个 CXL 接口的 Type 3 设备, 如图 11 设备中的多个 CXL EP 接口。

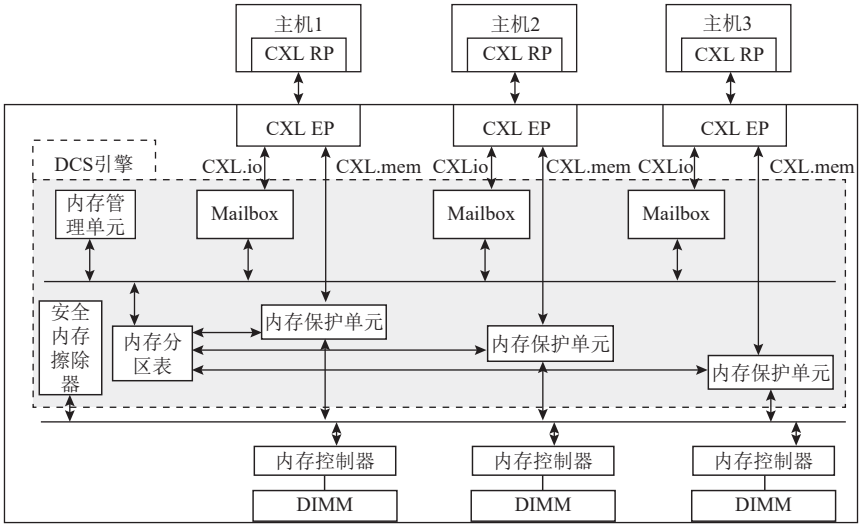


Fig. 11 DCS hardware architecture based on multi-headed CXL device memory pooling[71]

图 11 基于多头 CXL 设备内存池化的 DCS 硬件架构[71]

Ha 等人[71]提出了一个基于多头设备的内存池化系统原型，在多头 CXL 设备中实现了一套动态容量服务(dynamic capacity service, DCS)引擎，如图 11 所示。该引擎中的核心组件是内存管理单元，其通过 Mailbox 获得池化内存申请或者释放的请求，为了处理多个主机的内存请求，重新定义了新的 DCS 命令集来管理内存分区表中内存的状态。此外，Ha 等人[71]开发了完整的动态内存分配软件接口，能够使得应用无感知地使用池化内存，减少池化内存的使用冲突，提高池化内存整体的利用率，并在实际的应用场景中证实了 DCS 引擎有效保持了 CXL 内存池的高利用率。

文献 [72] 考虑了在云服务中使用软件管理池化的 CXL 内存所带来的开销大的问题，引入了 Intel 为管理 CXL 内存设计的 Flat Memory Mode 功能，并对其性能进行了原型验证。整体上虽然多达 82% 的任务负载只受到了 5% 的性能损失，但仍然有部分任务由于内存争用问题面临 34% 的性能损失，因此文献 [72] 进一步提出了轻量级的多用户内存分配器，避免了多虚拟机之间的内存争用问题。

3.1.3 基于 CXL 3.0 的内存共享

CXL 3.0 提供了内存的多主机共享功能，多个研究工作[75-77]将其应用在分布式共享内存领域，以提升现有基于网络的分布式共享内存的访问性能。文献 [75] 提出了 CXL-oF(CXL over fabric)的架构，以解决大规模图计算领域中分布式共享内存软件开销大及传输粒度粗导致的性能问题，特别是针对图计算集群中出现的远程高频内存访问严重的现象。结合图计算过程中的内存访问特性和 CXL.mem 的内存语义特性，通过在 Ethernet 帧数据中封装 CXL.mem 协议的 Flit 数据包，利用 CXL 协议中硬件级别的数据传输初始化和一致性维护，提供远端内存按字节寻址的细粒度访问功能，提升数据传输的效率，从而加速图处理过程。文献 [76] 中提出基于 CXL 互连构建的分布式共享内存系统 Yggdrasil，其创新软件栈突破传统网络套接字通信局限，充分利用 CXL 内存语义特性，使上层应用透明高效地使用 CXL 分布式共享内存。文献 [77] 分析了在分布式系统中基于 CXL 共享内存优化远端过程调用(remote procedure call, RPC)的可行性与期望收益。文献 [78] 针对 CXL 的分布式共享内存系统设计了一个基于引用计数自动化的分布式内存管理系统 CXL-SHM，以减少回收远程内存所涉及的手动工作量。

文献 [74] 提出了基于 CXL 共享内存的远程进程拷贝接口 CXLfork，通过支持 CXL 共享内存上检查点状态的可控去重，实现本地内存与 CXL 内存间的精细分级调度控制。CXLfork 在显著降低内存占用的同时，最小化冷启动开销，其性能超越现有常规方案和前沿技术，实现近乎零序列化、零拷贝的集群级进程克隆技术。

虽然 CXL3.0 协议中新增了内存被多个计算节点共享的功能，但是当某个节点需要独占式访问时，需要无效掉所有其他节点的数据，产生了大量开销。文献 [79] 提出支持高效内存共享的 CXL 分离式内存系统，采用新型的缓存一致性控制流，利用 CXL.io 协议中保留的消息字段管理远端共享内存资源，不影响 CXL.cache 和 CXL.mem 协议的原有功能，同时，内存请求被设计为推测性地在检查权限标签之前进行处理，因此计算节点不需要等待权限标记，有效提升了共享内存的访问效率。

3.1.4 基于 CXL 的大内存设备

在 CXL 协议规范中没有限制存储介质的类型，因此 CXL 可连接大容量存储介质，如 NAND Flash，以实现具有内存语义的大内存设备。

在学术界，Jung 等人[80]首次尝试将 PCIe 连接的 SSD 的块语义访问通过 CXL 进行拓展以支持内存语义访问。Kwon 等人[81]进一步为 CXL-SSD 提出了高效的预取器 ExPAND，其将 CPU 侧的 LLC 预取卸载到计算能力和存储能力更强的 CXL-SSD 上，通过 CXL.mem 的反向无效监听机制来维护 LLC 与 CXL-SSD 之间的数据一致性，并采用了基于机器学习的预测算法替换原有的基于规则的预取机制，以提升预取机制的准确性。与原有基于规则的预取机制相比，ExPAND 能提供精确的预取时机的估计，提升 CXL-SSD 上数据的访问性能。更进一步，Yang 等人[82]探索了基于 CXL 架构部署廉价闪存设备以应对“内存墙”问题的可行性，文献 [82] 分析了真实的应用运行过程中现有虚拟内存地址转换技术对于 CXL 闪存预取的局限性，并提出传递内核级访问模式的提示方式，使基于 CXL 的闪存提升至接近 DRAM 的性能级别。

在工业界，三星发布了内存语义的 SSD 控制器设计方案[83]，如图 12 所示。其中使用 CXL.io 协议读写(read/write 指令)基于块地址寻址的数据，使用 CXL.mem 协议读写(load/store 指令)字节寻址的数据，从而实现对 SSD 的具有内存语义的访问。

文献 [84] 针对这种支持双模式接口的 CXL-SSD 设备设计了一种新的可感知到 CXL 特性的文件系

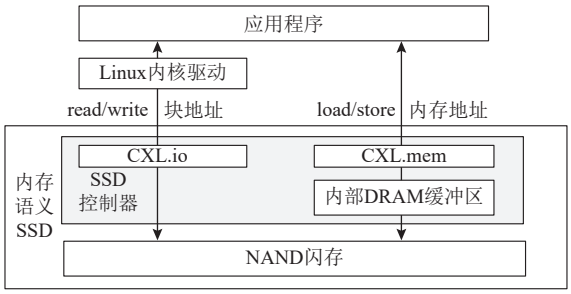


Fig. 12 SSD controller architecture in Sumsung's memory semantic[101]

图 12 三星内存语义的 SSD 控制器架构[101]

统 RomeFS, 并验证了其在不同访问模式下的性能优势。文献 [85] 创建了适配 CXL-SSD 的 ByteFS 文件系统和相应的固件, 以更好地利用 CXL-LinuxSSD 所提供的 2 种语义, ByteFS 既支持字节粒度以保持 SSD 数据的持久性, 同时扩展了文件系统中核心的数据结构, 使其能够支持字节或块语义的访问。为了提升字节粒度的写操作性能, ByteFS 以一种日志结构的方式管理 SSD 内部的 DRAM, 支持数据的合并以减少不必要的 IO 流量, 也支持数据在主机的页面缓存与 SSD 的缓存之间的协同, 以更好地利用宝贵的内存资源。

