CMS: Logical System Architecture

Version 1.0

Date: Oct 17, 2023

Author (s) :
Anjaneya Reddy Chagam, Intel Corporation

Reviewers (s):
Manoj Wadekar, Meta
Vikrant Soman, Uber
Siamak Tavallaei, Independent

Executive Summary

This document defines the **Open Compute Project (OCP) Composable Memory Systems (CMS)** Logical Architecture. CMS is an emerging paradigm that facilitates dynamic and unified memory management across diverse memory technologies, interconnects, and hierarchies. The CMS logical system architecture serves as a vital reference point for system designers, data center operators, and end customers as they navigate the landscape of commercial implementations. This entails selecting an ideal combination of host processor platforms, memory devices, chassis configurations, and rack setups. The initial CMS logical architecture primarily targets general-purpose applications running on CPUs and employs CXL interconnects to enable pooled memory use-cases.

Table of Contents

1. Compliance with OCP Tenets **4**

1.1 Openness 4

1.2 Efficiency 4

1.3 Impact 4

1.4 Scale 4

1.5 Sustainability 4

2. Introduction **4**

3. CMS Logical System Architecture **7**

Direct Attached 8

Direct Attached using a transport other than CXL/PCIe Physical Layer 9

Multi-Headed 9

Fabric 10

Network 11

Data Center Memory Fabric Manager (DCMFM) 11

4. Memory Buffer Provisioning **12**

5. Memory Management **13**

6. Conclusion **15**

7. Glossary **16**

8. References **17**

9. License **18**

10. About Open Compute Foundation **19**

1. Compliance with OCP Tenets

1.1 Openness

The OCP CMS Logical System Architecture was developed in dialogue with the entire OCP CMS community membership and the resulting document represents the initial logical architecture. CMS usage models are nascent, so this document will evolve over time, and that this architecture document represents the wisdom of the collective ecosystem on the deployment scenarios for creating CXL based composable memory systems.

1.2 Efficiency

The OCP CMS Logical System Architecture lays the foundation for developing CMS physical architecture for various deployments highlighting hardware/software co-design.

1.3 Impact

Having a common foundational logical architecture helps us to guide CMS workstream scope and follow consistent architecture to develop future white papers, specifications, case studies and vendor reference implementations.

1.4 Scale

CMS Logical System Architecture fosters consistent vendor neutral composable solutions and enables high degree of reusability using open eco-system software, hardware form factor specifications and best practices.

1.5 Sustainability

CMS software and hardware co-design solutions are designed to optimize performance and resource utilization for underlying physical memory technologies thereby achieving data center energy efficiency.

2. Introduction

In a rapidly expanding enterprise and hyperscale market, the role of memory has become increasingly crucial. Data-intensive applications like Artificial Intelligence (AI), Machine Learning (ML), and databases are pushing the boundaries of server performance. Newer memory technologies, interconnects, and hierarchies that promise enhanced memory bandwidth, capacity, increased throughput, improved data sharing, and greater scalability, all catering to the demands of these data-intensive applications. These hardware innovations are designed to optimize performance and resource utilization for underlying physical memory technologies.

The rise in ever-growing computational performance is driving the demand for greater memory performance, while diverse application categories require heightened memory capacity to overcome platform limitations. Design engineers are actively seeking balanced system solutions that harness the potential of new technologies, products, and software capabilities to enable these applications to leverage memory technologies to the fullest. The advent of new hardware, software solutions, and memory abstractions is expected to unlock novel value propositions for memory use-cases, pushing the boundaries of location transparency, automated tiering across hybrid memory technologies, and the realization of pooled capacities with unified namespaces.

In Figure 1 below, we illustrate the interaction of general-purpose software applications running on CPUs or specialized applications like AI operating on GPUs, both of which rely on load/store memory access semantics. The concept of "near memory" refers to memory that is closest to the CPU/GPU, including elements such as in-package High-Bandwidth Memory (HBM) and host system DRAM connected to local or remote sockets. Conversely, "far memory" resides farther away from the CPU/GPU and encompasses memory buffers with Compute Express Link (CXL) connected DRAM, Storage Class Memory (SCM), compressed memory pages in block devices, or pooled memory distributed over CXL fabrics. To optimize performance, frequently accessed data is typically stored in near memory, while less frequently accessed data is relegated to far memory.

