计算机研究与发展
Journal of Computer Research and Development
https://crad.ict.ac.cn
63(3): 567−584, 2026

# 基于 CXL 的存储阵列全局缓存技术

李 强[1] 刘 涛[1] 张雪庆[1,2] 孟宪伟[3] 孙明刚[1] 李博乐[3] 陈 曦[1]
[1]（浪潮电子信息产业股份有限公司 济南 250101）
[2]（华中科技大学计算机科学与技术学院 武汉 430074）
[3]（浪潮（北京）电子信息产业有限公司 北京 100095）
（li.qiangbj@ieisystem.com）

# Storage Arrays Global Cache Technology Based on CXL

Li Qiang[1], Liu Tao[1], Zhang Xueqing[1,2], Meng Xianwei[3], Sun Minggang[1], Li Bole[3], and Chen Xi[1]
[1]（*IEIT SYSTEMS Co., Ltd., Jinan* 250101）
[2]（*School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan* 430074）
[3]（*Inspur* (*Beijing*) *Electronic Information Industry Co., Ltd., Beijing* 100095）

**Abstract** Global cache is a core technology for enterprise-grade storage arrays, whose primary function is to improve data write performance. I/O operations return immediately once data is written into the memory cache of storage controllers, without waiting for the data to be persisted to slow HDDs or SSDs. However, the design of global cache in enterprise-grade storage products must comprehensively consider factors such as the number of replicas, system high availability, and failure scenarios. Constrained by the mutual interactions of these factors, the number of controllers in storage clusters typically cannot exceed 4, and system high availability fails to support the scenario where three out of four controllers fail concurrently. Furthermore, the transmission of cache replica data between controllers via RDMA(remote direct memory access) technology also impairs storage performance. Thus, a memory-disaggregated global cache system prototype is designed based on the latest CXL(compute express link) technology. With memory cache disaggregated from within controllers to external, the system offers three key advantages: first, controllers and memory cache can scale out independently, supporting multiple cache replicas; second, controller failures do not cause cache data loss, supporting concurrent failures of $n-1$ out of $n$ controllers; third, CXL technology significantly optimizes storage performance while enhancing system scalability and high availability. Experimental results demonstrate that the CXL-based global cache achieves a 38% performance improvement compared with the RDMA-based global cache.

**Key words** enterprise storage; storage array; CXL; global cache; disaggregate architecture

**摘 要** 全局缓存是企业级存储阵列的关键技术，其核心功能是提高数据的写入速度：用户 I/O 写入存储控制器的内存缓存后立即返回，无需等待数据持久化至低速 HDD 或 SSD 设备。然而，企业级存储产品的全局缓存设计需综合考虑副本数量、系统高可用及故障场景等因素。受这些因素的相互制约，存储集群的控制器数量通常难以超过 4 个，且系统可用性无法支持 4 个控制器同时坏 3 个。此外，控制器间通过 RDMA(remote direct memory access)传输缓存副本数据，也会影响存储性能。为此，基于最新的 CXL（compute express link）技术，设计了内存分离式的全局缓存系统原型。内存缓存从控制器内部解耦

**收稿日期**：2025-06-04；**修回日期**：2025-11-21
**基金项目**：国家重点研发计划项目（2022YFB4501100）
This work was supported by the National Key Research and Development Program of China (2022YFB4501100).
**通信作者**：张雪庆（zhangxq@ieisystem.com）

至外部后,系统具有以下优势:控制器与内存缓存可独立横向扩展,可以支持多个缓存副本;控制器故障不会导致缓存数据丢失,$n$ 个控制器可以同时故障 $n-1$ 个;CXL 技术在提升系统扩展性与可用性的同时,能显著优化存储性能。实验结果表明,CXL 全局缓存性能比 RDMA 全局缓存提升 38%。

**关键词** 企业存储;存储阵列;CXL;全局缓存;分离式架构

**中图法分类号** TP302.1

**DOI:** 10.7544/issn1000-1239.202550376 **CSTR:** 32373.14.issn1000-1239.202550376

以人工智能、大数据、云计算等战略性新兴产业为代表的新质生产力,对数据存储的访问性能、高可用性、可扩展性等方面提出了更加极致的需求。本文基于最新的 CXL(compute express link)[1] 技术,构建了新型的内存分离式全局缓存系统,在企业级存储阵列的先进数据分析处理技术领域进行了探索。

CXL 是 Intel 公司在 2019 年推出的开放性互连协议,用于处理器、内存扩展和加速器的高速缓存一致性互连协议。CXL 技术可以构建分离式的内存池。CPU, GPU 等处理器可以通过同步的 load/store 指令高效访问远端内存。而且,CXL 内存的时延很低,仅为 250~500 ns,远低于 RDMA 的硬件时延 2~3 μs。因此,CXL 是目前性能最好的远程内存访问技术。

全局缓存[2-3] 是企业存储的核心技术之一,最早出现在 20 世纪 90 年代的磁盘阵列产品上。其设计初衷是对 HDD 进行写加速,用户的 I/O 在写入 DRAM 缓存之后就会立即返回给用户,而无需等待长达 10 ms 的磁盘写入操作完成后再返回。该技术在当前的 NVMe(non-volatile memory express)全闪阵列中也被广泛应用。

在高性能全闪存阵列中,通过 RDMA 技术在控制器间传输全局缓存的写 I/O 数据,会严重影响存储的性能。以浪潮存储系统为例,使用 DRAM 内存替代 RDMA 远程内存做缓存副本,同时保留相关软件功能,写 I/O 的 IOPS 从 16.37 万提升到了 22.84 万,提高了 39.5%①。

高可用设计是全局缓存技术的另一个技术难点。当存储集群中的某个控制器断电或故障后,为了避免故障控制器中的 DRAM 数据丢失,全局缓存中的备份数据负责把用户的 I/O 数据写入到 HDD 或 SSD 中。为此,需要解决数据副本数量、副本如何在不同数量的控制器中分布、控制器故障和恢复等技术难题。当前最常见的实现方法是存储控制器间通过 RDMA 进行高速互连,2 个数据副本在 4 个存储控制器中构成循环镜像结构。工程上的难点在于 4 个控制器的缓存镜像实现非常复杂,要考虑各种异常情况。因此,市场上没有超过 4 个控制器的全局缓存存储产品。

基于 CXL 的远程内存技术可以解决全局缓存的性能和高可用技术难题。从性能的角度来看,CXL 比 RDMA 时延更低,能消除控制器间的数据流量,同时还能实现控制器外部的写缓存数据备份。从高可用角度来看,CXL 把内存从控制器内部解耦到控制器外部,控制器故障和 CXL 缓存数据的故障相互隔离,两者互不影响。同时,存储软件无需考虑数据副本的分布问题,也无需在控制器故障后改变多控缓存系统的拓扑结构,控制器可以扩展至 8 个。

本文在浪潮信息存储产品 AS18000 上实现了系统原型,主要贡献有 4 个方面:

1) 在存储产品上引入最新的 CXL 硬件,提出了一种内存解耦的全局缓存架构,实现了原型系统,验证了该架构的技术可行性。

2)通过 libnuma 的函数 *numa_alloc_onnode* 申请 CXL 内存时延高达 19.9 μs,本文开发了用户态内存管理模块,把申请内存时延降至 1.31 μs。

3)CXL 全局缓存技术显著提升了存储的性能,实验数据表明 100% 写性能提高了 38%。

4)CXL 全局缓存在现有的硬件上可以支持 8 控集群。支持 $n$ 个控制器的同时坏 $n-1$ 个,好于现有的 RDMA 循环镜像技术依次坏 $n-1$ 个控制器。

## 1 背景介绍

为实现高可用性与高性能服务,企业级存储阵列通常采用多控制器设计。中低端存储采用双控制器设计,高端存储是 4 控。全局缓存具有以下技术价值:

1) 高可用。当存储系统中有任何 1 个控制器故障时,另外 1 个控制器可以接替其工作,从而保障用

① 存储的性能受软件版本、RAID 配置、卷数量、CPU、网卡、扩展柜、盘类型、组网结构、测试方法等众多因素影响。本文的测试结果不代表产品的最大性能。

户读写数据的连续性和可用性。

2）数据保护。用户数据写入时，双控会将数据同时写入 2 个控制器的缓存中，然后再写入到 HDD 或 SSD 中，这样即使 1 个控制器出现故障时，另外 1 台控制器中有数据备份，用户数据不会丢失。

3）高性能。双控设计可以实现负载均衡，将数据访问请求均匀地分配到 2 个控制器上，避免单个控制器负载过高，从而提高系统的整体性能和响应速度。

下面介绍常见的全局缓存架构和技术。

### 1.1 双副本循环镜像

最常见的全局缓存技术是双副本的循环镜像[4-5]，即数据副本以循环的结构分布在各个控制器中，如图 1 所示。

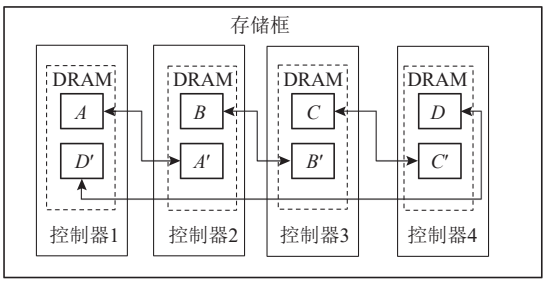



Fig. 1 Four controller global cache（loop mirror）

图 1 4 控全局缓存（循环镜像）

该架构设计的好处：一是，正常 I/O 时仅多 1 次 RDMA 的网络传输，性能高。二是，当控制器数量变化时，比如 1 个控制器故障，集群控制器数量从 4 个变成 3 个时，副本的循环结构不变。

该架构的缺点如下：

1）控制器数量超过 4 个时，故障处理会非常复杂，厂商一般会在 4 控的基础上进一步做更大规模集群，如 8 个 4 控互连形成 32 控集群，而不会做 1 个 32 控的大集群。

2）当用户提出更高的可用性要求，比如缓存 3 副本，需要单独设计实现方式。比如华为公司把第 3 副本放在了另外 1 个 4 控集群中[5]，但是跨框的网络通信会影响存储性能。

3）2 个副本无法支持控制器同时 4 坏 3，只能支持依次 4 坏 3。当 1 个控制器故障时，需要等系统完成缓存数据重建，形成 3 个控制器的循环镜像后，才能允许下一个控制器的故障。如果系统中有 2 个控制器同时故障，用户数据一定会丢失。

此外，基于 DRAM 的写缓存还需要独立的电池和掉电保护软件，当系统断电时，在独立电池的支持下可以把 DRAM 中的写缓存数据转移到 SSD 上，以避免数据丢失。

现在，浪潮信息[4]、华为[5]两家公司的高端存储产品都在使用该技术，国际厂商如 EMC，HPE 在 2020 年以前也在使用该技术。

### 1.2 双控横向可扩展的全局缓存

由于循环镜像在多控制器场景下，软件设计和实现的难度太大，系统的扩展性受限。因此，2018 年，EMC 的 PowerMax 产品[6]改变了全局缓存的架构设计，变成了双控制器的横向扩展架构，如图 2 所示。

