# Memory Disaggregation: Open Challenges in the Era of CXL

Hasan Al Maruf, Mosharaf Chowdhury
SymbioticLab, University of Michigan

## Abstract

Compute and memory are tightly coupled within traditional datacenter servers. Large-scale datacenter operators have identified this coupling as a root cause behind fleet-wide resource underutilization and increasing Total Cost of Ownership (TCO). With the advent of ultra-fast networks and cache-coherent interfaces, *memory disaggregation* has emerged as a potential solution, whereby applications can leverage available memory even outside server boundaries. In this paper, we discuss some open challenges from a software perspective toward building next-generation memory disaggregation systems leveraging emerging cache-coherent interconnects.

## 1 Cache-Coherent Interconnects

Compute Express Link (CXL) [2] is a new processor-to-peripheral/accelerator cache-coherent interconnect protocol that builds on and extends the existing PCIe protocol by allowing coherent communication between the connected devices.[1] It allows cache-line granularity access to the connected devices and underlying hardware maintains cache-coherency and consistency. With PCIe 5.0, CPU-to-CXL interconnect bandwidth is similar to the cross-socket interconnects on a dual-socket machine [18]. CXL adds around 50-100 nanoseconds of extra latency over normal DRAM access.

**CXL Roadmap.** Today, CXL-enabled CPUs and memory devices support CXL 1.0/1.1 that enables a point-to-point link between CPUs and accelerator memory or between CPUs and memory extenders. CXL 2.0 spec enables one-hop switching that allows multiple accelerators without (*Type-1 device*) or with memory (*Type-2 device*) to be configured to a single host and have their caches be coherent to the CPUs. It also allows memory pooling across multiple hosts using memory expanding devices (*Type-3 device*). A CXL switch has a fabric manager (it can be on-board or external) that is in charge of the device address-space management. Devices can be hot-plugged to the switch. A virtual switch partitions the CXL-Memory and isolate the resources between multiple hosts.

CXL 3.0 adds multi-hop hierarchical switching – one can have any complex types of network through cascading and fan-out. CXL 3.0 supports PCIe 6.0 (64 GT/s i.e., up to 256 GB/s of throughput for a x16 duplex link) and expand the horizon of very complex and composable rack-scale server design with varied memory technologies. A new Port-Based Routing (PBR) feature provides a scalable addressing mechanism that supports up to 4,096 nodes. Each node can be any of the existing three types of devices or the new Global Fabric Attached Memory (GFAM) device that supports different types of memory (i.e., Persistent Memory, Flash, DRAM, other future memory types, etc.) together in a single device. Besides memory pooling, CXL 3.0 enables memory sharing across multiple hosts on multiple end devices. Connected devices (i.e., accelerators, memory expanders, NICs, etc.) can do peer-to-peer communicate bypassing the host CPUs. In essence, CXL 3.0 enables large networks of memory devices.

## 2 CXL-Disaggregated Memory at Rack-Scale and Beyond: Open Challenges

Next-generation memory disaggregation systems that operate between rack-scale and a little beyond, may experience the following non-exhaustive list of challenges.

### 2.1 Abstractions

**Memory Access Granularity.** CXL enables cache-line granular memory access over the connected devices, whereas existing OS VMM modules are designed for page-granular (usually, 4KB or higher) memory access. Throughout their lifetimes, applications often write a small part of each page; typically only 1-8 cache-lines out of 64 [8]. Page-granular access causes large dirty data amplification and bandwidth overuse. In contrast, fine-grained memory access over a large memory pool causes high meta-data management overhead. Based on an application's memory access patterns, remote memory abstractions should support transparent and dynamic adjustments to memory access granularity.