3.1.5 基于 CXL 内存的近数据处理

CXL 设备与 CPU 解耦的特点可以支持在 CXL 内存设备中集成计算单元以实现近数据处理, 在基于内存扩展的前提下提升数据处理的效率, 满足数据密集型应用的计算需求。

虽然 CXL 相比于 PCIe 4.0 提供了更大的互连带宽, 但与内存带宽相比仍然不足, 尤其是当 CXL 设备连接多个内存条时, 设备内部内存的带宽将远大于 CXL 总线的物理带宽上限, 从而造成主机与 CXL 设备间数据传输时的带宽瓶颈。为了解决此挑战, 文献 [86] 中提出在 CXL 内存中集成近数据处理模块, 在设备侧进行计算以最大化内部带宽的使用, 只传输处理后的结果以减少通过 CXL 链路的数据量。

文献 [87] 中提出了在 CXL 内存池上实现近数据处理的架构 BEACON, 用以加速基因数据分析应用, 为此, 设计了 CXL 内存的管理优化框架, 包括内存分配和地址映射, 以实现有效的按需内存分配与使用, 此外, 结合基因分析应用的特点进行多芯片合并实现进一步的通信优化, 以解决现有基于 DIMM 近数据处理方法的通信带宽和内存扩展瓶颈。

CXL-ANNS 是由 Jang 等人[88] 提出的基于 CXL 内存池来加速大规模图上的最近邻搜索的方法。通过 CXL 内存池存储大规模图数据, 同时根据最近邻搜索的特点, 采用本地内存存储频繁访问的邻居节点, 基于图遍历的行为预取最有可能即将被访问的节点, 以提升访问效率; 此外, 通过将最近邻搜索拆分成更小的子任务, 在 CXL 设备中集成计算单元 (如 DSA) 执行子任务并放松它们之间的依赖关系, 协同调度异构硬件资源以提升计算并行性, 优化十亿级图上最近邻搜索性能。

在 LLM 推理场景中, 单加速器的显存容量限制迫使多加速器协同并行计算, 但由此产生了大量的通信开销。文献 [89] 提出了 CXL-PNM 架构, 在 CXL 大容量 LPDDR 内存中加入推理计算, 可以显著降低 CXL 加速器间的通信开销, 从而提升 LLM 的推理效率。相比 8 GPU 系统, 该架构在 LLM 推理服务中可以降低时延、能耗和硬件成本, 并提升吞吐量。文献 [93] 提出使用基于 CXL 的 PIM 架构来提升大模型推理的性能, 并设计了 CXL 卡之间的 P2P 的集合通信原语, 在多种并行推理方式下, 相比于 GPU 的系统, 基于 CXL 的 PIM 架构在提升推理吞吐性能的同时, 显著降低了系统能耗, 提升了单位成本下的推理性能。

深度学习模型的嵌入层因内存消耗大且数据复用率低, 在执行时间中占比较高。在基于 DIMM 接口的近数据处理架构中, 其规约操作因主机与内存 Rank 内处理单元间的串行数据传输, 导致传输时间长, 且不同 Rank 间处理单元负载不均衡引发计算效率低下。Yun 等人[90] 提出了基于 CXL 的 CLAY 架构去提升嵌入层计算的性能, 其将处理单元放置在设备端分组的 DRAM 模块中, 将全局规约拆分成局部规约, 从而更好地利用处理单元间的并行性, 此外, 利用 CXL 接口的灵活性设计了合适的地址映射方案, 以缓解 DRAM 分组间的负载不均衡的问题。工业界的大型推理系统模型中同样也存在内存消耗大的嵌入层计算, 为此文献 [91] 提出了基于 CXL 的近内存处理 ReCXL 方法, 通过使用软硬协同的方式, 针对推荐模型的特点设计了复杂的无依赖预取机制和细粒度的更新调度方法, 以最大化 CXL 硬件资源的利用率。

上述的 NDP 架构只考虑了特定应用场景下的优化, 文献 [92] 提出了一种针对多种应用的更通用的基于 CXL 内存的内存映射 (memory mapped) NDP 框架 M^{2}NDP, 该框架包含 2 个模块: M^{2}func 和 M^{2}uthread, 前者支持兼容 CXL 内存的主机与 NDP 控制器之间的低开销通信机制, 后者通过引入通用目的的可高度并行执行的支持低成本的通用 NDP 单元来设计。

3.2 CXL 内存研究实验方案

由于真实的 CXL Type 3 设备尚未大规模应用,获取成本较高,当前大部分 CXL 的应用研究实验采用的是模拟方案。因此,本节首先介绍 CXL 内存模拟方案的研究现状,然后总结与分析实验方案的难度、成本与适用场景。

3.2.1 CXL 内存模拟方案

在 2.2 节的调研结果中显示,产业界在 CXL 接口 IP 及内存扩展领域已有实质性推进(部分厂商推出多版本接口 IP 与设备产品),但真实硬件 IP 在学术界的应用尚不普遍,原因在于 CXL 硬件 IP 研发周期长,且实验环境搭建成本高昂(需同时配备支持 CXL 的设备(包括交换机)及 CPU 处理器)。因此,如何实现低成本且高精度的 CXL 内存性能模拟成为核心研究焦点,当前学术工作主要包括 3 类模拟方法:

1)在双 Socket 的主机系统中使用远端 NUMA 节点的内存访问性能去近似 CXL 内存的性能[37,62],如图 13 所示。NUMA 架构模拟 CXL 有 2 点优势:①与 CXL 相同,NUMA 架构具有缓存一致性,其使用 load/store 命令访问远端内存;②从主机 CPU 侧看,CXL 内存访问时延与远端 NUMA 节点内存访问时延相近。

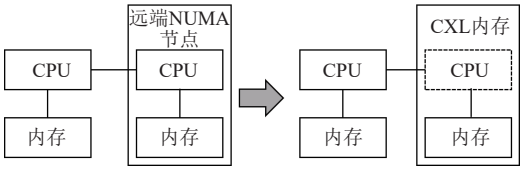


Fig. 13 CXL memory simulated by remote NUMA node

图 13 远端 NUMA 节点模拟 CXL 内存

2)QEMU 模拟器[102]中增加了对 CXL 内存设备的软件支持,包括内存访问的软件接口、呈现方式等,因此在 QEMU 中可实现对 CXL 内存访问[52]的功能。但根据实际测试,这种模拟方式下的 CXL 内存性能不精确,因此并没有得到广泛使用。

3)使用纯软件的方式对 CXL 内存进行模拟,比如 CXL-MemSim[64], Emu-CXL[103]等工作,通过在软件中模拟 CXL 内存的访问环节,并在各个环节配置真实设备的参考性能数据,构建出高精度的 CXL 内存访问行为模型。

具体地,Gem5 模拟器[104]目前暂不支持对 CXL 协议级的模拟。CXL-MemSim 可以在传统服务器上正常运行应用(该应用不需要做特定的修改),记录下应用的性能事件,然后在应用中插入延迟来模拟用户提供的 CXL.mem 拓扑结构的时延和带宽,性能参考图 8 中各个链路的时延和带宽。使用现有的硬件可以支持 CXL-MemSim 使用 load/store 接口来访问 CXL.mem 内存池,模拟方式轻量级且快速。

Emu-CXL[103]旨在为分离式内存提供标准化 CXL 模拟框架及软件抽象接口,通过在 QEMU 虚拟机中模拟 CXL 内存,定义并实现用户空间 API 以访问与分配远端内存,并耦合基于 NUMA 的模拟后端,从而支持分离式内存应用场景。该框架主要用于模拟 CXL 内存的软件访问方式,聚焦软件访问方式模拟以适配开发者需求,但暂未提供具体性能参数指标。

DRackSim[105]搭建了一个可扩展分离式内存系统的模拟框架,建模了分离式系统内的所有必要组件,比如计算节点 CPU、CXL 内存池,可用于 CXL 远端内存相关的模拟。