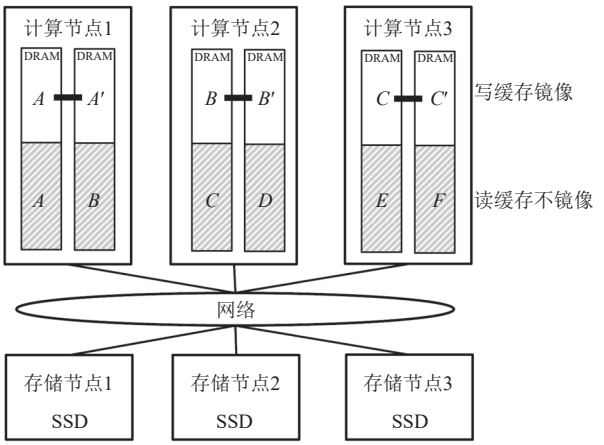



Fig. 2 EMC global cache diagram

图 2 EMC 全局缓存图

存储控制器内依然使用 DRAM 写缓存，控制器间依然使用 RDMA 传输 I/O。与循环镜像不同的是，写缓存只在每个存储框中的 2 个控制器间做镜像，写缓存镜像数据不会写入第 3 个控制器中。其结果是把全局缓存的高可用设计简化为 2 个控制器。

该架构的优点是存储的集群规模与全局缓存的复杂度无关，系统可以横向扩展。PowerMax 最多可以支持 16 个存储框和 32 个存储控制器。

该架构设计的缺点是只能支持 1 个框中的 1 个控制器故障，如果 1 个存储框中的 2 个控制器同时故障，用户的写入数据会丢失。EMC 的全局缓存设计是牺牲了系统的可用性来提高了存储集群的扩展性，用更大规模的存储集群来对外提供高性能。

### 1.3 基于非易失内存的全局缓存

2019 年成立的 VastData[7]公司在全局缓存技术上进行了创新。VastData 采用了非易失内存（storage class memory，SCM）和 NVMe-oF（NVMe over fabric）技术替代了 DRAM 和 RDMA。写缓存被保存在存储控制器外部的 SCM 上，任意一个控制器可以通过 NVMe-oF 协议来访问任意一块 SCM 上的写缓存数

据, 如图 3 所示。

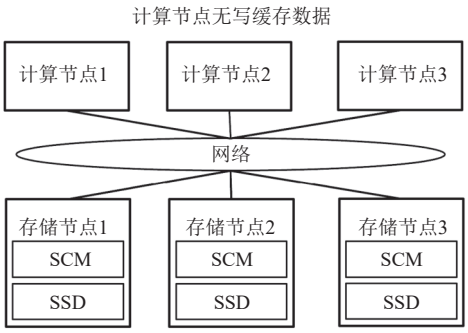



Fig. 3 Global cache based on SCM

图 3 基于 SCM 的全局缓存

基于 SCM 的全局缓存的优点是:

1) 缓存数据被解耦到了控制器外部, 控制器故障不会导致写数据的丢失, 硬件上可以做到控制器 $n$ 坏 $n-1$ 的高可用。

2) SCM 是非易失介质, 存储系统内无需增加单独的供电电池和无需掉电保护软件。

3) 避免了复杂的循环镜像技术。

该技术有 2 个缺点。一是性能差。SCM 的性能不如 DRAM, NVM-oF 的性能不如 RDMA。二是现在市场上没有好的 SCM 产品。Intel optane persistent memory 在 2024 年 12 月 31 日后不再接受订单, 2025 年 12 月 31 日后停止供货, 虽然市场上还有三星的 Z-NAND 和铠侠的 XL-Flash 可以作为 optane 的替代品, 但其性能均不如 optane。

HPE 的 Alletra MP B10000[8](2025 年以前, 产品名称为 greenLake for block storage)则更为激进, 直接去掉了 SCM 和其他形式的全局缓存, 所有 I/O 直接写入 SSD, 该方法的缺点是写性能更低。

表 1 列出了以上 3 种全局缓存技术的优缺点。

**Table 1 Advantages and Disadvantages of Typical Global Cache Technologies**

**表 1 典型全局缓存技术的优缺点**

| 特点 | 循环镜像 | 双控横向扩展 | SCM 全局缓存 |
|---|---|---|---|
| 性能 | 高 | 高 | 低 |
| 高可用 | 依次 $n$ 坏 $n-1$ | 任意双控中的 1 个故障 | 同时 $n$ 坏 $n-1$ |
| 扩展性 | 4 控制器 | 可横向扩展 | 可横向扩展 |
| 技术复杂度 | 高 | 中 | 低 |

## 2 相关工作

本节介绍 CXL 的相关产品和技术现状。虽然 2019 年就已经发布了 CXL 1.0 规范, 但是, 从 2022 年才开始出现 CXL 的相关硬件产品, 包括芯片、CPU、内存设备、交换机和 CXL SSD。

1) 芯片。澜起科技(Montage Technology)公司在 2022 年发布了世界首款 CXL 内存扩展控制器芯片 MXC[9], 该芯片支持 JEDEC DDR4 和 JEDEC DDR5 标准, 符合 CXL 2.0 规范, 支持 PCIe 5.0 的速率。

2) CPU。2023 年初, Inter 公司发布了支持 CXL 1.1 的 SPR(sapphire rapids) CPU, 随后又发布了支持 CXL 1.1 的 EMR(emerald rapids)CPU。2024 年第 3 季度, Inter 公司发布了 2 款支持 CXL 2.0 协议的 CPU, granite rapids 和 sierra forest(已更名为“至强 6”)。AMD 在 2023 年发布了支持 CXL 1.1 协议的 genoa CPU, 2024 年第 2 季度发布了支持 CXL 2.0 协议的第 5 代 EPYC 处理器“Turin”。

3) 内存设备。包括 AIC(add in card)扩展卡、E3.S 形态的内存模组和内存簇(just a bunch of memory, JBOM): 2022 年, 澜起科技公司在开发计算项目全球峰会(Open Compute Project Global Summit, OCP Global Summit)上发布了 CXL AIC 卡的评测板[10]。2022 年, 三星公司发布了容量为 512 GB、形态为 E3.S 的 CXL 内存模组[11]。2024 年, 三星公司发布了名为 CMM-B 的 CXL 内存盒模组[12], 浪潮信息公司自研了总容量为 16 TB 的 JBOM 原型产品 Amber[13]。

4) 交换机。2023 年 8 月, XConn 公司发布了首个支持 CXL 2.0 的交换机[14]。

5) CXL SSD 盘。三星公司在 2023 年发布了 CXL SSD[15], 该 SSD 提供了 2 种接口 CXL.mem 和 CXL.io, 用于访问盘内部的 DRAM 内存和 NAND, 并具备 3 种模式: 在非易失模式下, 应用通过 load/store 语义访问盘内的 DRAM, 当服务器断电时, 由外部电池供电, 服务器负责转移 DRAM 中的数据到盘内的 NAND 中; 在 NAND 模式下, 服务器使用 CXL.io 和 NVMe 协议直接访问盘内的 NAND; 在缓存模式下, 服务器通过 load/store 语义访问 DRAM 中的缓存数据, NAND 做容量层, 但是需要服务器自己做缓存的数据管理。该产品的成果发表在 HotStorage 2022[16]。

2023 年的闪存峰会(the Future of Memory and Storage, FMS)上, 三星、MemVerge 和 H3 等公司联合展示了 2 TB 的池化 CXL 内存系统解决方案 Pooled CXL Memory System[17]。该方案最多可以支持 8 台服务器, 允许主机按需使用内存资源。同时, 还配备了用于可视化、池化、分层、动态内存分配的软件。

在学术领域, CXL 相关的研究工作有: 内存池、

硬件测试、CXL SSD、跨机柜的 CXL 内存池、分层软件、共享软件、CXL 应用等。

在硬件方面，文献 [18−22] 在云环境下使用 CXL 内存池以提高内存的利用率，降低数据中心的成本。文献 [23−25] 是系统测试工作，Liu 等人[23] 在 5 个 CPU 平台下测试了 4 个 CXL 设备的 265 个工作负载，多数负载可以容忍 CXL 内存的延迟，性能下降不明显，这和本文的测试结果一致，CXL 全局缓存和 DRAM 内存环回性能相近。文献 [16, 26−27] 开展的是 CXL SSD 相关的研究工作。Wang 等人[28] 通过 CXL 和 RDMA 相结合的方式来实现跨机柜的大规模内存池。Wang 等人[29] 则是通过让 CXL 协议运行在以太网上来实现跨机柜的内存池。两者解决的都是 CXL 设备的传输距离约为 1 m 且只能在 1 个机柜内应用的问题。

在软件方面，文献 [30−32] 研究的是 Linux 系统中 DRAM 内存和 CXL 内存之间的高效分层，让更多的数据访问落在低时延的 DRAM 内存中。Zhong 等人[33] 在 CXL 2.0 上做了内存共享软件，采用了绕开 CPU 缓存的方法来解决 2.0 协议不支持缓存一致性的问题。Wang 等人[34] 研究哈希表的优化，通过存储更多元数据减少 CXL 内存访问。文献 [35−36] 研究的是把 CXL 内存应用到数据库。文献 [37−41] 应用大容量的 CXL 内存来解决 AI 训练过程中的性能问题。

## 3 基于 CXL 2.0 实现全局缓存的技术可行性

在存储阵列上使用 CXL 技术，要解决 3 个问题：一是，要让存储系统支持 CXL；二是，解决内存共享问题；三是，保证带宽够用，且不会成为系统瓶颈。针对第 1 点，我们开发了 Retimer 卡、系统固件和 BIOS。本节分析 CXL 内存共享和带宽。

### 3.1 CXL 内存共享

存储阵列是一个多控制器构成的小型集群系统，用户的 I/O 可以发送给任意一个控制器，然后控制器再访问全局缓存中的数据。多个控制器的 CPU 有可能在同一时间访问同一个 CXL 内存的物理地址，多个 CPU 的缓存有可能会保存同一个 CXL 数据的缓存，这产生了多个 CPU 缓存间以及 CPU 缓存与 CXL 内存间的缓存数据一致性问题。CXL 3.0 协议[1] 在 MESI 协议基础上设计了反向失效嗅探（back-invalidate snoop, BISnp）的机制来解决该问题：首先，CXL 内存芯片会使用部分内存空间记录各个 CPU 缓存的数据缓存情况。当芯片嗅探到 CPU 发来的写操作时，CXL 芯片会主动让所有缓存该数据 CPU 缓存都失效。然后，再允许 CPU 完成写操作，以此保证多个控制器的缓存和 CXL 内存的缓存数据一致性。

但是，现在还没有支持 CXL 3.0 的 CPU 和产品，CXL 2.0 不支持缓存一致性。常见的解决方法有 2 种：一是绕开 CPU 缓存[42]，如利用 Intel CPU 的 MTRR (memory type range registers)[43]，所有的内存数据都直接写入 CXL 内存。二是在向 CXL 写入数据前，用带外的方式，如 UDP，协调多个控制器间的读、写权限，当有 1 个 CPU 写 CXL 内存时，其他 CPU 停止访问该数据，以此保证缓存数据的一致性，如 Memverge 的 CXL 内存共享软件 Gismo[44]。

与上述 2 个方法不同，本文则是在 CXL 上层的存储软件利用数据分片来解决 CXL 内存共享问题。即让每个 I/O 进入到每个控制器前进行数据分片，保证每一个 I/O 只会被发送到某一个固定的控制器上，然后每个控制器再访问与自己相连的 CXL 内存。这样用户在使用存储的过程中，不会发生同一个 I/O 会经由不同控制器读、写同一个 CXL 物理内存的情况，也就消除了缓存不一致的现象。