**Memory-QoS Interface.** Traditional solutions for memory page management focus on tracking (a subset of) pages and counting accesses to determine the heat of the page and then moving pages around. While this is enough to provide a two-level, hot-vs-cold QoS, it cannot capture the entire spectrum of page temperature. Potential solutions include assigning a QoS level to (1) an entire application; (2) individual data structures; (3) individual `mmap()` calls; or even (4) individual memory accesses. Each of these approaches have their pros and cons. At one extreme, assigning a QoS level to an entire application maybe simple, but it cannot capture time-varying page temperature of large, long-running applications. At the other end, assigning QoS levels to individual memory accesses requires recompilation of all existing applications as

[1]Prior industry standards in this space such as CCIX [1], OpenCAPI [6], Gen-Z [4] etc. have all come together under the banner of CXL consortium. While there are some related research proposals (e.g., [10]), CXL is the de facto industry standard at the time of writing this paper.

well as cumbersome manual assignments, which can lead to erroneous QoS assignments.

### 2.2 Management and Runtime

**Memory Address Space Management.** Memory located on a CXL device can either be mapped as Host-managed Device Memory (HDM) or Private Device Memory (PDM). To update the memory address space for connected devices to different host devices, a system reset is needed; traffic towards the device needs to stop to alter device address mapping during this reset period. An alternate solution to avoid this system reset is to map the whole physical address space to each host when a CXL-device is added to the system. The VMM or fabric manager in the CXL switch will be responsible to maintain isolation during address-space management. How to split the whole address-space in to sizable memory blocks for the efficient physical-to-virtual address translation of a large memory network is an interesting challenge [10, 17].

**Unified Runtime for Compute Disaggregation.** CXL Type-2 devices maintains cache coherency with the CPU. CPU and Type-2 devices can interchangeably use each other's memory and both get benefited. In such a setup, remote memory abstractions should track the availability of compute cores and efficiently perform near-memory computation to improve the overall system throughput. Besides, future datacenters will likely be equipped with numerous domain-specific compute resources/accelerators. In such a heterogeneous system, one can borrow the idle cores of one compute resource and perform extra computation to increase the overall system throughput. A unified runtime to support malleable processes that can be immediately decomposed into smaller pieces and offloaded to any available compute nodes can improve both application and cluster throughput [14, 16].

### 2.3 Allocation Policies

**Memory Allocation in Heterogenous NUMA Cluster.** For better performance, hottest pages need to be on the fastest memory tier. However, due to memory capacity constraints, it may not always be possible to utilize the fastest or performant memory tier. Determining what fraction of memory is needed at a particular memory tier to maintain the desired performance of an application at different points of its life cycle is challenging. This is even more difficult when multiple applications coexist. Efficient promotion or demotion of pages of different temperatures across memory tiers at rack scale is necessary. One can incorporate a lightweight but effective algorithm to select the migration target considering node distances from the CPU, load on CPU-memory bus, current load on different memory tiers, network state, and the QoS requirements of the migration-candidate pages.

**Allocation Policy for Memory Bandwidth Expansion.** For memory bandwidth-bound applications, CPU-to-DRAM bandwidth often becomes the bottleneck and increases the average memory access latency. CXL's additional memory bandwidth can help by spreading memory across the top-tier and remote nodes. Instead of only placing cold pages into CXL-Memory, which has low bandwidth consumption, an ideal solution should place the right amount of bandwidth-heavy, latency-insensitive pages to CXL-Memory.

**Memory Sharing and Consistency.** CXL 3.0 allows memory sharing across multiple devices. Through an enhanced coherency semantics, multiple hosts can have a coherent copy of a shared segment, with back invalidation for synchronization. Memory sharing improves application-level performance by reducing unnecessary data movement and improves memory utilization. Sharing a large memory address space, however, results in significant overhead and complexity in the system that plagued classic distributed shared memory (DSM) proposals [12]. Furthermore, sharing memory across multiple devices increases the security threat in the presence of any malicious application run on the same hardware space.

### 2.4 Rack-Level Objectives

**Rack-Scale Memory Temperature.** To obtain insights into an application's expected performance with multiple temperature tiers, it is necessary to understand the heat map of memory usage for that application. Existing hot page identification mechanisms [3, 5, 11, 15] are limited to a single host OS or user-space mechanism. So far, there is no distributed mechanism to determine the cluster-wide relative page temperature. Combining the data of all the OS or user-space tools and coordinating between them to find rack-level hot pages is an important problem. CXL fabric manager is perhaps the place where one can get a cluster-wide view of hardware counters for each CXL device's load, hit, and access-related information. One can envision extending Chameleon [11] for rack-scale environments to provide observability into each application's per-device memory temperature.