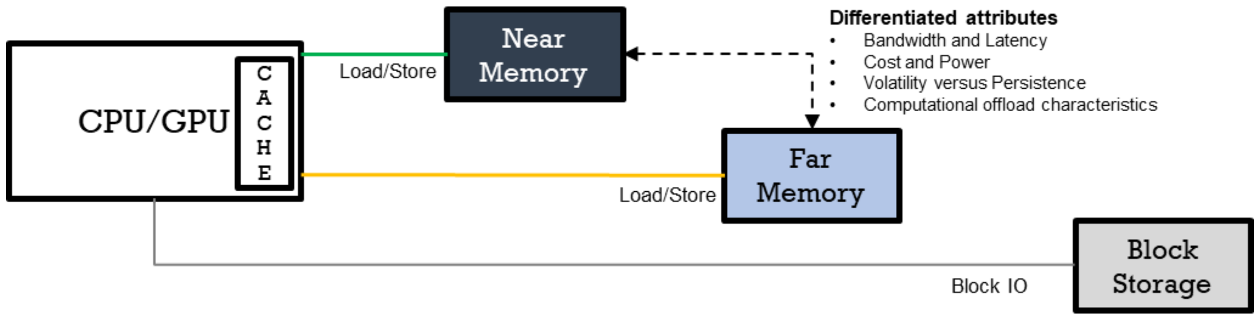


Figure 1: CMS Memory Hierarchy - Logical View

CMS is an emerging paradigm that facilitates dynamic and unified memory management across diverse memory technologies, interconnects, and hierarchies. This approach addresses the complex demands of memory performance, capacity, latency, throughput, data sharing, scalability, and security posed by emerging applications, all without requiring a server reboot. The CMS logical system architecture serves as a vital reference point for system designers, data center operators, and end customers as they navigate the landscape of commercial implementations. This entails selecting an ideal combination of host processor platforms, memory devices, chassis configurations, and rack setups. The initial CMS logical architecture primarily targets general-purpose applications running on CPUs and employs CXL interconnects to enable pooled memory use-cases.

3. CMS Logical System Architecture

CMS logical architecture includes abstractions and interfaces between the various hardware and software components required to implement composable memory systems targeted for diverse use cases such as virtualization, caching/databases and AI/ML. A key enabler in CMS architecture is the Compute Express Link (CXL), an open and industry-supported cache-coherent interconnect designed to facilitate seamless communication among processors, memory expansion resources, and accelerators. CXL introduces memory expansion capabilities using load/store memory semantics, specifically through the cxl.mem protocol, which interfaces with the host system and the underlying operating system or hypervisor runtime. In addition, CXL.io provides device and fabric control plane management that can be used to integrate into data center orchestration for data center fleet management.

The CMS Logical Architecture in Figure 2 below illustrates five distinct types of composable system solutions that harness the data plane capabilities of CXL.mem and the control plane abstractions provided by the CXL standard. These solutions encompass: 1) Direct Attached, 2) Multi-Headed, 3) Fabric, and 4) Network configurations, and 5) Direct-attached using a transport other than CXL/PCIe physical layer. Each of these configurations tailors the composable memory system to specific use cases and requirements.