具体做法是修改存储的服务器多路径软件，引入数据分片算法，在向存储发送 I/O 前，多路径软件根据 I/O 的 LBA 地址计算出目的控制器，然后发送 I/O。这样保证了在任何时刻，同一个地址的 I/O 都会发往同一个控制器，每个控制器只会访问自己的缓存，避免了不同的控制器访问同一缓存数据的情况。

### 3.2 带 宽

与 DRAM 相比，CXL 内存没有带宽优势[23-25]，即便是通过 CXL 交换机聚合了多条内存后，PCIe x8 的 CXL 内存的单向带宽也仅为 32 GBps，而 DDR4 3200 内存的单向带宽为 25.6 GBps，DDR5 4800 内存的单向带宽为 38.4 GBps，因此，CXL 内存带宽相当于 1 条内存的带宽。但是，即便这样，CXL 内存的带宽也可以满足企业级盘阵存储的需求。我们从与 RDMA 带宽对比和系统性能瓶颈 2 方面来分析。

表 2 给出了 PCIe 5.0 和 PCIe 6.0 的 CXL 内存带宽以及 RoCE（remote direct memory access over converged Ethernet）的带宽。可以看出，即便是本文使用的 PCIe 5.0 x8 的 CXL 内存带宽也达到了 32 GBps，好于在研存储产品使用的 RoCE 200 Gbps。

下面分析现有的企业盘阵产品的实际最大性能。存储性能委员会（Storage Performance Council, SPC）的 2025 年 2 月 SPC-1 基准评测榜[45] 排名第 1 的存储产品，在 39% 随机读和 61% 写的混合负载下，32 控

**Table 2 Bandwidth of PCIe and RoCE**

**表 2 PCIe 和 RoCE 的带宽**

| 高速互连技术 | 带宽 | 说明 |
|---|---|---|
| PCIe 5.0 x8 | 单向 32 GBps | CXL 2.0 |
| PCIe 5.0 x16 | 单向 64 GBps | CXL 2.0 |
| PCIe 6.0 x16 | 单向 128 GBps | CXL 3.0 |
| RoCE | 100 Gbps | 市场存储产品 |
| RoCE | 200 Gbps | 在研存储产品 |
| RoCE | 400 Gbps | 新的 RoCE |
| RoCE | 800 Gbps | 下一代 RoCE |

制器总 IOPS 为 3 000.276 5 万, 平均每个控制器是 93.76 万 IOPS, 32 控制器的峰值带宽是 350 GBps, 平均每个控制器是 10.9 GBps 带宽。而 PCIe 5.0 x8 CXL 内存的单向物理带宽为 32 GBps, 实测带宽为 21.6 GBps, 完全可以满足盘阵的性能需求。

从上面的分析可知, 虽然 CXL 内存的带宽不如 DRAM, 但其 PCIe 5.0 x8 的 21.6 GBps 的带宽仍然能满足存储的需求, 并不会成为系统性能的瓶颈。另一方面, 与现在广泛采用的 RoCE 技术相比, 不管是现在还是将来, CXL 都有带宽优势。

## 4 系统设计

图 4 展示了基于 CXL 技术的全局缓存的设计。从系统拓扑图来看, CXL 全局缓存更接近于 VastData 的 SCM 全局缓存设计, 同样是计算-缓存的分离式架构, 控制器可以独立横向扩展, 所有控制器都能访问到全局缓存中的数据。不同的是用分离式的内存替代 SCM 以及控制器使用更高效的 load/store 语义和 CXL 协议访问内存, 盘阵的性能会更高, 甚至好于目前性能最好的 RDMA 全局缓存。

CXL 内存把原位于控制器内部的缓存数据解耦到外部, 具备 4 个优点:

1) 扩展性更强。计算资源不足时增加控制器数量; 要提高系统可用性, 可以增加 JBOM 和缓存副本数量, 软件复杂度不变。CXL 2.0 交换机最多支持 8 个控制器, 存储集群最高支持 8 控。CXL 3.0 协议支持交换机级联, 对于存储来说集群规模几乎可以无限扩展。

2) 性能更高。在硬件方面, CXL 的时延仅为百纳秒, 好于 RDMA 的 2~3 μs。在软件方面, CXL 内存复制使用的是标准的函数 *memcpy*, 也可以采用汇编指令、Intel DSA(data streaming accelerator)硬件加速

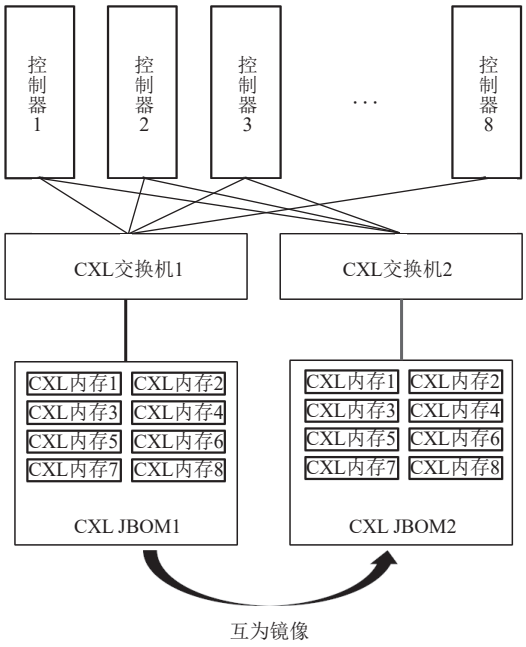



Fig. 4 Architecture diagram of CXL global cache

图 4 CXL 全局缓存架构图

库进行性能优化。而 RDMA 则需要运行驱动程序, CXL 的内存访问性能优势明显[22,28,46-48]。在构建分离式内存池方面, CXL 在性能、池化、缓存一致性上优于 RDMA[18,20]。在实际应用方面, 阿里 PolarDB 使用 CXL 内存池后, 吞吐量比 RDMA 内存池高 2.1 倍[49]。因此, 基于 CXL 在企业存储阵列上构建全局缓存, 性能会有明显的提升。

3) 可用性更高。传统存储使用控制器内的 DRAM 保存写缓存副本, 如果是 $m$ 副本的系统, 最多支持 $m-1$ 个控制器故障, 否则就会有丢失写缓存数据的风险。控制器和内存解耦后, 不管控制器故障多少个, 外部 JBOM 中的数据都不会丢失。因此, $n$ 个控制器可以同时故障 $n-1$ 个, 写缓存数据不丢失, 存储系统的服务不中断。另外, JBOM 本身是纯硬件, 没有存储软件, 其故障率仅受硬件影响, 系统故障率远低于控制器的故障率。

4) 故障处理过程比循环镜像更简单。无需考虑复杂的故障组合, 故障处理软件的复杂度与控制器数量无关, 解决了 RDMA 循环镜像的软件扩展难的问题。

下面我们从硬件、CXL 内存管理、写缓存设计、持续镜像 4 个方面说明系统设计。

### 4.1 硬件设计

本文在澜起科技公司的 CXL 芯片[9]基础上, 自研了 CXL JBOM 原型产品 Amber[13] 和 OCP(open compute project)形态的 Retimer 卡来实现存储的 CXL 全

局缓存硬件系统。

CXL JBOM 是 19 寸 2U 标准机箱，对外提供 16 个 x16 端口，每个端口对应 4 条容量为 256 GB 的 DDR5 4800 MTps 内存，总共支持 64 条内存，最大容量为 16 TB；单端口支持 1 TB 内存扩展；支持 2 个 CRPS（common redundant power supply）电源，1+1 冗余；外部线缆采用 CDFP DAC(c form-factor pluggable direct attach cable) 线缆互连。图 5 是实物图。



Fig. 5 Appearance diagram of JBOM hardware

图 5 JBOM 硬件外观图

每台 JBOM 内部由 2 块内存板构成，单块内存板具备 16 个 PCIe 5.0 x8 的澜起 MXC 芯片，每颗 MXC 芯片配 1 个 SPI（serial peripheral interface）Flash 用于存放芯片固件，支持 2 条内存实现内存拓展，单板总计支持 32 条内存。每 2 颗 MXC 的 x8 CXL 链路组成 1x16 的 CXL 链路，并通过 CDFP 连接器对外互连。CDFP 连接器同时提供同源时钟以及 PERST 信号，用于 MXC 芯片时钟输入及复位。同时板卡提供非同源时钟方案，提供更可靠的输入时钟源冗余特性。单板同时提供 debug 连接器用于调试。图 6 为其硬件拓扑图。

服务器上一般会使用 Retimer 卡把 PCIe 接口转成 CDFP 接口，然后通过 CDFP DAC 线缆连接到 CXL JBOM 或者交换机上。同时，Retimer 卡还能延长 PCIe 高速信号的传输距离。它通过重新定时（Retiming）信号来增强信号质量，避免传输过程中信号衰减导致的错误。经过 Retimer 卡后，控制器外部的线缆的可用长度从 0.9 m 延长到 1.1 m。但是，因为企业级存储采用的是定制化硬件，内部空间有限，无法使用标准服务器 Retimer 卡来连接 JBOM 和存储的控制器。因此，我们自研了 OCP 形态的 Retimer 卡，图 7 是实物图。

### 4.2 CXL 内存管理

高性能盘阵时延通常为数百微秒，低负载场景下可低至 100 μs。而通过函数 *numa_alloc_onnode* 向内核申请 4 KB CXL 内存的时延达 19.9 μs，难以满足高性能存储的需要。为此，本文设计了用户态内存管理模块，以减少内存分配的时间开销。