整体上当前对于 CXL 内存的模拟方案与准确反映真实设备内存的性能存在一定差距[38],为了更好地开展 CXL 的应用研究,仍然需要设计出更准确模拟 CXL 内存性能的方案。

3.2.2 CXL 内存实验方案对比

除了模拟 CXL 内存之外,也有部分工作使用了真实 CXL 设备或者 FPGA 硬件搭建了原型系统进行实验验证,因此,本节对当前应用研究工作的实验方案进行系统梳理与对比,如表 10 所示,分为真实物理设备、硬件设备原型与软件模拟方案。真实 CXL 设备的获取成本较高,可用于应用负载的精确的性能特点分析,但无法进一步地支持策略设计。硬件设备原型可用于创新想法实现,例如使用可编程硬件 FPGA 可组合 CXL 控制器 IP 与 NDP 计算单元,实现不同应用场景下的自定义功能。但系统实现所需的硬件组件多,如支持 CXL 的 CPU 和 CXL 交换机等,开发难度较大。软件模拟可快速迭代策略,根据不同的应用场景,有不同的模拟方案,比如近数据处理时可以使用 Ramulator 内存模拟器,支持 CXL 的 GPU 时可以使用 Accel-Sim 模拟器,但性能数据不够精确。除了这 3 种实验方式之外,有研究工作[85,88]将原型系统与模拟方案进行结合,原型系统用于功能验证,模拟方案用于与其他工作的性能对比。

4 CXL Type 1/2 设备的应用研究进展

与 CXL Type 3 设备类似,CXL Type 1/2 设备也可以使用带 CPU 的 NUMA 节点对进行模拟,但其性能特点与真实设备存在差异[41],同时,Type 1/2 的真实设备非常稀缺,因此关于 Type 1/2 设备的应用研究进展较少。本节首先介绍 CXL Type 1/2 设备的典型应

Table 10 Summary of Experimental Scheme for CXL Research

表 10 CXL 研究实验方案总结

实验方案类型	实验方案	优势	劣势	适用场景
真实物理设备	支持 CXL 的商用 CPU 与 CXL 设备[38-40,67]	实现难度较小	获取成本较高，商用设备研发进度慢，且难以在设备侧进行优化	适用于真实 CXL 设备的应用负载性能特点分析
硬件设备原型	基于 FPGA 实现的设备硬件原型[69,80,106]	可以在设备上进行设计以优化特定应用的性能	实现难度较大，除实现 CXL 设备原型之外，需实现（基于 RISC-V）的 CPU 和 CXL 交换机等	用于内存分离、内存池化、大内存设备研制等场景
软件模拟方案	多 Socket 的主机系统中 NUMA 节点模拟 CXL 内存[37,44,62,76]	实现难度小，可以通过禁用远端 NUMA 节点的 CPU 实现	性能数据不够精确	快速验证 CXL 内存上工作负载的性能表现
	使用 Ramulator 内存模拟器实现基于 CXL 内存的近数据处理[87,90-91]	可快速针对策略进行模拟实现，模拟器均已开源	性能数据不够精确，需根据应用场景将策略设计集成至模拟器	用于实现基于 CXL 内存上的存内计算操作
	使用 Gem5 模拟 CPU，CXL-SSD 软件模拟如 simpleSSD[81]，CXL-Flash 等[82]	可快速针对策略进行模拟实现，模拟器均已开源	性能数据不够精确，需根据应用场景将策略设计集成至模拟器	用于 CXL-SSD 大内存设备的性能优化
	ChampSim 模拟服务器系统[107]，DRAMSim3 模拟内存[53,108]	可快速针对策略进行模拟实现，模拟器均已开源	性能数据不够精确，需根据应用场景将策略设计集成至模拟器	用于可组合的基于 DDR 内存的服务器系统验证
	Gem5 模拟支持 CXL 的 CPU，Accel-Sim 模拟支持 CXL 的 GPU 设备[109]	可快速针对策略进行模拟实现，模拟器均已开源	性能数据不够精确，需根据应用场景将策略设计集成至模拟器	适用于支持 CXL 的 GPU 上的应用性能优化

用案例，如智能网卡加速、异构计算协同等，然后分析 CXL Type 1/2 设备研发所面临的挑战。

4.1 应用场景及案例

CXL Type 1 设备中包含一致性缓存和用于加速页表查询的 DTLB 组件，它的潜在应用是智能网卡这类适合流式处理逻辑的设备。在卸载 CPU 计算的同时，智能网卡可以基于 CXL.cache 协议以缓存一致性的方式访问主机内存，以提升与主机的交互效率。文献 [110] 提到支持缓存一致性接口的智能网卡，相比 PCIe 网卡，其拥有更低的时延和更高的吞吐量。文献 [111] 中提到使用支持 CXL 的智能网卡可以驱逐缓存行的特性去激活 RowHammer 攻击地址中的行数据。

CXL Type 2 设备中同时拥有内存和计算单元，它的典型应用是异构计算场景。现有的异构计算模式是 CPU 负责控制和任务调度，各个加速器负责专门的任务，CPU 负责把数据分发到各个加速器上，加速器完成计算任务后将结果发送回 CPU 主存[112]。引入 CXL.cache 之后，设备可以缓存主机内存数据，支持加速器与主机间频繁的细粒度数据交互，提高主机与设备间数据传输效率，从而加速整个任务的计算。

文献 [41] 在数据中心应用中验证了 CXL Type 2 设备的细粒度异构协同计算场景下的能力，它将 Linux 内核中 zswap 和 ksm 等相关内存优化工具卸载到 CXL Type 2 设备硬件端以实现加速，相较于 PCIe 的实现方式，其操作时延显著降低。

Kwon 等人[106] 提出了容错式推荐模型训练架构 TrainingCXL，使用 CXL 持久内存来替换现有的 SSD，以降低训练过程中检查点（checkpoint）的开销。其架构如图 14 所示，主机 CPU，CXL Type 2 GPU（CXL-GPU），CXL Type 2 持久内存（CXL-MEM）通过 CXL 交换机连接，构成了一个一致性域，支持 CXL-GPU 与 CXL-MEM 之间的自主高效通信。通过在 CXL-MEM 中集成检查点逻辑，有效地掩盖了模型训练过程中检查点计算和恢复的开销。

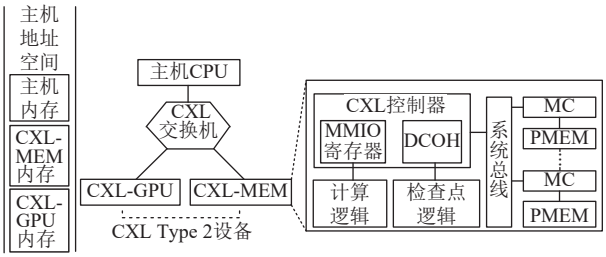


Fig. 14 Hardware architecture of TrainingCXL[106]

图 14 TrainingCXL 硬件架构[106]

文献 [109] 针对大模型优化技术 ZeRO[113-114] 中 CPU 与 GPU 之间的数据效率进行了优化。其分析表明，相邻训练步骤的 CPU 与 GPU 之间传输的参数数据差异很小，大部分参数值维持不变或仅发生微小的字节变化（浮点数通常用 4 B 表示）。该研究探索了此场景下使用 CXL.cache 协议缓存这些微小变化的参数数据，并对 CXL 中的缓存一致性 MESI 协议做了轻量级的修改，以支持 CXL 协议数据包的合并，避免大量数据的冗余传输，提升传输效率。

4.2 Type 1/2 硬件实现的挑战

当前商用 CXL Type 1/2 设备的普及程度远不及 CXL Type 3，一个重要原因是 CXL Type 1/2 设备需要维护设备缓存与主机的一致性，这对设备侧厂商来说难度和成本都很高。以图 15 中 Intel Agilex™ 7 CXL Type 2 FPGA[41] 设备为例，设备由 HIP（hard IP），DCOH，CAFU（coherent request accelerator functional unit），HDM