**Energy- and Carbon-Aware Memory Disaggregation.** Datacenters represent a large and growing source of energy consumption and carbon emissions [7]. Some estimates place datacenters to be responsible for 1-2% of aggregate worldwide electricity consumption [9, 13]. To reduce the TCO and carbon footprint, and enhance hardware life expectancy, datacenter rack maintain a physical energy budget or power cap. Rack-scale memory allocation, demotion, and promotion policies can be augmented by incorporating energy-awareness in their decision-making process. In general, we can introduce energy-awareness in the software stack that manage compute, memory, and network resources in a disaggregated cluster.

## 3 Conclusion

With diverse cache-coherent interconnects finally converging under the CXL banner, the entire industry are at the cusp of taking a leap toward next-generation software-hardware codesigned disaggregated systems. This will not only simplify and better implement previous-generation memory disaggregation solutions but also open up new possibilities.

## References

[1] CCIX. https://www.ccixconsortium.com/.

[2] Compute Express Link (CXL). https://www.computeexpresslink.org/.

[3] DAMON: Data Access MONitoring Framework for Fun and Memory Management Optimizations. https://www.linuxplumbersconf.org/event/7/contributions/659/attachments/503/1195/damon_ksummit_2020.pdf.

[4] Gen-Z. https://genzconsortium.org/.

[5] Idle page tracking-based working set estimation. https://lwn.net/Articles/460762/.

[6] OpenCAPI. https://opencapi.org/.

[7] T. Anderson, A. Belay, M. Chowdhury, A. Cidon, and I. Zhang. Treehouse: A case for carbon-aware datacenter software. In *HotCarbon*, 2022.

[8] I. Calciu, M. T. Imran, I. Puddu, S. Kashyap, H. A. Maruf, O. Mutlu, and A. Kolli. Rethinking software runtimes for disaggregated memory. In *ASPLOS*, 2021.

[9] N. Jones. How to stop data centres from gobbling up the world's electricity. *Nature*, 561:163–166, 2018.

[10] S.-s. Lee, Y. Yu, Y. Tang, A. Khandelwal, L. Zhong, and A. Bhattacharjee. MIND: In-network memory management for disaggregated data centers. In *SOSP*, 2021.

[11] H. A. Maruf, H. Wang, A. Dhanotia, J. Weiner, N. Agarwal, P. Bhattacharya, C. Petersen, M. Chowdhury, S. Kanaujia, and P. Chauhan. TPP: Transparent page placement for CXL-enabled tiered-memory. In *ASPLOS*, 2023.

[12] B. Nitzberg and V. Lo. Distributed shared memory: A survey of issues and algorithms. *Computer*, 24(8):52–60, 1991.

[13] F. Pearce. Energy hogs: Can world's huge data centers be made more efficient? *Yale Environment*, 2018.

[14] Z. Ruan, S. J. Park, M. K. Aguilera, A. Belay, and M. Schwarzkopf. Nu: Achieving microsecond-scale resource fungibility with logical processes. In *NSDI*, 2023.

[15] Vladimir Davydov. Idle Memory Tracking. https://lwn.net/Articles/639341/.

[16] J. You, J. Wu, X. Jin, and M. Chowdhury. Ship compute or ship data? why not both? In *NSDI*, 2021.

[17] Z. Yu, Y. Zhang, V. Braverman, M. Chowdhury, and X. Jin. NetLock: Fast, centralized lock management using programmable switches. In *SIGCOMM*, 2020.

[18] W. Zhao and J. Ning. Project Tioga Pass Rev 0.30 : Facebook Server Intel Motherboard V4.0 Spec. https://www.opencompute.org/documents/facebook-server-intel-motherboard-v40-spec.