如图 8 所示，为满足存储的高性能需求，本文采用用户态方式统一管理 CXL 与 DRAM 内存。CXL

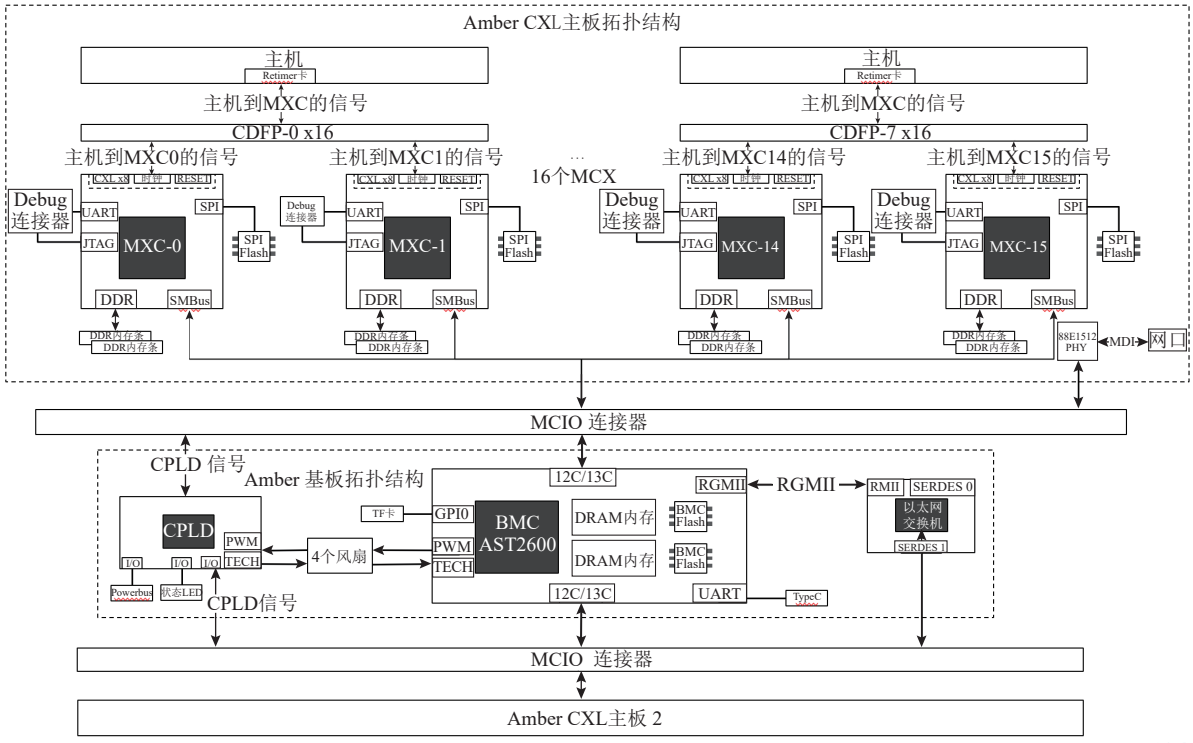



Fig. 6 Topology diagram of JBOM board

图 6 JBOM 主板拓扑结构图

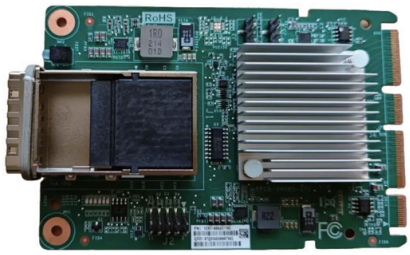


Fig. 7 Retimer card in OCP form factor

图 7 OCP 形态的 Retimer 卡

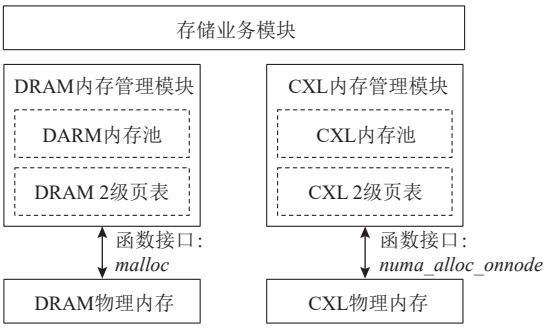



Fig. 8 Memory management module

图 8 内存管理模块

与 DRAM 内存管理模块相互独立, 内部软件设计则完全一致, 差异仅在于 CXL 模块的函数、变量名及 API 接口均带有“cxl_”前缀。

在系统初始化阶段, 2 个模块通过函数 *numa_alloc_onnode* 分别独立地向内核申请内存资源。随后, 系统在用户态以 4 KB 为单位管理内存空间, 并通过内存池机制向各软件模块提供预留的内存资源。

4.2.1 2 级页表设计

为了便于管理和扩展, 内存管理模块设计成了 2 级页表, 包括 pd(page descriptor)、1 级页表和 2 级页表, 最大可管理 2 TB 的内存空间。

1) pd 是指向 4 KB 内存页的指针, 大小为 8 B。

2) 1 级页表由若干个 4 KB 的 1 级页表项构成, 每个页表项保存 512 个指向 2 级页表项的 pd。1 级页表固定占用 1 MB 的空间, 有 256 个 1 级页表项, 可以保存 128K 个 pd。

3) 2 级页表由若干 2 级页表项构成, 每个 2 级表项大小为 32 KB, 2 级页表项内部由 4 096 个 pd 构成, 每个 pd 指向 1 个大小为 4 KB 的物理内存页。

4) 物理内存页每个内存页大小为 4 KB, 可以是 CXL 内存或者是 DRAM 内存。设计为 4 KB 的原因是企业硬盘的块大小一般是以 4 KB 为单位。每个物理内存页都在 2 级页表中对应 1 个 pd 以及 1 个页号。用户可以在 2 级页表数组中根据页号找到对应内存页的 pd。

pd、1 级页表、2 级页表、物理内存页的关系如图 9 所示。

2 级页表数据结构设计一共可以管理 512M 个 pd 和 4 KB 内存页, 1 个 DRAM 或 CXL 内存管理模块总共能管理 2 TB 内存, 2 级页表最多占用额外的 4 GB 内存, 1 级页表占用固定的 1MB 空间。

用户通过页面号访问 4 KB 内存页, 假设用户给的页面号为 $p$, 1 级表项的变量名为 *level*1_*memorytable*。当系统初始化完毕时, 用户拿到的 2 级页表项是一个连续的空间, 因此获得 $p$ 对应的 4 KB 页面指针的伪代码为:

void∗*ptr* = *level*1_*memtoryTable*[*p*/4096][*p*%4096] 。

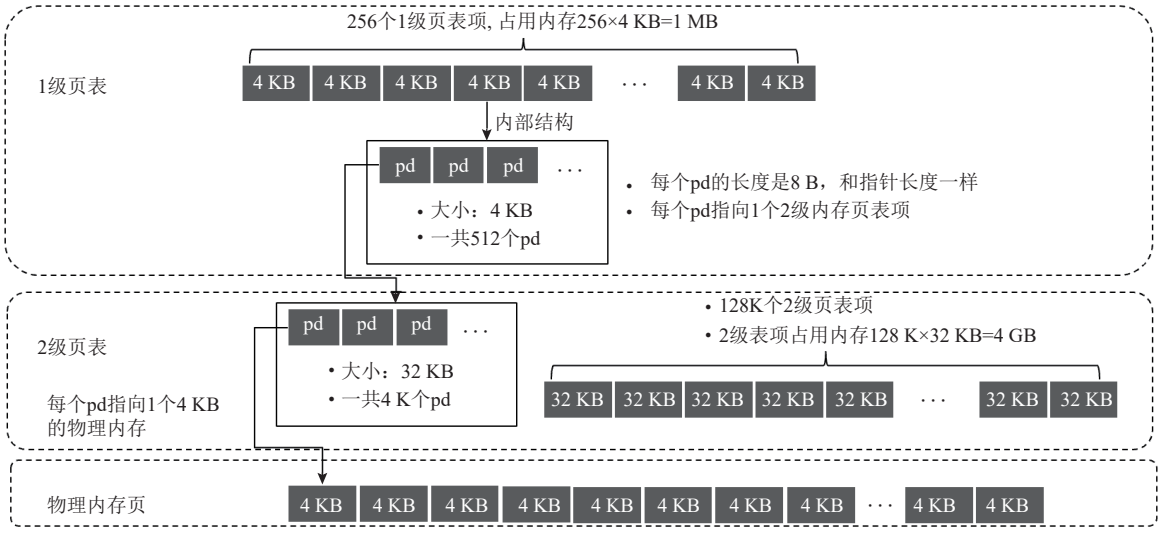



Fig. 9 Design of two-level memory page table

图 9 2 级内存页表设计

4.2.2 内存空间区域设计

为了管理和使用 DRAM 或 CXL 的内存页空间，我们把所有 4 KB 的内存页划分成了 5 个区域：pd 区域、general pages 区域、空闲页（free pages, FPS）区域、IOB 区域（32 KB 的大 IO 块）和 anchor 区域。其中，general pages 区域是业务模块能用到的内存空间，其他都是必要的管理 general pages 所需的空间。

1）pd 区域。对应 1 级和 2 级页表，总管所有的内存页。软件以 1 级页表为入口，可以访问到下述所有的内存区域。

2）general pages 区域。业务模块使用的 4 KB 内存空间。当业务模块向 DRAM 或 CXL 内存管理模块申请时，获得的是该区域的内存地址，存储软件运行过程中使用的也是该内存区域。

3）空闲页区域。管理“空闲”内存页，用于用户态内存的分配和回收。

4）IOB 区域（IOBlock）。向上层模块提供 32 KB 的大块内存。每个 IOB 由 8 个 4 KB 内存页构成，每个 IOB 都有 1 个 iob number。软件模块通过 iob number 拿到 IOB，最后从 pte 中拿到需要的内存页。

5）anchor 区域。保存的是内存管理模块的基础信息，包括可用页面数量、空闲页面数量、IOB 数量，以及存储控制器 id、集群信息等。anchor 占用固定的 32 KB 空间。

4.2.3 内存池设计

存储 I/O 处理是一个高并发过程，存在很多竞争条件。每个存储软件模块在运行阶段以内存池为单位，预留内存可以有效降低竞争和死锁的概率。

本文采用了树形结构的多级内存池来管理内存资源，树的每个节点是一个内存池。内存池的关键数据结构有：存储资源对象的链表、维护资源的计数等。

每个节点池都有各自的配额，配额即从内存管理模块中预留的内存空间。根节点是该资源类型的总池，每个模块可以从根节点建立多级子池，子池的配额是上级池或总池的一部分。

图 10 展示了 3 级内存池的结构。通常情况下，资源申请都在内存池的配额内，纯用户态运行，效率非常高。但是，当某个池上的资源申请超过其配额时，该池就会向父节点的内存池借用内存页，用完了再还回去。如果一直到根节点都没有借到，就要等待其他人释放了才能申请成功，这样就允许超配额使用资源，避免配额成为系统瓶颈。

### 4.3 写缓存设计

4.3.1 缓存双副本

由于企业级存储有高可用的需求，写缓存至少要存 2 份，以保证当 1 个数据缓存出现硬件故障后，数据不丢失。采用 CXL 内存后，有 2 种缓存双副本方案：方案 1 是 2 个 CXL JBOM 各存 1 个副本；方案 2 是 DRAM 内存 1 个副本和 CXL1 副本。

在现阶段，方案 2 具有明显的优势。理由如下：

1）DRAM 的性能好于 CXL，时延和带宽都优于 CXL。但是，这不是主要因素。因为，硬件时延只相差 150~350 ns，*memcpy* 的总时间相差不到 2 μs，这对于总时延接近 1 ms 的存储系统来说，影响不大。前文分析过带宽，CXL 带宽相比 DRAM 带宽虽然不高，但不是存储系统的性能瓶颈。