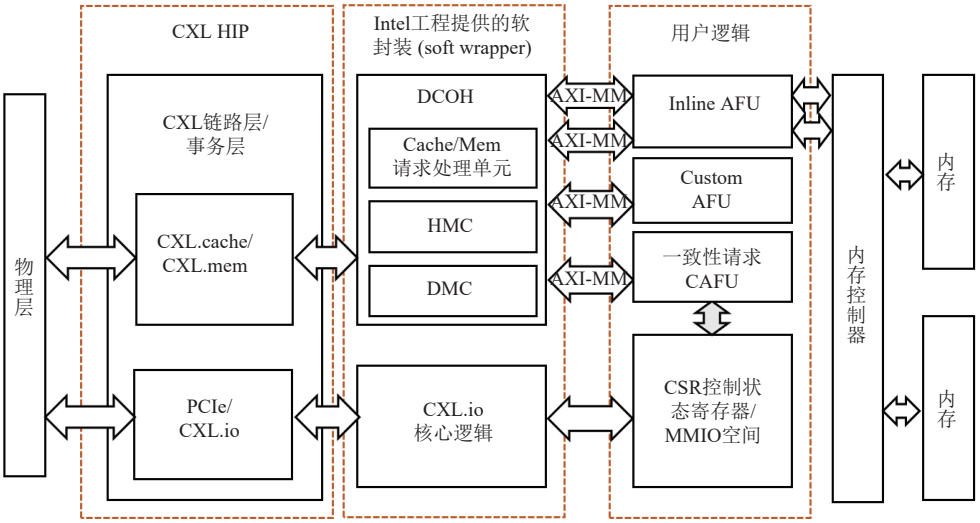


Fig. 15 Device example of Intel CXL Type 2 Agilex™ 7 FPGA

图 15 Intel CXL Type 2 Agilex™ 7 FPGA 设备示例

内存子系统等组成，HIP 负责 3 个子协议的收发，DCOH 负责维护设备缓存与主机缓存的一致性，CAFU 是设备具备一致性请求的加速功能单元。

DCOH 要实现的一致性维护功能重要且繁杂，主要有：1)对于 CXL Type 1 设备，DCOH 负责管理 CXL.cache 的请求和处理响应，即接收 CAFU 发出的系统内存访问请求，访问 DCOH cache(HMC，对主机内存的 cache；DMC，对设备内存的 cache)，根据 cache 命中情况再确定是否发送以及发送哪种 CXL.cache 请求事务给主机，或是直接访问设备本地内存。相应地，也需要根据 CXL IP 接收到的主机 cache 请求或响应进行对应的处理操作。2)对于 CXL Type 2 设备，DCOH 不仅需要处理 CXL.cache 协议请求，还需处理来自主机的 CXL.mem 内存访问请求，即接收并处理来自主机的 CXL.mem 请求事务、向 HDM 子系统发出 HDM 请求，向主机返回相应的响应事务。3)如果 CXL Type 2 设备需要支持反向无效监听，DCOH 还需要实现监听过滤器(snoop filter)等组件，支持对主机缓存状态的跟踪(tracking)，这对设备厂商的门槛要求进一步提高。

而对于 DCOH cache，HMC 与 DMC(device memory cache)的缓存大小、结构和缓存管理策略需要根据设备自身需求特点合理设计。例如 Intel 给出的 FPGA DCOH 实现中[41]，HMC 被设计为 128 KB 大小、4 路组相连结构，DMC 被设计为 32 KB 大小、直接映射结构。缓存大小与主机内存以及设备内存容量相关，缓存结构设计也需要考虑特定应用场景下的需求、不同应用访存行为特点等，这些因素会影响 DCOH cache 命中率，进而影响 CXL 互连性能。

此外，用户逻辑 CAFU 接口设计中，需要通过读、写通道信息通知 DCOH，设备访问的目标是主机内存还是 HDM 地址、指定 DCOH 缓存的状态，以及发出刷新缓存行等请求。同时，作为 CXL IP 和 CAFU 之间交互的桥梁，DCOH 还要实现 CXL 协议接口 CPI(CXL cache/mem protocol interface)与 CAFU AXI 接口之间的一致性转换，确保一致性请求的正确交互。

DCOH 和 CAFU 模块的设计与设备具体需求密切相关，通常 CXL IP 供应商并不会直接提供这部分代码实现。由上述 3 点也可以看出，DCOH 及相关接口的开发难度较大，同时也需要解决加速器软件等生态(例如 NVIDIA GPU 的 CUDA 生态)与 CXL 系统软件栈适配的问题，短期性能收益可能不及其复杂性，这也是目前 CXL Type 2 设备市场应用相对较晚的原因。

5 未来趋势展望

在异构计算模式中，互连技术是实现处理器和加速器高效协同的重要一环，尤其是在信息技术国产化的背景下，实现服务器整机内国产处理器、加速器之间的高效互连变得非常迫切。CXL 互连协议作为业界支持的开放协议标准，获得了学术界和产业界的大量关注，在异构计算、国产化等需求的推动下将对当前的计算机体系结构产生非常大的影响。

随着支持 CXL 2.0 以及 CXL 3.0 协议的硬件产品逐渐成熟，其支持的功能将更丰富，比如内存共享，而这些新的特点将进一步促进 CXL 互连技术的应用研究，为各种类型的应用提供更为广阔的设计空间。

下面将简要展望 CXL 互连技术研究的未来发展趋势，主要列举了 6 个可能的研究方向。

1）基于 CXL 扩展内存的管理优化。CXL 的扩展内存主要包括对主机内存、其他加速器设备内存的扩展。在主机内存扩展场景中，虽然已经优化了设备侧 CXL 内存页面的冷热程度感知的性能，但 CPU 侧仍然需要适应长延迟的 CXL 内存，如 CPU 侧缓存与预取策略优化。针对基于 CXL 的大内存扩展设备，单一存储介质难以满足高性能与大容量的特点，需要在控制器层面进行合理设计，融合混合类型的存储介质，以提高数据访问效率以及存储介质的使用寿命。当前针对加速器设备内存扩展的研究相对有限，而 GPU 显存容量已成为制约大模型计算的关键瓶颈，在此背景下，利用 CXL 技术扩展 GPU 显存的重要性显著提升，亟需研究低时延的 CXL 加速器、多加速器共享 CXL 内存池架构，以提升加速器内存访问效率。

2）基于 CXL 的内存分离、池化与共享。基于 CXL 的内存分离、池化与共享方案比基于网络的方案表现出了更高的性能，但在内存池化系统架构设计、细粒度内存管理、容量动态分配及（异构）内存分层管理等方向仍有很大的优化空间，这需要硬件设备（CXL 交换机、CXL 控制器、支持细粒度分配的 CXL 内存设备 MLD 等）的支持，也需要系统软件的协同（如 Linux 内核的设备驱动和内存管理）。更进一步，随着 CXL 3.0 的出现，未来有望通过 CXL 连接各种处理器和异构加速器，解耦计算和内存等硬件资源，实现多种资源的灵活调度。

3）大模型计算场景下的 CXL 加速。当前，混合专家架构大模型训练、高并发长序列推理等场景对内存的需求急剧增长。如推理场景下，KV 缓存的空间占用量随序列长度呈平方级增长，当序列超过临界长度时，其内存消耗将显著超越模型参数本身。CXL 内存凭借其低成本、大容量特性，为缓解大模型场景下的内存压力提供了新范式，如推理场景中的自回归解码依赖的 KV 缓存计算、训练场景中的周期性检查点保存与恢复计算等内存密集型的操作，均可卸载至 CXL 内存池中，以充分利用 CXL 的大内存容量与带宽。

4）异构协同计算场景下 CXL Type 1 和 CXL Type 2 设备的应用探索。未来针对 CXL Type 1 智能网卡，可利用缓存一致性接口压缩主机至网卡的数据路径延迟，加速网络数据包处理并提升网络性能；对于支持主机侧和设备侧内存访问的 CXL Type 2 加速器，在异构协同计算时，需考虑主机偏置模式与设备偏置模式的快速切换，当执行协同任务时高频切换主机与设备内存的操作需针对性优化，例如将交互密集型内存管理内核部署至设备端。因此需在设备侧设计主机内存和设备内存间的高效协同管理机制，提升加速器与主机间的双向数据访问效率。