2）CXL 不支持 DMA（direct memory access）。企业存储在访问前、后端网卡，或者访问控制器内的

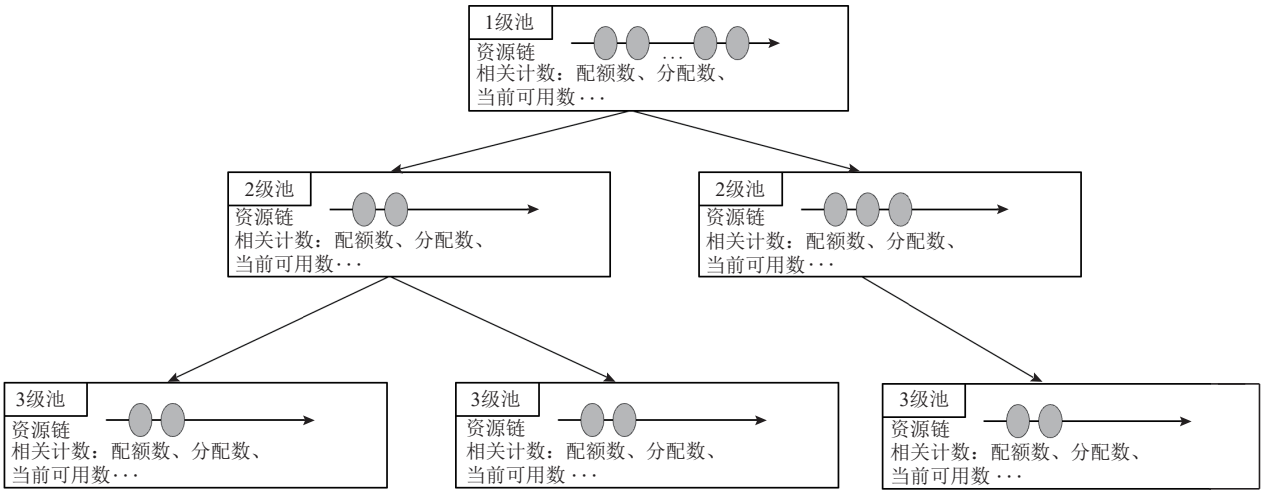



Fig. 10 Three-level memory pool

图 10 3 级内存池

SSD 盘时，使用的都是 DMA。但是，现在的 DMA 并不支持 CXL。因此，如果要维持之前系统的功能，使用网卡的 DMA，写数据时要先由网卡写入到 DRAM，然后用 *memcpy* 复制到 CXL 内存。读数据时，先把 CXL 内存中的缓存数据复制到 DRAM 中，然后才能返回给网卡。在落盘时也是一样的过程。很明显，这个过程中，各多了 1 次内存拷贝。

为了提高性能和减少不必要的数据拷贝，我们采用了方案 1 DRAM+CXL 的方式。用户的写 I/O 先经过网卡的 DMA 写入到 DRAM，然后软件调用函数 *memcpy* 复制 1 份到 CXL 内存。后续缓存数据落盘时，把 DRAM 的缓存数据写入到 SSD 中，在正常 I/O 过程中，该方式不会带来额外的内存复制开销。CXL 内存起到备份作用，当控制器故障时，CXL 内存中的缓存数据保证 I/O 可以继续进行，用户数据不丢失。

与使用 2 个 CXL 内存的方案 1 相比，方案 2 的劣势是要同时管理 DRAM 和 CXL 缓存数据，软件实现相对复杂。

#### 4.3.2 I/O 流设计

图 11 对比了 RDMA 全局缓存和 CXL 全局缓存的 I/O 处理过程。图 12 是 RDMA 缓存镜像的 I/O 流程图。本节通过对比，说明 CXL 全局缓存的优势。下面简述 CXL 全局缓存过程。

① 主机经过数据分片的 Hash 计算，知道该 I/O 应发往控制器 1。

② 写缓存模块把 I/O 保存在 DRAM 中，然后通过函数 *memcpy* 把数据复制到 CXL 内存中，同时调用系统，把这段数据刷写到 CXL 内存中，不要仅仅保存在自己的 CPU 缓存里。此时 1 份数据保存在 DRAM 内存中，另一份数据保存在 CXL 内存中。

③ 控制器 1 响应主机：数据写入成功。

④ 开始把写缓存数据异步刷写到硬盘。

⑤ 异步刷写成功，内存中的数据被写入到了硬盘中。

⑥ 脏数据转非脏：DRAM 的数据已经写到了硬盘中，CXL 中的副本数据无需再次下刷，但是要转换数据状态为非脏。

从以上流程可以看出，控制器 1 没有和其他控制器有任何的交互，如果把控制器增加到 4 个甚至 8 个，缓存的拓扑结构也不会变化，因此系统具备很好的横向扩展能力。图 11 右图的②中也只是执行了 1 次函数 *memcpy*，系统开销小。与图 12 左侧的 RDMA 缓存镜像相比，控制流程大幅度简化，因此性能有较大幅度的提升。

需要进一步说明的是，图 12 左侧②和③的数据传输过程包含 3 次 RDMA 交互。之所以如此设计，是因为 RDMA 作为一种异步的、绕过对端 CPU 的高性能网络通信技术，与存储缓存镜像所需的同步机制存在差异，缓存镜像要求同步处理，且需要对端 CPU 参与本地数据处理。因此，需进行以下开发工作。

1）异步改同步。当 1 份缓存数据成功写入到对端控制器的内存中后，需要告知发送端写入成功。

2）缓存写入成功后，要唤醒对端的 CPU 进行相关逻辑的处理。实现上，我们采用的是轮训机制。

上述 2 个因素致使向对端写入缓存数据需经历 3 次 RDMA 传输，而且轮询操作会消耗多余的 CPU

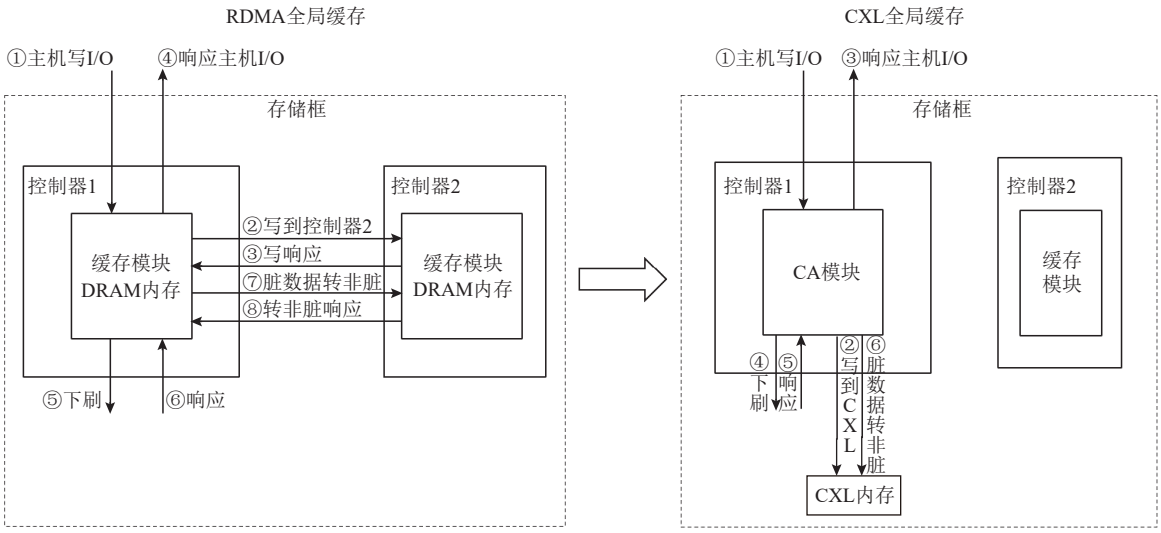



Fig. 11 I/O flowchart of CXL global cache

图 11 CXL 全局缓存的 I/O 流程图

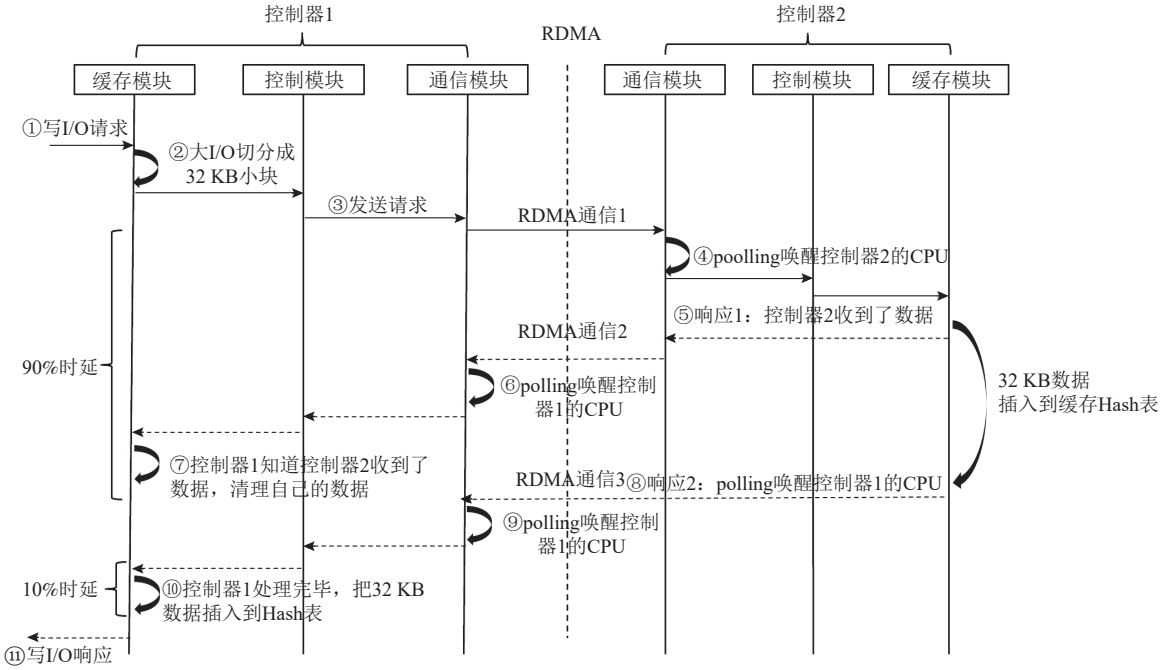



Fig. 12　I/O flowchart of RDMA cache mirroring

图 12　RDMA 缓存镜像的 I/O 流程图

资源。而 CXL 作为同步写入技术，只需 CPU 的 store 指令返回，就表明数据已成功写入 CXL 内存中，这极大简化了 RDMA 的缓存镜像处理流程。因此，CXL 全局缓存的性能提升不仅仅是由硬件的 2~3 μs 降低到 0.35 μs。

### 4.4　持续镜像设计和可用性

文献 [46] 阐述了面向 4 控集群，基于 RDMA 的持续镜像技术原理，实际的多控产品的情况要更复杂。

图 13 展示了 4 控存储系统发生节点依次故障时的拓扑视图变化。每次控制器故障，存储都要经历节点配对、传输镜像数据 2 个主要过程。配对过程是要为每个数据块找到新的配对节点，一般 5 s 以内完成，在此过程中 I/O 流会中断。配对完成后，原来的副本节点对外提供 I/O 服务。数据传输过程会把故

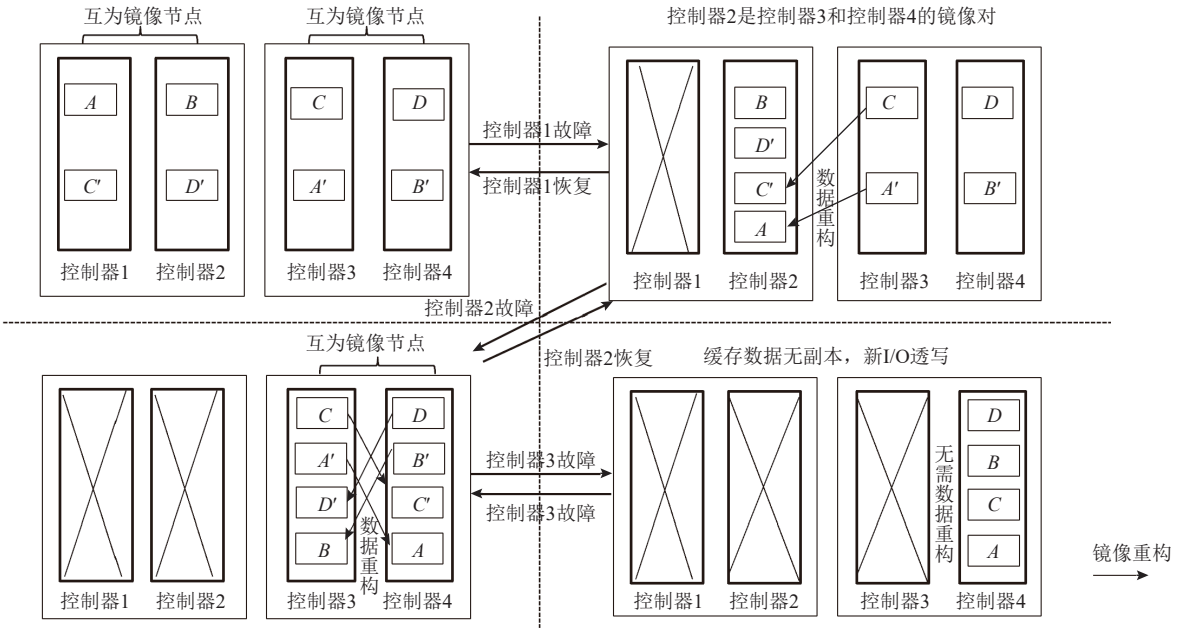



Fig. 13　Fault topology diagram of four controller cache mirroring

图 13　4 控缓存镜像的故障拓扑图

障节点的几十到上百 GB 的缓存副本传输到新的镜像节点上。该过程耗时数十秒,并消耗很多 CPU 资源,新配对节点在完成数据传输后才能提供 I/O 服务。

此外,循环镜像还要处理其他相关复杂问题:及时下刷缓存脏数据;严格保证缓存副本数据的一致性;当节点恢复时,新加入的节点会多 1 个副本,要处理短暂的“3 副本”状态;控制器重启会在短时间内引发 2 次重新结对,数据来回迁移 2 次,我们称之为“震荡”,软件上要设计防“震荡”策略。

如果存储集群规模变大,还需要解决跨框的“多控循环镜像”的难题。常见的做法是把 8 个控制器连在了一起,没有实现真正的跨框 32 控循环镜像。此时,如果 1 个框故障,缓存数据就会丢失。有的厂商在另外 1 个控制框中增加了第 3 副本[5],容忍了 1 个控制框的故障。缺点:1)实现复杂;2)多了 1 次框间数据传输,降低了存储性能。

CXL 全局缓存可以大大简化多控缓存镜像的实现逻辑,如图 14 所示。当有 1 个或多个控制器同时故障时,集群的主控节点依据负载均衡的原则,让其他控制器直接接管故障控制器的 CXL 缓存数据即可实现,难度大大降低。同时,在这个过程中无需进行数据迁移或复制。

本文在理论上计算循环镜像、CXL 全局缓存、32 控集群等不同全局缓存架构的可用性,以说明 CXL 全局缓存在高可用方面的优势。表 3 汇总了全局缓存技术的存储集群可用性。

根据控制器或控制框的数量,以及同时允许故障的控制器数量来计算存储的可用性。$n$ 个控制器可以视作 $n$ 次独立重复试验,每个节点的试验只有“故障”和“正常”两种结果,而且每个控制器的故障相互独立,而且故障概率均为 $p$。因此,$m$ 个控制器同时故障的概率符合二项分布,可以用式(1)估算 $n$

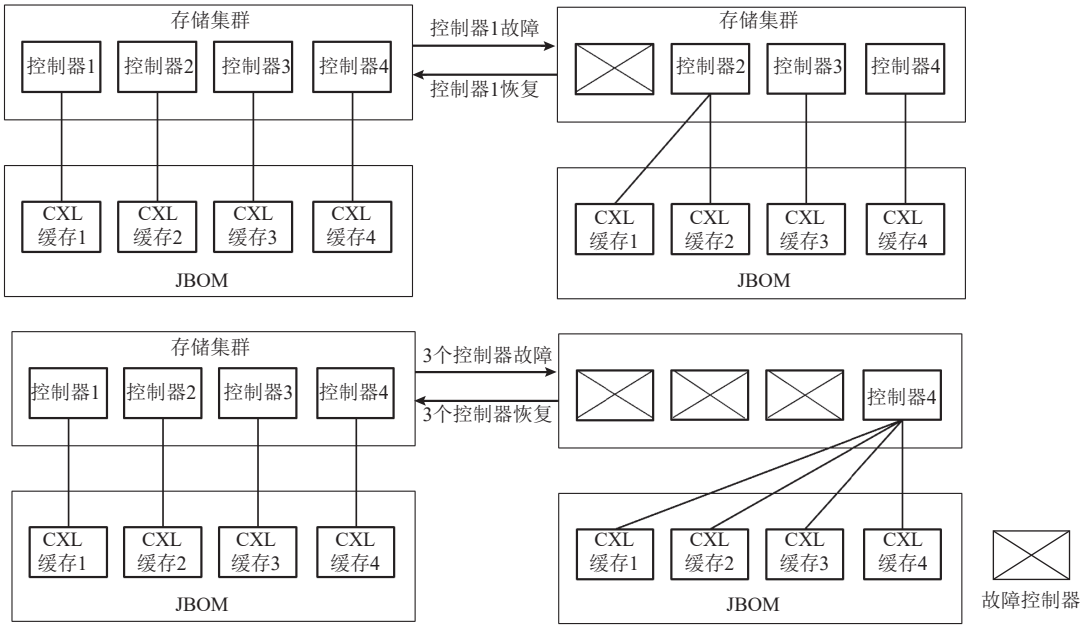



Fig. 14 Fault topology diagram of CXL global cache mirroring

图 14 CXL 全局缓存镜像的故障拓扑图

**Table 3 Availability of Storage Clusters with Different Global Cache Designs**

**表 3 不同全局缓存设计的存储集群可用性**

| 编号 | 集群/全局缓存设计 | 控制器数量 | 代表厂商 | 可用性水平 | 系统可用性/% |
|---|---|---|---|---|---|
| 1 | 1 框 2 控 | 2 | 多数厂商 | 2 坏 1 | 99.999 999 |
| 2 | 1 框 4 控 | 4 | 多数厂商 | 4 坏 1 | 99.999 994 |
| 3 | 4 控,分离式 CXL 缓存 | 4 | 本文 | 4 坏 3 | 99.9…9 小数点后 16 个 9 |
| 4 | 8 控,分离式 CXL 缓存 | 8 | 本文 | 8 坏 7 | 99.9…9 小数点后 32 个 9 |
| 5 | 1 框 2 控,横向可扩展 | 32 | EMC | 框内 2 坏 1,任何存储框不可故障 | 99.999 98 |
| 6 | 1 框 4 控,8 框集群 | 32 | 多数厂商 | 框内 4 坏 1,任何存储框不可故障 | 99.999 952 |
| 7 | 1 框 4 控,8 框集群,框间 3 副本 | 32 | 华为 | 框内 4 坏 1,存储框 8 坏 1 | 99.999 999 999 989 92 |

个节点最多同时有 $m$ 个故障的概率。多数存储产品都用缓存双副本，因此 $m$=1。1 框 2 控和 1 框 4 控存储的可用性均按照式(1)直接计算。

$$A=\sum_{i=0}^{m}\mathrm{C}_n^i(p)^{n-i}(1-p)^i\ 。\tag{1}$$

CXL 全局缓存硬件虽然是由控制器和 CXL 设备构成，但是我们认为 CXL 设备是纯硬件，可用性极高，只需要根据控制器的可用性估算系统可用性。CXL 全局缓存的可用性特点是 $n$ 个控制器允许 $n$−1 个同时故障，其可用性采用式(2)计算。

$$A=1-(1-p)^n\ 。\tag{2}$$

表 3 第 5 行和第 6 行的 32 控存储中，单个控制框的可用性按照式(1)计算。由于框间没有写缓存数据副本，任何一个框的故障都会导致写 I/O 的数据丢失。因此，该设计下任何存储框都不能故障，其集群可用性计算为式(3)，其中 $q$ 为单个框的可用性。

$$A=q^n\ 。\tag{3}$$

8 框 32 控 3 副本存储(第 7 行)的实际效果是支持框内控制器 4 坏 1，集群最多支持 1 个框故障。因此，可用性计算方法是先按照式(1)计算单个框的可用性，然后再次使用式(1)计算 8 框的可用性。

高端存储的可用性一般为 5~6 个 9[4-8]，单控制器可用性一般为 4 个 9，即 $p$ = 99.99%。在此基础上，表 3 计算了不同控制器数量、不同全局缓存设计的集群可用性。

从表 3 可知，如果单节点的可用性均为 4 个 9，CXL 全局缓存的可用性是最高的，4 控可达 16 个 9，8 控集群的理论可用性达 32 个 9。CXL 全局缓存支持 $n$ 个控制器同时坏 $n$−1，在所有的全局缓存设计中，其可用性水平也是最高的。

需要说明，可用性和可靠性是存储产品系统设计的一个方面，但也要考虑性能、容量、可扩展性和实现难度。更大规模的 16 控、32 控集群的优点是：多控制器并发处理 I/O 会为用户提供更高的性能，更多的控制器能管理更多数量的硬盘，为用户提供更多的卷数量和更大的卷空间。因此，EMC 在其高端产品中使用了 1 框 2 控横向可扩展的架构，没有采用可用性更高的框间 3 副本设计。框间 3 副本的问题：一是使用的 DRAM 内存数量多、成本高；二是相同规模的集群性能差，多复制 1 份数据会带来额外的数据传输，同时框间的传输速率更低。本文设计的 CXL 全局缓存能保证高可用的同时提升性能，有很强的技术先进性和应用价值。

## 5 实验评测

### 5.1 实验环境

本文基于浪潮信息存储产品实现了用户态内存管理模块与 CXL 全局缓存。实验环境部署了 2 台服务器、1 台浪潮信息存储设备、2 台 JBOD 及 1 台 JBOM，硬件拓扑结构如图 15 所示。

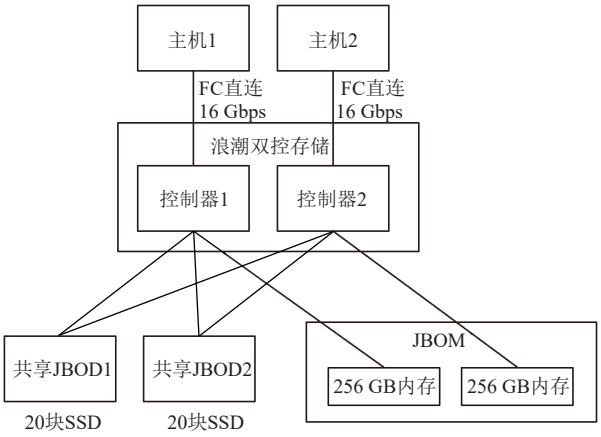



Fig. 15 Physical topology diagram of the experimental environment

图 15 实验环境的物理拓扑图

测试服务器是 2 台服务器：一台 CPU 为 Intel® Xeon® Platinum 8358，主频为 2.6 GHz，内存为 312 GB；另一台 CPU 为 Intel® Xeon® Gold 6330，主频为 2.0 GHz，内存为 512 GB。2 台服务器均使用 16 GBps 的 FC 卡与存储直连。

存储使用的是浪潮信息双控存储产品 AS18000，每个控制器具有 2 路 CPU 和 256 GB 内存，CPU 型号为 INTEL® XEON® GOLD 6454S，主频为 2.2 GHz，支持 CXL 1.1，可以访问 CXL 2.0 的内存设备。