5）CXL 协议的模拟研究。低成本的 CXL 协议模拟可促进 CXL 互连技术应用的快速迭代。目前已有多种针对 CXL Type 3 设备的模拟方案，但 CXL Type 1 和 CXL Type 2 设备的模拟方案较少，且均未涉及 CXL 协议事务交互的行为级模拟，难以精准刻画其在缓存行同步、内存语义等核心场景的真实性能。未来可在主机侧与设备侧联合模拟，主机侧利用现有模拟器如 Gem5, Zsim, 实现 CPU 发起 CXL 事务的周期级精确模拟；在设备侧，建立 CXL 控制器、计算单元、内存控制器、DCOH、DTLB 等组件的由真实硬件参数校准的性能模型，量化组件间交互时延对端到端性能的影响，建立从微架构行为到系统级性能的精准映射关系。

6）CXL 与存算一体等新兴技术的结合。存算一体作为一种新型计算范式，通过将计算能力集成到内存芯片内部，有效减少了数据在内存和处理器间的传输开销，显著提升计算效率。在此背景下，CXL 技术展现出独特优势：一方面，相较于受主板上物理插槽限制的 CPU 直连 DDR DRAM, CXL 可集成更高的内存容量与带宽，为构建“内存即算力”架构奠定基础；另一方面，CXL 具备比 PCIe 更低的主机交互延迟，有利于实现主机与存算一体设备的紧密协同。因此，CXL 不仅突破了传统架构内存带宽瓶颈，还将推动存算一体架构的性能提升和广泛应用。

虽然 CXL 互连技术在多领域应用前景广阔，但其国产化进程仍面临硬件、软件与生态协同的多重挑战：1）CPU 支持不完善。海光等国产 CPU 虽已实现基础 CXL 协议支持，但高级功能（如内存池化、多级交换）尚未成熟，且市场占有率较低，不易推广。2）核心控制器 IP 短板。尚未有完全自主研发、可控成熟的 CXL 控制器 IP，这使得相关的设备和交换机研发进程都受到限制。3）软件与生态瓶颈。国产操作系统对 CXL 驱动的适配滞后，缺乏成熟的内存池化与调度管理方案，上层应用难以充分发挥 CXL 优势。这些挑战折射出国内在高端芯片协议栈和全产业链协同的短板，亟需攻克核心 IP 自主化、CXL 设备性能优化、构建软硬件协同验证平台等关键技术。

6 结 论

CXL 技术通过提供处理器与加速器以及外部存储器之间的缓存一致性互连，实现了异构硬件资源(尤其是内存)的高效共享，在大内存应用时代展现出其具有的关键支撑价值。本文对 CXL 技术的现有研究进行了全面综述。首先，介绍了 CXL 协议的互连架构、设备类型，分析了 CXL 技术能实现低时延、高带宽互连的原因；其次，概述 3 种 CXL 设备、CXL 交换机的特点及应用场景，以及产业界相关设备的最新研发进展，并根据现有研究中 CXL 真实物理设备的测试结果总结了其在实际应用中的性能特点；最后，从内存扩展、内存分离及池化、大内存设备、NDP 等方面对不同类型 CXL 设备的应用研究进行了介绍与论述，并对未来的研究趋势进行了展望。

CXL 互连技术的研究目前尚处于早期阶段，仍存在硬件实践较慢、系统软件支持不足等问题。当前基于真实 CXL 硬件的上层应用研究较为有限，亟需更多实际设备上的性能数据验证。但鉴于 CXL 技术在数据中心、人工智能计算及异构计算等新兴领域已展现的显著潜力，其有望在相关技术的深入发展驱动下实现更广泛的应用与价值拓展。

作者贡献声明：李仁刚设计了论文主体框架、具体内容；李霞和刘俊总结并撰写了 CXL 技术的背景、设备性能特点和学术界 CXL 应用研究相关的内容，并对未来趋势进行了展望；侯鹏宇分析并撰写了 CXL 协议事务交互机制，调研了产业界的有关研发工作；王彦伟和杨广文提出论文撰写和修改指导意见。

参 考 文 献

[1] Amirgholami. Scaling of peak hardware FLOPS, and memory/interconnect bandwidth[EB/OL]. (2023-09-02)[2025-03-05]. https://github.com/amirgholami/ai_and_memory_wall/blob/main/imgs/pdfs/hw_scaling. pdf

[2] CXL Consortium. Gen-Z specification archive[EB/OL]. (2024-02-16)[2024-12-18]. https://computeexpresslink.org/resource/gen-z-specification-archive/

[3] CXL Consortium. OpenCAPI specification archive[EB/OL]. (2024-02-16)[2024-12-18]. https://computeexpresslink.org/resource/open-capi-specification- archive/

[4] Wei Ying, Huang Y, Tang Haiming, et al. NVLink-C2C: A coherent off package chip-to-chip interconnect with 40Gbps/pin single-ended signaling[C]//Proc of the 70th IEEE Int Solid-State Circuits Conf (ISSCC). Piscataway, NJ: IEEE, 2023: 160−162

[5] CCIX Consortium. CCIX[EB/OL]. (2021-04-28)[2024-01-15]. https://www.ccixconsortium.com/

[6] CXL. Our members[EB/OL]. (2023-10-31)[2025-01-19]. https://www.computeexpresslink.org/members

[7] Sharma D D, Blankenship R, Berger D. An introduction to the compute express link (CXL) interconnect[J]. ACM Computing Surveys, 2024, 56(11): 1−37

[8] Chen Chen, Zhao Xinkui, Cheng Gauanjie, et al. Next-GEN computing systems with compute express link: A comprehensive survey[J]. arXiv preprint, arXiv: 2412.20249, 2025

[9] Tan Chengsong, Donaldson A F, Wickerson J. Formalising CXL cache coherence[C]//Proc of the 30th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOS'25). New York: ACM, 2025: 437–450

[10] CXL Consortium. Compute express link 3.1 specification[EB/OL]. (2023-09-27)[2025-02-23]. https://computeexpresslink.org/cxl-specification/

[11] Rambus. Compute express link (CXL) controller IP[EB/OL]. (2024-08-05)[2025-02-06]. https://www.rambus.com/interface-ip/cxl/

[12] Rambus. CXL 2.0 controller[EB/OL]. (2022-10-31)[2025-01-25]. https://go.rambus.com/cxl2-controller-product-brief

[13] Synopsys. Synopsys CXL controller IP[EB/OL]. (2025-06-05)[2025-06-25]. https://www.synopsys.com/dw/ipdir.php?ds=dwc_cxl_controller

[14] Cadence. Controller for CXL[EB/OL]. (2024-03-06)[2025-01-27]. https://www.cadence.com/zh_CN/home/tools/ip/design-ip/pcie-and-compute-express-link/controller-for-pcie-and-cxl/controller-for-cxl.html

[15] Alphawave Semi. KappaCORE−64 CXL controller[EB/OL]. (2024-08-19)[2025-02-05]. https://awavesemi.com/silicon-ip/controller-ip/pcie-cxl-controllers/kappacore-64-cxl-controller/

[16] Rapid Silicon. CXL 3.0 IP[EB/OL]. (2024-05-22)[2025-01-25]. https://rapidsilicon.com/cxl-ip/

[17] Rapid Silicon. CXL 3.0 host/device dual mode controller IP[EB/OL]. 2023[2025-01-25]. https://rapidsilicon.com/wp-content/uploads/2023/06/CXL-3.0-Controller-IP-Product-Brief.pdf

[18] Intel. Intel FPGA compute express link (CXL) IP[EB/OL]. (2025-05-02)[2025-05-25]. https://www.intel.com/content/www/us/en/products/details/fpga/intellectual-property/interface-protocols/cxl-ip.html

[19] Intel. Interface protocols-CXL[EB/OL]. (2020-12-11)[2025-02-26]. https://ark.intel.com/content/www/cn/zh/ark/products/series/237677/interface-protocols-cxl.html

[20] Microchip. SMC 2000 smart memory controller[EB/OL]. (2024-05-24)[2025-02-06]. https://ww1.microchip.com/downloads/aemDocuments/documents/DCS/ProductDocuments/Brochures/SMC-2000-Smart-Memory-Controllers-00004551.pdf

[21] AsteraLabs. Memory expansion: Reduce TCO and increase CPU efficiency with memory bandwidth and capacity expansion[EB/OL]. (2025-03-14)[2025-03-30]. https://www.asteralabs.com/applications/

memory-expansion/

[22] AsteraLabs. Leo CXL smart memory controllers[EB/OL]. (2025-05-27)[2025-05-30].https://www.asteralabs.com/products/leo-cxl-smart-memory-controllers/

[23] Montage Technology. Montage Technology partners with Samsung Electronics to deliver CXL™ memory products[EB/OL]. (2022-05-26)[2024-01-29]. https://www.montage-tech.com/Press_Releases/20220526

[24] Micron, AMD. CXL memory expansion: A closer look on actual platform[EB/OL]. (2025-07-21)[2025-08-28]. https://assets.micron.com/adobe/assets/urn:aaid:aem:17ee2655-7d57-4aed-b7b2-70bc971-5db8d/original/as/cxl-memory-expansion-a-close-look-on-actual-platform.pdf

[25] Samsung. Samsung unveils industry-first memory module incorporating new CXL interconnect standard[EB/OL]. (2021-05-11)[2025-01-26].https://news.samsung.com/global/samsung-unveils-industry-first-memory-module-incorporating-new-cxl-interconnect-standard

[26] Samsung. Samsung Electronics introduces industry's first 512GB CXL memory module[EB/OL]. (2022-05-10)[2024-01-26]. https://news.samsung.com/global/samsung-electronics-introduces-industrys-first-512gb-cxl-memory-module

[27] Samsung. Samsung Electronics unveils far-reaching, next-generation memory solutions at Flash Memory Summit 2022[EB/OL]. (2022-08-03)[2025-01-03]. https://news.samsung.com/global/samsung-electronics-unveils-far-reaching-next-generation-memory-solutions-at-flash-memory-summit-2022

[28] Samsung. Samsung develops industry's first CXL DRAM supportingCXL 2.0[EB/OL]. (2023-05-12)[2025-01-26]. https://news.samsung.com/global/samsung-develops-industrys-first-cxl-dram-supporting-cxl-2-0

[29] Montage Technology. M88MX5891/5851[EB/OL]. (2023-08-24)[2025-03-05]. https://www.montage-tech.com/MXC/M88MX5891-%265851

[30] Micron. Powering the next generation of supermicro petascale platforms with Micron CZ120 memory expansion using CXL[EB/OL]. (2024-08-22)[2025-01-29]. https://www.micron.com/about/blog/memory/dram/powering-the-next-generation-of-supermicro-petascale-platforms-with-micron-cz120-memory-expansion-using-cxl

[31] SK Hynix. SK hynix develops DDR5 DRAM CXL memory to expand the CXL memory ecosystem[EB/OL]. (2022-08-01)[2024-01-29]. https://news.skhynix.com/sk-hynix-develops-ddr5-dram-cxl-tm-memory-to-expand-the-cxl-memory-ecosystem/

[32] Lightelligence. Advantages of optical CXL for disaggregated compute architectures[EB/OL]. (2023-08-08)[2025-02-08]. https://memverge.com/wp-content/uploads/Lightelligence_CXL-Forum-FMS-2023.pdf

[33] Montage Technology. M88RT61632 (16-Lane PCIe 6.x/CXL 3.x Retimer)[EB/OL]. (2025-01-22)[2025-07-01]. https://web.montagetech.com/PCIe_Retimer/M88RT61632

[34] IEIT Systems. Meta brain general purpose servers NF5280G8[EB/OL]. (2024-10-22)[2025-06-27]. https://www.ieisystem.com/product/server/15918.html(in Chinese)
(浪潮信息. 元脑通用服务器 NF5280G8[EB/OL]. (2024-10-22)[2025-06-27]. https://www.ieisystem.com/product/server/15918.html)

[35] Lenovo. ThinkSystem SR860 V3 server[EB/OL]. (2024-08-22)[2025-07-01]. https://pubs.lenovo.com/sr860-v3/

[36] Yang Xinjun, Zhang Yingqiang, Chen Hao, et al. Unlocking the potential of CXL for disaggregated memory in cloud-native databases[C]//Proc of the 44th ACM SIGMOD/PODS 2025 Int Conf on Management of Data. New York: ACM, 2025: 689−702

[37] Li Huaicheng, Berger D S, Hsu L, et al. Pond: CXL-based memory pooling systems for cloud platforms[C]//Proc of the 28th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOS'23). New York: ACM, 2023: 574−587

[38] Sun Yan, Yuan Yifan, Yu Zeduo, et al. Demystifying CXL memory with genuine CXL-ready systems and devices[C]//Proc of the 56th Annual IEEE/ACM Int Symp on Microarchitecture (MICRO'23). New York: ACM, 2023: 105−121

[39] Tang Yupeng, Zhou Ping, Zhang Wenhui, et al. Exploring performance and cost optimization with ASIC-based CXL memory[C]//Proc of the 19th European Conf on Computer Systems. New York: ACM, 2024: 818–833

[40] Liu Jinshu, Hadian H, Xu Hanchen, et al. Dissecting CXL memory performance at scale: Analysis, modeling, and optimization[J]. arXiv preprint, arXiv: 2409.14317, 2024

[41] Ji Houxiang, Vanavasam S, Zhou Yang, et al. Demystifying a CXL type-2 device: A heterogeneous cooperative computing perspective[C]//Proc of the 57th IEEE/ACM Int Symp on Microarchitecture (MICRO). Piscataway, NJ: IEEE, 2024: 1504−1517

[42] Cho A, Daglis A. StarNUMA: Mitigating NUMA challenges with memory pooling[C]//Proc of the 57th IEEE/ACM Int Symp on Microarchitecture (MICRO). Piscataway, NJ: IEEE, 2024: 997−1012

[43] Wang Xi, Liu Jie, Wu Jianbo, et al. Exploring and evaluating real-world CXL: Use cases and system adoption[J]. arXiv preprint, arXiv: 2405.14209, 2024

[44] Wahlgren J, Gokhale M B, Peng I B. Evaluating emerging CXL-enabled memory pooling for HPC systems[C]//Proc of the 2022 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC). Piscataway, NJ: IEEE, 2022: 11−20

[45] Fridman Y, Desai S M, Singh N, et al. CXL memory as persistent memory for disaggregated HPC: A practical approach[C]//Proc of the Workshops of the Int Conf on High Performance Computing, Network, Storage, and Analysis (SC-W'23). New York: ACM, 2023: 983−994

[46] Ryu S, Kim S, Jun J, et al. System optimization of data analytics platforms using compute express link (CXL) memory[C]//Proc of the 10th IEEE Int Conf on Big Data and Smart Computing (BigComp). Piscataway, NJ: IEEE, 2023: 9−12

[47] Lee D, Willhalm T, Ahn M, et al. Elastic use of far memory for in-memory database management systems[C]//Proc of the 19th Int

Workshop on Data Management on New Hardware (DaMoN'23). New York: ACM, 2023: 35−43

[48] Ahn M, Chang A, Lee D, et al. Enabling CXL memory expansion for in-memory database management systems[C]//Proc of the 18th Data Management on New Hardware (DaMoN'22). New York: ACM, 2022: 1−5

[49] Guo Yunyan, Li Guoliang. A CXL-powered database system: Opportunities and challenges[C]//Proc of the 40th IEEE Int Conf on Data Engineering (ICDE). Piscataway, NJ: IEEE, 2024: 5593−5604

[50] Ahn M, Willhalm T, May N, et al. An examination of CXL memory use cases for in-memory database management systems using SAP HANA[J]. Proceeding of the VLDB Endowment, 2024, 17(12): 3827−3840

[51] Huang Wentao, Sha Mo, Lu Mian, et al. Bandwidth expansion via CXL: A pathway to accelerating in-memory analytical processing[J]. Proceeding of the VLDB Workshops, 2024, 17: 1−9

[52] Yang Qirui, Jin Runyu, Davis B, et al. Performance evaluation on CXL-enabled hybrid memory pool[C]//Proc of the 17th IEEE Int Conf on Networking, Architecture and Storage (NAS). Piscataway, NJ: IEEE, 2022: 1−5