后端的存储扩展柜是 2 台共享 JBOD，一共有 40 块 1.92 TB 的 SAS SSD 盘，型号为三星 PM1643。

JBOM 使用的是自研的 Amber 产品，使用的是 SK 海力士 DDR5 内存，每条内存的容量为 128 GB，主频为 5 600 MHz。每个控制器通过 1 条 CDFPDAC 线缆连接到 JBOM，可以访问到 2 条 DDR5 内存，共 256 GB。

### 5.2 CXL 硬件性能测试

首先，我们使用 Intel MLC(memory latency checker)工具[50]测试 JBOM 的硬件基础性能，测试结果如图 16 所示。JBOM 的时延约是 DRAM 近端 numa 节点时延的 3.36 倍，远端 CXL 内存时延比近端时延多

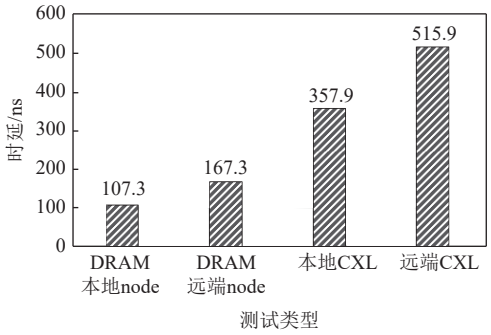
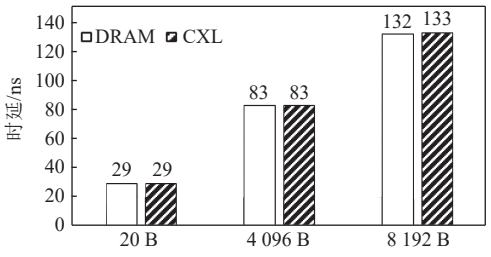



Fig. 16 Hardware performance of CXL JBOM

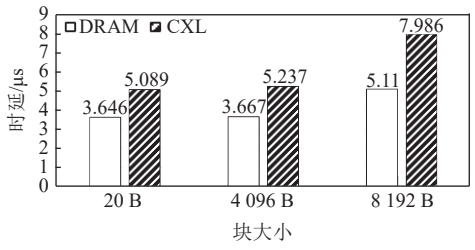
图 16 CXL JBOM 的硬件性能

了 158 ns。

存储缓存软件在访问 CXL 内存时，主要通过函数 *memcpy* 把 4 KB 缓存数据从 DRAM 复制到 CXL 中。因此，本文开展了相关验证测试工作，将测试分为 CPU 缓存命中和缓存不命中 2 组，测试结果如图 16 所示。之所以设置 2 组测试，是因为在顺序写入或用户数据存在热点的情况下，CPU 缓存会出现大量命中现象，这会导致函数 *memcpy* 的性能发生显著变化。图 17 是 DRAM 内存和 CXL 内存的 *memcpy* 函数测试结果。




(a) CPU缓存命中时的内存时延




(b) CPU缓存不命中时的内存时延

Fig. 17 *memcpy* performance of DRAM and CXL memory

图 17 DRAM 和 CXL 内存的 *memcpy* 性能

图 17(a)的测试结果表明，当 CPU 缓存命中时，DRAM 内存和 CXL 内存的性能基本没有差别。图 17(b)展示了不同数据块大小、缓存不命中时的 2 种内存的 *memcpy* 性能。缓存不命中的测试方法为：在测试 CXL 内存前，先在 1 块超过 CPU 缓存总容量的大块内存上运行函数 *memcpy* 或 *memset*，随后再执行 DRAM 或 CXL 内存的 *memcpy* 操作。由于此时 CPU 缓存已被大块内存数据完全覆盖，因此测得的时延值必然是缓存不命中的测试结果。此外，测试过程中需避免重复访问同一内存地址，若 *memcpy* 操作针对同一内存地址执行，时延会大幅降至 0.3 μs。图 17(b) 的测试结果显示，通过调用函数 *memcpy* 后，DRAM 与 CXL 的性能差距显著缩小。尽管 CXL 的硬件时延是 DRAM 的 3.34 倍，但在使用 *memcpy* 复制 4 KB 数据块时，CXL 时延仅为 DRAM 的 1.42 倍。因此，在复杂软件系统中，若对 CXL 内存数据进行有限次的复制操作，并不会对系统性能造成影响。

### 5.3 内存申请性能测试

图 18 展示了使用 malloc、numa 及本文 4.2 节开发的用户态内存管理模块进行内存申请的性能测试对比结果。测试数据表明：若通过 numa_alloc_onnode 接口申请 CXL 内存，时延高达 19.907 4 μs，是 malloc 内存时延的 4.73 倍。该数值在总时延为 100 μs 的高端存储系统中占比过高，会显著影响存储性能。而通过用户态内存管理模块申请内存，时延可降至 1.31 μs，优化效果显著。

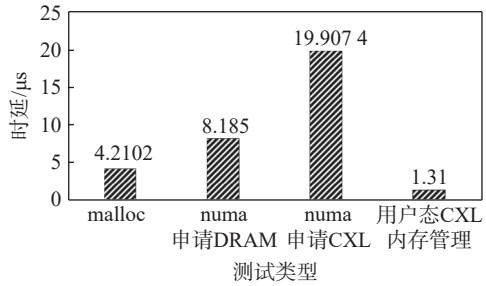


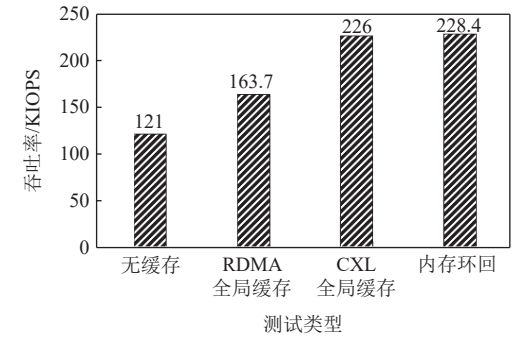

Fig. 18 Test result of applying memory

图 18 申请内存的测试结果

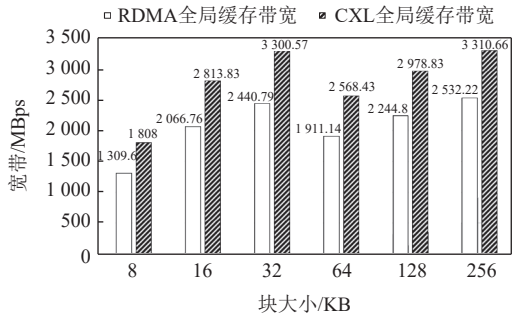
### 5.4 CXL 全局缓存性能测试

我们采用金融、通信行业的存储集中采购测试方案，对比 RDMA 全局缓存、CXL 全局缓存的性能，重点关注 8 KB 小 I/O 随机写场景下的 IOPS 性能。系统配置为：48 个卷，每个卷 500 GB，采用分布式 RAID5，条带宽度为 9，SSD 盘数量为 40。测试的软件采用 vdbench 5.04.07，随机写，预热时间和测试时间都是 5 min。性能评价方法为：调整 vdbench 的线程数，在保证时延不超过 1 ms 的条件下，存储 IOPS 的最大值即为测试结果。图 19 是全局缓存的性能测试结果。

图 19(a)比较了无缓存、RDMA 全局缓存、CXL 全局缓存、内存环回的性能。无缓存代表了 1.3 节无缓存架构的性能值。RDMA 缓存代表了 1.1 节循环镜像和 1.2 节双控横向扩展架构的性能值。内存环




(a) 8 KB, 100%随机写的性能测试结果




(b) 100%随机写、不同块大小的性能测试结果

Fig. 19 Test result of CXL global cache performance

图 19 CXL 全局缓存性能测试结果

回指的是在本地 DRAM 中做数据副本，同时保留相关的软件逻辑，环回代表了双副本全局缓存的性能极限。测试结果表明 RDMA 全局缓存比无缓存的性能提升了 35.29%，如果取消内存缓存，那么存储的性能会下降。CXL 全局缓存的性能比 RDMA 全局缓存的性能提升了 38%，同时仅比内存环回性能低 0.24%，接近双副本系统的性能上限。

CXL 全局缓存和内存环回性能相近的原因：一是 CXL 的软硬件时延和 DRAM 相近；二是在复杂的存储软件栈中，多几次 CXL 和 DRAM 的内存拷贝，对性能影响极小。

图 19(b)展示了不同块大小的性能测试结果，性能提升幅度在 30.7%~38% 之间。图中 64 KB 带宽比 32 KB 下降的原因是线程数减少了一半。如果 64 KB 测试用例的线程数和 32 KB 的线程数一样，那么时延会显著超过 1 ms，这不符合企业客户对全闪存储的测试要求。

## 6 产品化面临的挑战

虽然本文的研究工作证实了 CXL 全局缓存的技术优势和性能表现。但是，大规模产品化仍面临以下挑战。

1）集成到存储内部。自研的 CXL JBOM 和交换机[13]均是独立的 2U 设备。而 4 控企业存储的高度为 4U，与之相比 CXL 硬件占用的空间太大。因此，存储产品需要把 JBOM 和交换机都小型化并集成到内部。对于双控存储来说，因为写缓存容量不需要太大，只需要少量内存和 2 个 CXL 芯片，而且 CXL 芯片现在已支持双端口，无需额外配置交换机。因此，双控存储内部集成 CXL 是可行的。但是，4 控、8 控集群必须要使用交换机才能实现控制器 4 坏 3 和 8 坏 7，需要在现有的存储上增加 2U 的高度，系统总高度将达到 6U。

2）热插拔。这是产品化过程中最具挑战性的环节。由于 CXL 设备位于存储外部，所以不管是人为拔除线缆，还是内存或 CXL 芯片故障，若没有热插拔功能，就会导致存储系统的崩溃。实现热插拔需要服务器、OS、存储软件协同解决：服务器硬件要能感知到 CXL 设备故障，然后生成中断给 OS，OS 完成内核的故障处理后通知存储软件，最后由存储软件完成应用侧异常处理，并保证不再访问该段内存。尽管 CXL 2.0 规范的 9.9 章节对热插拔实施流程给出了简要说明，但实现该功能属于系统级工程，难度极大。

3）DMA 设备对 CXL 协议的支持适配。存储是通过网卡的 DMA 收发 I/O，数据落盘时也要通过磁盘控制器的 DMA 把内存数据写入到盘中。但是，当前的网卡和磁盘控制器都不支持 CXL 协议，因此如果将全部缓存数据都放在 CXL 内存上，则需要在 DRAM 与 CXL 内存之间进行 1 次数据拷贝才能完成存储的 I/O 流程。若 DMA 设备能够支持 CXL 协议，存储系统便可减少 1 次内存拷贝操作。

## 7 结 论

CXL 是目前性能最好的高速互连技术[47-48]，本文提出了一种企业级盘阵存储的 CXL 全局缓存设计，在浪潮信息存储产品上实现了系统原型。该设计充分利用了 CXL 内存的分离式和高性能特征，将存储写缓存从控制器内解耦到外部，具有以下技术优势：控制器故障不会导致缓存数据丢失，可以实现真正的控制器横向扩展、$n$ 个控制器同时故障 $n-1$、性能大幅提升。4.4 节中的分析证明了这是当前可用性最高的全局缓存技术。实验数据表明，CXL 全局缓存比 RDMA 全局缓存的写性能提升了 38%。