[53] Cho A, Saxena A, Qureshi M, et al. CoAXIAL: A CXL-centric memory system for scalable servers[C]//Proc of the 36th Int Conf for High Performance Computing, Networking, Storage and Analysis (SC24). Piscataway, NJ: IEEE, 2024: 1−15

[54] Arif M, Assogba K, Rafique M M, et al. Exploiting CXL-based memory for distributed deep learning[C]//Proc of the 51st Int Conf on Parallel Processing (ICPP'22). New York: ACM, 2022: 1−11

[55] Wang Zhao, Chen Yiqi, Li Cong, et al. CTXNL: A software-hardware co-designed solution for efficient CXL-based transaction processing[C]//Proc of the 30th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOS'25). New York: ACM, 2025: 192–209

[56] Li Huaicheng, Berger D S, Novakovic S, et al. First-generation memory disaggregation for cloud platforms[J]. arXiv preprint, arXiv: 2203.00241, 2022

[57] Tang Yupeng, Cheng Runxiang, Zhou Ping, et al. Exploring CXL-based KV cache storage for LLM serving[C/OL]//Proc of the 5th Machine Learning for Systems Workshop at NeurIPS 2024. 2024[2025-09-28]. https://mlforsystems.org/assets/papers/neurips-2024/paper17.pdf

[58] Gouk D, Kang S, Bae H, et al. Breaking barriers: Expanding GPU memory with sub-two digit nanosecond latency CXL controller[C]//Proc of the 16th ACM Workshop on Hot Topics in Storage and File Systems. New York: ACM, 2024: 108−115

[59] Arif M, Maurya A, Rafique M M. Accelerating performance of GPU-based workloads using CXL[C]//Proc of the 13th Workshop on AI and Scientific Computing at Scale using Flexible Computing. New York: ACM, 2023: 27−31

[60] Lee J, Kim J. Synergizing CXL with unified memory for scalable GPU memory expansion[C]//Proc of the 23rd Int Conf on Electronics, Information, and Communication (ICEIC). Piscataway, NJ: IEEE, 2024: 1−4

[61] Sano S, Bando Y, Hiwada K, et al. GPU graph processing on CXL-based microsecond-latency external memory[C]//Proc of the 35th Workshops of the Int Conf on High Performance Computing, Network, Storage, and Analysis (SC'23). New York: ACM, 2023: 962–972

[62] Maruf H A, Wang Hao, Dhanotia A, et al. TPP: Transparent page placement for CXL-enabled tiered-memory[C]//Proc of the 28th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOS'23). New York: ACM, 2023: 742−755

[63] Lee Y, Kim Y, Yeom H Y. Lightweight memory tracing for hot data identification[J]. Cluster Computing, 2020, 23(3): 2273−2285

[64] Yang Yiwei, Safayenikoo P, Ma Jiacheng, et al. CXLMemSim: A pure software simulated CXL. mem for performance characterization[J]. arXiv preprint, arXiv: 2303.06153, 2023

[65] Boles D, Waddington D, Roberts D A. CXL-enabled enhanced memory functions[J]. IEEE Micro, 2023, 43(2): 58−65

[66] Lee H, Lee S, Jung Y, et al. T-CAT: Dynamic cache allocation for tiered memory systems with memory interleaving[J]. IEEE Computer Architecture Letters, 2023, 22(2): 73−76

[67] Zhou Zhe, Chen Yiqi, Zhang Tao, et al. NeoMem: Hardware/software co-design for CXL-native memory tiering[J]. arXiv preprint, arXiv: 2403.18702, 2024

[68] Sun Yan, Kim J, Yu Zeduo, et al. M5: Mastering page migration and memory management for CXL-based tiered memory systems[C]//Proc of the 30th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOS'25). New York: ACM, 2025: 604−621

[69] Gouk D, Lee S, Kwon M, et al. Direct access, high-performance memory disaggregation with DIRECTCXL[C]//Proc of the 31st USENIX Annual Technical Conf (USENIX ATC'22). Berkeley, CA: USENIX Association, 2022: 287−294

[70] Wang Chenjiu, He Ke, Fan Ruiqi, et al. CXL over Ethernet: A novel FPGA-based memory disaggregation design in data centers[C]//Proc of the 31st IEEE Annual Int Symp on Field-Programmable Custom Computing Machines (FCCM). Piscataway, NJ: IEEE, 2023: 75−82

[71] Ha M, Ryu J, Choi J, et al. Dynamic capacity service for improving CXL pooled memory efficiency[J]. IEEE Micro, 2023, 43(2): 39−47

[72] Zhong Yuhong, Berger D S, Waldspurger C, et al. Managing memory tiers with CXL in virtualized environments[C]//Proc of the 18th USENIX Symp on Operating Systems Design and Implementation (OSDI'24). Berkeley, CA: USENIX Association, 2024: 37−56

[73] Wang Zhonghua, Guo Yixing, Lu Kai, et al. Rcmp: Reconstructing RDMA-based memory disaggregation via CXL[J]. ACM Transactions on Architecture and Code Optimization, 2024, 21(1): 1−26

[74] Alverti C, Psomadakis S, Ocalan B, et al. CXLfork: Fast remote fork over CXL fabrics[C]//Proc of the 30th ACM Int Conf on Architectural Support for Programming Languages and Operating

Systems (ASPLOS'25). New York: ACM, 2025: 210–226

[75] Zhang Xu, Chang Yisong, Lu Tianyue, et al. Rethinking design paradigm of graph processing system with a CXL-like memory semantic fabric[C]//Proc of the 23rd IEEE/ACM Int Symp on Cluster, Cloud and Internet Computing (CCGrid). Piscataway, NJ: IEEE, 2023: 25−35

[76] Tang Wenda, Han Ying, Ai Tianxiang, et al. Yggdrasil: Reducing network I/O tax with (CXL-based) distributed shared memory[C]//Proc of the 53rd Int Conf on Parallel Processing. New York: ACM, 2024: 597–606

[77] Alvaro P, Adiletta M, Cheng D, et al. NotNets: Accelerating microservices by bypassing the network[C]//Proc of the 15th ACM SIGOPS Asia-Pacific Workshop on Systems. New York: ACM, 2024: 67–73

[78] Zhang Mingxing, Ma Teng, Hua Jinqi, et al. Partial failure resilient memory management system for (CXL-based) distributed shared memory[C]//Proc of the 29th Symp on Operating Systems Principles. New York: ACM, 2023: 658–674

[79] Lee H, Choi K, Lee H J, et al. SDM: Sharing-enabled disaggregated memory system with cache coherent compute express link[C]//Proc of the 32nd Int Conf on Parallel Architectures and Compilation Techniques (PACT). Piscataway, NJ: IEEE 2023: 86−98

[80] Jung M. Hello bytes, bye blocks: PCIe storage meets compute express link for memory expansion (CXL-SSD)[C]//Proc of the 14th ACM Workshop on Hot Topics in Storage and File Systems. New York: ACM, 2022: 45−51

[81] Kwon M, Lee S, Jung M. Cache in hand: Expander-driven CXL prefetcher for next generation CXL-SSD[C]//Proc of the 15th ACM Workshop on Hot Topics in Storage and File Systems (HotStorage' 23). New York: ACM, 2023: 24−30

[82] Yang Shaopeng, Kim M, Nam S, et al. Overcoming the memory wall with CXL-enabled SSDs[C]//Proc of the 31st USENIX Annual Technical Conf (USENIX ATC'23). Berkeley, CA: USENIX Association, 2023: 601−617

[83] Robinson C. Samsung memory-semantic CXL SSD at FMS 2022 powered by AMD-Xilinx[EB/OL]. (2022-08-18)[2025-02-07]. https://www.servethehome.com/samsung-memory-semantic-cxl-ssd-at-fms-2022-powered-by-amd-xilinx/

[84] Zhan Yekang, Hu Haichuan, Yang Xiangrui, et al. RomeFS: A CXL-SSD aware file system exploiting synergy of memory-block dual paths[C]//Proc of the 15th ACM Symp on Cloud Computing. New York: ACM, 2024: 720−736

[85] Li Shaobo, Zhou Yirui, Ren Hao, et al. ByteFS: System support for (CXL-based) memory-semantic solid-state drives[C]//Proc of the 30th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOS'25). New York: ACM, 2025: 116−132

[86] Sim J, Ahn S, Ahn T, et al. Computational CXL-memory solution for accelerating memory-intensive applications[J]. IEEE Computer Architecture Letters, 2023, 22(1): 5−8

[87] Huangfu Wenqin, Malladi K T, Chang A, et al. BEACON: Scalable near-data-processing accelerators for genome analysis near memory pool with the CXL support[C]//Proc of the 55th IEEE/ACM Int Symp on Microarchitecture (MICRO). Piscataway, NJ: IEEE, 2022: 727−743

[88] Jang J, Choi H, Bae H, et al. CXL-ANNS: Software-hardware collaborative memory disaggregation and computation for billion-scale approximate nearest neighbor search[C]//Proc of the 31st USENIX Annual Technical Conf (USENIX ATC' 23). Berkeley, CA: USENIX Association, 2023: 585−600

[89] Park S, Kim K, So J, et al. An LPDDR-based CXL-PNM platform for TCO-efficient inference of transformer-based large language models[C]//Proc of the 30th IEEE Int Symp on High-Performance Computer Architecture (HPCA). Piscataway, NJ: IEEE, 2024: 970−982

[90] Yun Sungmin, Nam H, Kyung K, et al. CLAY: CXL-based scalable NDP architecture accelerating embedding layers[C]//Proc of the 38th ACM Int Conf on Supercomputing. New York: ACM, 2024: 338–351

[91] Liu Haifeng, Zheng Long, Huang Yu, et al. Enabling efficient large recommendation model training with near CXL memory processing[C]//Proc of the 51st ACM/IEEE Annual Int Symp on Computer Architecture (ISCA). Piscataway, NJ: IEEE, 2024: 382−395

[92] Ham H, Hong J, Park G, et al. Low-overhead general-purpose near-data processing in CXL memory expanders[C]//Proc of the 57th IEEE/ACM Int Symp on Microarchitecture (MICRO). Piscataway, NJ: IEEE, 2024: 594−611

[93] Gu Yufeng, Khadem A, Umesh S, et al. PIM is all you need: A CXL-enabled GPU-free system for large language model inference[J]. arXiv preprint, arXiv: 2502.07578, 2025

[94] Kim K, Kim H, So J, et al. SMT: Software-defined memory tiering for heterogeneous computing systems with CXL memory expander[J]. IEEE Micro, 2023, 43(2): 20−29

[95] Aguilera M K, Amaro E, Amit N, et al. Memory disaggregation: Why now and what are the challenges[J]. ACM SIGOPS Operating Systems Review, 2023, 57(1): 38−46

[96] Calciu I, Imran M T, Puddu I, et al. Rethinking software runtimes for disaggregated memory[C]//Proc of the 26th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOS'21). New York: ACM, 2021: 79–92

[97] Guo Zhiyuan, Shan Yizhou, Luo Xuhao, et al. Clio: A hardware-software co-designed disaggregated memory system[C]//Proc of the 27th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems (ASPLOS'22). New York: ACM, 2022: 417−433

[98] Angel S, Nanavati M, Sen S. Disaggregation and the application[C]//Proc of the 12th USENIX Conf on Hot Topics in Cloud Computing (HotCloud'20). New York: ACM, 2020: 15–22

[99] Kevin L, Yoshio T, Jose S R, et al. System-level implications of disaggregated memory[C]//Proc of the 18th IEEE Int Symp on High-Performance Computing Architecture. Piscataway, NJ: IEEE, 2012:

1−12

[100] Maruf H A, Chowdhury M. Effectively prefetching remote memory with leap[C]//Proc of the 29th USENIX Conf on USENIX Annual Technical Conf (USENIX ATC'20). New York: ACM, 2020: 843−857

[101] Robinson C. Samsung and AMD Xilinx launch 2nd gen SmartSSD[EB/OL]. (2022-07-12)[2025-03-03]. https://www.servethehome.com/samsung-and-amd-xilinx-launch-2nd-gen-smartssd/

[102] QEMU. Compute express link (CXL)[EB/OL]. 2025[2025-02-27]. https://www.qemu.org/docs/master/system/devices/cxl.html

[103] Raja Gond, Kulkarni P. Emu-CXL: An emulation framework for CXL-based disaggregated memory applications[J]. arXiv preprint, arXiv: 2404.08311, 2023

[104] Lowe-Power J. Introduction of Gem5[EB/OL]. (2025-08-21)[2025-09-03]. https://www.gem5.org/documentation/learning_gem5/introduction/

[105] Puri A, Bellamkonda K, Narreddy K, et al. DRackSim: Simulating CXL-enabled large-scale disaggregated memory systems[C]//Proc of the 38th ACM SIGSIM Conf on Principles of Advanced Discrete Simulation. New York: ACM, 2024: 3–14

[106] Kwon M, Jang J, Choi H, et al. Failure tolerant training with persistent memory disaggregation over CXL[J]. IEEE Micro, 2023, 43(2): 66−75

[107] Gober N, Chacon G, Wang Lei, et al. ChampSim[EB/OL]. (2024-12-06)[2025-03-04]. https://github.com/ChampSim/ChampSim

[108] Li Shang, Yang Zhiyuan, Reddy D, et al. DRAMsim3: A cycle-accurate, thermal-capable DRAM simulator[J]. IEEE Computer Architecture Letters, 2020, 19(2): 106−109

[109] Xu Dong, Feng Yuan, Shin K, et al. Efficient tensor offloading for large deep-learning model training based on compute express link[C]//Proc of the 36th Int Conf for High Performance Computing, Networking, Storage and Analysis (SC'24). Piscataway, NJ: IEEE, 2024: 1−18

[110] Schuh H N, Krishnamurthy A, Culler D, et al. CC-NIC: A cache-coherent interface to the NIC[C]//Proc of the 29th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2024: 52−68

[111] Song Chihun, Kim M J, Wang Tianchen, et al. TAROT: A CXL smartNIC-based defense against multi-bit errors by row-hammer attacks[C]//Proc of the 29th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2024: 981–998

[112] Mittal S, Vetter J S. A survey of CPU-GPU heterogeneous computing techniques[J]. ACM Computing Surveys, 2015, 47(4): 1−35

[113] Rajbhandari S, Rasley J, Ruwase O, et al. ZeRO: Memory optimizations toward training trillion parameter models[C]//Proc of the 32nd Int Conf for High Performance Computing, Networking, Storage and Analysis (SC'20). Piscataway, NJ: IEEE, 2020: 1−16

[114] Ge Xuran, Ou Yang, Wang Bo, et al. Survey of storage optimization techniques in large language model inference[J]. Journal of Computer Research and Development, 2025, 62(3): 545−562 (in Chinese)
(葛旭冉, 欧洋, 王博, 等. 大语言模型推理中的存储优化技术综述[J]. 计算机研究与发展, 2025, 62(3): 545−562)



Li Rengang, born in 1980. PhD candidate, senior engineer. Member of CCF. His main research interests include computer architecture and heterogeneous computing.
李仁刚, 1980 年生。博士研究生, 正高级工程师。CCF 会员。主要研究方向为计算机体系结构、异构计算。



Li Xia, born in 1993. PhD. Member of CCF. Her main research interests include computer architecture and heterogeneous computing.
李　霞, 1993 年生。博士。CCF 会员。主要研究方向为计算机体系结构、异构计算。



Liu Jun, born in 1990. PhD. Member of CCF. His main research interests include computer architecture and heterogeneous computing.
刘　俊, 1990 年生。博士。CCF 会员。主要研究方向为计算机体系结构、异构计算。



Hou Pengyu, born in 1996. PhD. His main research interests include computer architecture and heterogeneous computing.
侯鹏宇, 1996 年生。博士。主要研究方向为计算机体系结构、异构计算。



Wang Yanwei, born in 1985. PhD candidate, associate researcher. Member of CCF. His main research interests include computer architecture and heterogeneous computing.
王彦伟, 1985 年生。博士研究生, 副研究员。CCF 会员。主要研究方向为计算机体系结构、异构计算。



Yang Guangwen, born in 1963. PhD, professor, PhD supervisor. Member of CCF. His main research interest includes high performance computing.
杨广文, 1963 年生。博士, 教授, 博士生导师。CCF 会员。主要研究方向为高性能计算。