下一步工作有 3 个方面：1)把 CXL 内存和交换机集成到存储内部。2)解决 CXL 设备的热插拔问题，

确保 CXL 设备突然被拔出时,存储 I/O 不中断,提升系统可用性。3)把存储的自精简、重删元数据从 SSD 迁移到大容量的 CXL 内存上,把集群数据放在 CXL 内存中,提高存储的性能。

**作者贡献声明:**李强负责方案整体设计、实验环境搭建和测试工作,并撰写论文;刘涛制定研究方向,指导整体研究路线;张雪庆完成了可用性分析工作,并提出论文修改意见;孟宪伟负责内存管理模块的开发工作;孙明刚负责缓存模块的软件开发工作;李博乐完善论文内容,负责论文结构的讨论与修改;陈曦完成了 CXL JBOM 的设计和开发工作。

## 参 考 文 献

[1] CXL. CXL® specification - compute express[EB/OL]. [2025-05-24]. https://computeexpresslink.org/cxl-specification

[2] Shu Jiwu. Data Storage Architecture and Technology[M]. Beijing: Posts & Telecom Press, 2023: 50−58 (in Chinese)
(舒继武, 数据存储架构与技术[M]. 北京: 人民邮电出版社, 2023: 50−58)

[3] Wu Chentao. Information Storage and IT Management[M]. Beijing: Posts & Telecom Press, 2015 (in Chinese)
(吴晨涛, 信息存储与 IT 管理[M]. 北京: 人民邮电出版社, 2015)

[4] IEIT. IEIT high-end storage system HF18000G6-CGF[EB/OL]. [2025-05-24]. https://www.ieisystem.com/product/storage/15598.html (in Chinese)
(浪潮信息. 浪潮信息高端存储系统 HF18000G6-CGF[EB/OL]. [2025-05-24]. https://www.ieisystem.com/product/storage/15598.html)

[5] Huawei. OceanStor Dorado V6 high-end all-flash storage product technical white paper[EB/OL]. [2025-05-24]. https://e.huawei.com/cn/material/storage/all-flash-storage/42b9aab381c743b1b68c73db12931e3d (in Chinese)
(华为. OceanStor Dorado V6 高端全闪存存储产品技术白皮书[EB/OL]. [2025-05-24]. https://e.huawei.com/cn/material/storage/all-flash-storage/42b9aab381c743b1b68c73db12931e3d)

[6] EMC. A detailed overview of the next generation PowerMax platform[EB/OL]. [2025-05-24]. https://infohub.delltechnologies.com/static/media/client/7phukh/DAM_3ddff5ea-f554-4502-9304-59a014d9daea.pdf

[7] VastData. White paper: The VAST data platform[EB/OL]. [2025-05-24]. https://www.vastdata.com/whitepaper

[8] HPE. HPE Alletra storage MP B10000 architecture[EB/OL]. [2025-05-24]. https://www.hpe.com/psnow/doc/a50008302enw?jumpId=in_psnow_ee8f0a3a-72f8-47de-bf38-7b94e0de4c77_gaiw

[9] Montage. Montage technology delivers the world's first CXL™ memory eXpander controller[EB/OL]. [2025-05-24]. https://www.montage-tech.com/Press_Releases/20220506

[10] Roland K. Montage technology CXL add in card[EB/OL]. [2025-05-24]. https://drive.google.com/file/d/1bpVLz533F2YTVqN3mIqtr--ngiR35lBe/view?usp=share_link

[11] Samsung. Samsung electronics introduces industry's first 512GB CXL memory module[EB/OL]. (2022-05-10)[2025-05-24]. https://semiconductor.samsung.com/us/news-events/news/samsung-electronics-introduces-industrys-first-512gb-cxl-memory-module/

[12] Samsung. CMM-B: Memory pooling appliance for rack computing [EB/OL]. (2022-05-10)[2025-05-24]. https://semiconductor.samsung.com/us/about-us/us-office/us-r-and-d-labs/memory-labs/cxl-memory-module-box/

[13] Gao Xianyang, Chen Xi, Wang Yun. Investigating the memory extension and disaggregated memory pooling system[C]//Proc of the 4th Int Conf on Electronic Information Engineering and Computer Science. Piscataway, NJ: IEEE, 2024: 667−671

[14] XConn. XConn technologies launches the industry's first and only hybrid CXL 2.0 and PCIe gen 5 switch[EB/OL]. (2023-08-08)[2025-05-24]. https://www.prnewswire.com/news-releases/xconn-technologies-launches-the-industrys-first-and-only-hybrid-cxl-2-0-and-pcie-gen-5-switch-301895313.html?tc=eml_cleartime

[15] Samsung. Samsung memory-semantic CXL SSD[EB/OL]. (2022-08-18)[2025-05-24]. https://www.servethehome.com/samsung-memory-semantic-cxl-ssd-at-fms-2022-powered-by-amd-xilinx/

[16] 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

[17] Memverge. Memory pooling and sharing solution[EB/OL]. (2023-08-08)[2025-05-24]. https://memverge.com/samsung-memverge-h3-platform-and-xconn-demonstrate-memory-pooling-and-sharing-for-endless-memory/

[18] 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. New York: ACM, 2023: 574−587

[19] Berger D S, Ernst D, Li H, et al. Design tradeoffs in CXL-based memory pools for public cloud platforms[J]. IEEE Micro, 2023, 43(2): 30−38

[20] Gouk D, Lee S, Kwon M, et al. Direct access, high-performance memory disaggregation with directCXL[C]//Proc of USENIX Annual Technical Conf 2022. Berkeley, CA: USENIX Association, 2022: 287−294

[21] Sharma D D. Compute express link (CXL): Enabling heterogeneous data-centric computing with heterogeneous memory hierarchy[J]. IEEE Micro, 2022, 43(2): 99−109

[22] Gouk D, Kwon M, Bae H, et al. Memory pooling with CXL[J]. IEEE Micro, 2023, 43(2): 48−57

[23] Liu Jinshu, Hadian H, Xu Hanchen, et al. Systematic CXL memory characterization and performance analysis at scale[C]// Proc of the 30th ACM Int Conf on Architectural Support for Programming Languages and Operating Systems. New York: ACM, 2025: 1203−1217

[24] 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

[25] 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. New York: ACM, 2023: 105−121

[26] 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. New York: ACM, 2023: 24−30

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

[28] 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

[29] 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. Piscataway, NJ: IEEE, 2023: 75−82

[30] 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

[31] 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. New York: ACM, 2023: 742−755

[32] Xiang Lingfeng, Lin Zhen, Deng Weishu, et al. Nomad: Non-exclusive memory tiering via transactional page migration[C]//Proc of the 18th USENIX Symp on Operating Systems Design and Implementation. Berkeley, CA: USENIX Association, 2024: 19−35

[33] 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. Berkeley, CA: USENIX Association, 2024: 37−56

[34] Wang Hancheng, Dai Haipeng, Chen S, et al. Rethinking Hash tables: Challenges and opportunities with compute express link (CXL)[C]//Proc of the ACM Turing Award Celebration Conf-China 2024. New York: ACM, 2024: 23−27

[35] Huang Wentao, Sha Mo, Lu Mian, et al. Bandwidth expansion via CXL: A pathway to accelerating in-memory analytical processing[C]//Proc of the 16th Int Workshop on Accelerating Analytics and Data Management Systems Using Modern Processor and Storage Architectures. New York: ACM, 2024: 1−12

[36] 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]. Proceedings of the VLDB Endowment, 2024, 17(12): 3827−3840

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

[38] Park S S, Kim K S, 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. Piscataway, NJ: IEEE, 2024: 970−982

[39] 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. Piscataway, NJ: IEEE, 2024: 382−395

[40] 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. Piscataway, NJ: IEEE, 2022: 1−11

[41] 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 USENIX Annual Technical Conf 2023. Berkeley, CA: USENIX Association, 2023: 585−600

[42] Ma Teng, Liu Zheng, Wei Chengkun, et al. HydraRPC: RPC in the CXL era[C]//Proc of USENIX Annual Technical Conf 2024. Berkeley, CA: USENIX Association, 2024: 387−395

[43] Reichard G, Luis R R. MTRR(memory type range register) control [EB/OL]. [2025-05-24]. https://www.kernel.org/doc/html/next/x86/mtrr.html

[44] Memverge. Gismo: World's first CXL-based multi-server shared memory[EB/OL]. (2023-05-23)[2025-05-24]. https://memverge.com/memverge-unveils-worlds-first-CXL-based-multi-server-shared-memory-at-isc/

[45] Storage Performance Council. Storage performance council[EB/OL]. [2025-05-24]. https://www.storageperformance.org

[46] Shu Jiwu. Technical prospects of new compute-storage disaggregation architecture[J]. CCF Communications, 2022, 18(11): 53−60 (in Chinese)
(舒继武, 新型存算分离架构技术展望[J]. 中国计算机学会通讯, 2022, 18(11): 53−60)

[47] Shu Jiwu, Chen Youmin, Wang Qing, et al. Progress on storage systems for disaggregated data centers[J]. SCIENTIA SINICA Informationis, 2023, 53(8): 1503−1528 (in Chinese)
(舒继武, 陈游旻, 汪庆, 等. 分离式数据中心的存储系统研究进展[J]. 中国科学: 信息科学, 2023, 53(8): 1503−1528)

[48] Wang Jing, Li Chao, Wang Taolei, et al. Survey of disaggregated memory: Cross-layer technique insights for next-generation datacenters[J]. arXiv preprint, arXiv: 2503.20275, 2025

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

[50] Intel. Intel memory latency checker (MLC)[EB/OL]. [2025-05-24]. https://www.intel.cn/content/www/cn/zh/download/736633/736634/intel-memory-latency-checker-intel-mlc.html



**Li Qiang**, born in 1982. PhD, professor. His main research interests include computer architecture, storage system, and memory pooling.

**李　强**, 1982 年生。博士, 正高级工程师。主要研究方向为计算机体系结构、存储系统、内存池化。



**Liu Tao**, born in 1983. Master, senior engineer. His main research interests include centralized storage architecture and hardware design.

**刘　涛**, 1983 年生。硕士，高级工程师。主要研究方向为集中式存储架构、硬件设计。



**Zhang Xueqing**, born in 1984. PhD candidate, assistant professor. His main research interests include storage system and system architecture.

**张雪庆**, 1984 年生。博士研究生，助理研究员。主要研究方向为存储系统、系统架构。



**Meng Xianwei**, born in 1986. Master. His main research interests include storage system design and memory pooling.

**孟宪伟**, 1986 年生。硕士。主要研究方向为存储系统设计、内存池化。



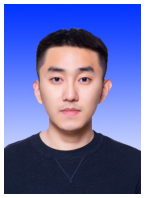
**Sun Minggang**, born in 1984. Bachelor, senior engineer. Member of CCF. His main research interests include storage software architecture and storage system performance.

**孙明刚**, 1984 年生。学士，高级工程师。CCF 会员。主要研究方向为存储软件架构、存储系统性能。



**Li Bole**, born in 1984. Master. His main research interests include storage technology and hardware design.

**李博乐**, 1984 年生。硕士。主要研究方向为存储技术、硬件设计。



**Chen Xi**, born in 1993. Master. Member of CCF. His main research interests include server system architecture and hardware design.

**陈　曦**, 1993 年生。硕士。CCF 会员。主要研究方向为服务器系统架构、硬件设计。