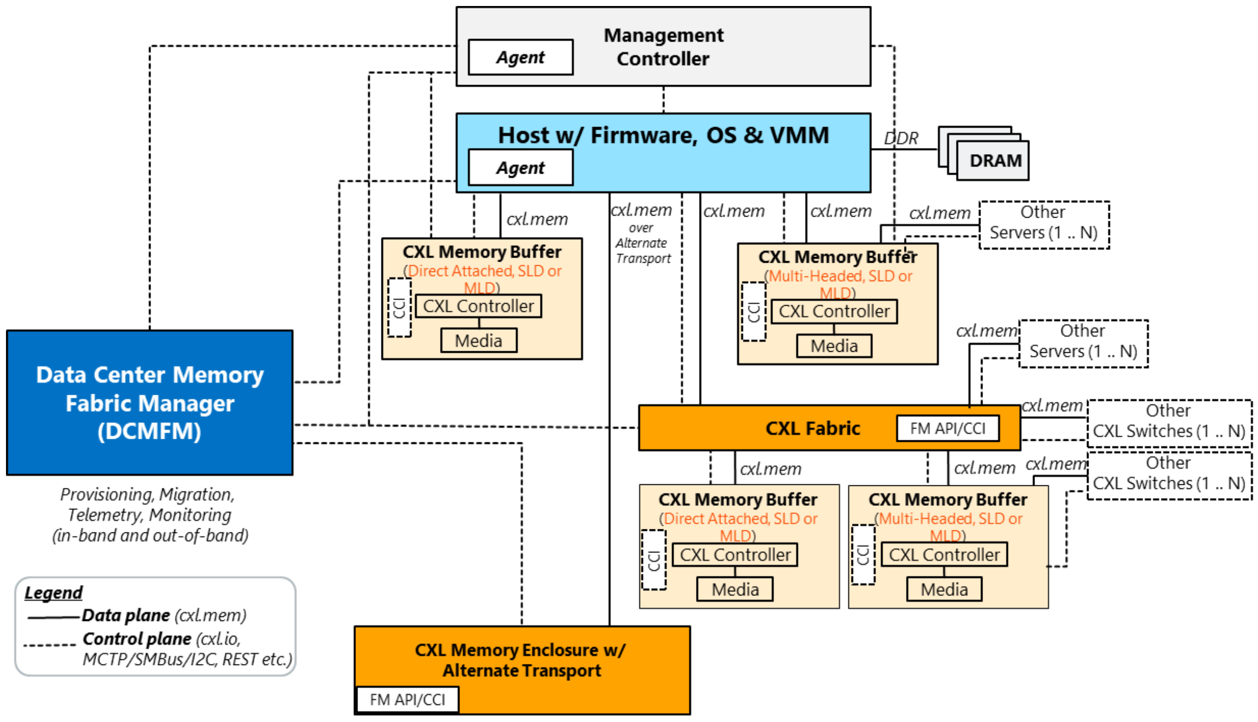

Management Controller
Agent
Host w/ Firmware, OS & VMM
Agent
DDR
DRAM
cxl.mem
cxl.mem over Alternate Transport
CXL Memory Buffer
(Direct Attached, SLD or MLD)
CCI
CXL Controller
Media
CXL Memory Buffer
(Multi-Headed, SLD or MLD)
Other Servers (1 .. N)
Data Center Memory Fabric Manager (DCMFM)
Provisioning, Migration, Telemetry, Monitoring (in-band and out-of-band)
CXL Fabric
FM API/CCI
Other CXL Switches (1 .. N)
Legend
Data plane (cxl.mem)
Control plane (cxl.io, MCTP/SMBus/I2C, REST etc.)
CXL Memory Enclosure w/ Alternate Transport
FM API/CCI

Figure 2: CMS Logical System Architecture

Direct Attached

This represents the most straightforward memory expansion capability, wherein a CXL device is directly linked to the host platform, exclusively serving the host it's physically connected to. CXL memory buffers can utilize a variety of memory technologies like DDR4, DDR5, HBM, or SCM. The management of a CXL device can be achieved through an in-band agent within the host or through out-of-band protocols using a management controller. The choice of the preferred management method for CXL device management is left to system vendors. A CXL memory buffer exposes host-managed device memory (HDM). Host firmware maps HDM memory to the system's coherent address space, facilitating access from the host using standard write-back memory semantics. HDM memory can be configured by host firmware as either special-purpose memory or conventional memory.

During boot time, the Operating System kernel configures CXL HDM special-purpose memory as a DAX (Direct Access) device. DAX command-line utilities allow promotion and demotion of memory in and out of kernel-managed memory once the operating system is booted. Conventional memory is provisioned as kernel-managed memory in either scenario. CXL HDM regions are presented as memory-only NUMA domains. The Operating System or Application Software is expected to comprehend the CXL memory's capacity, bandwidth, and latency attributes within the CXL tier and implement efficient tiering algorithms for effective data movement between host and CXL memory regions.

Direct Attached using a transport other than CXL/PCIe Physical Layer

This use case extends the same logical architectural abstraction as the Direct-attached for transports other than CXL physical layer. Examples include UCIe, HBM3, Photonics for Memory Extension (placing memory in a suitable physical location by lengthening the distance between the Host/Accelerator and memory), Memory Expansion (providing larger memory footprint via different media types and benefiting from memory-tiering concepts), and Memory Pooling/Sharing.
The management of a CXL device will be achieved through the same in-band or out-of-band methods as described elsewhere in this paper. In this model, a non-CXL memory buffer will expose host-managed device memory (HDM). Host firmware maps HDM memory to the system's coherent address space, facilitating access from the host using standard write-back memory semantics. HDM memory can be configured by host firmware as either special-purpose memory or conventional memory.

Multi-Headed

The multi-headed CXL buffer provides physical connectivity to multiple hosts, thereby enabling the pooling and shared utilization of memory resources among these hosts. CXL provides logical abstractions to dynamically allocate memory to hosts based on specific workload requirements. In cases where the same memory region is made accessible to multiple hosts, ensuring cache coherency becomes a responsibility shared among the host, software, and/or the associated device. Using the same logical abstraction, this multi-headed use case may be extended to “Direct-attached using a different transport” case as well.

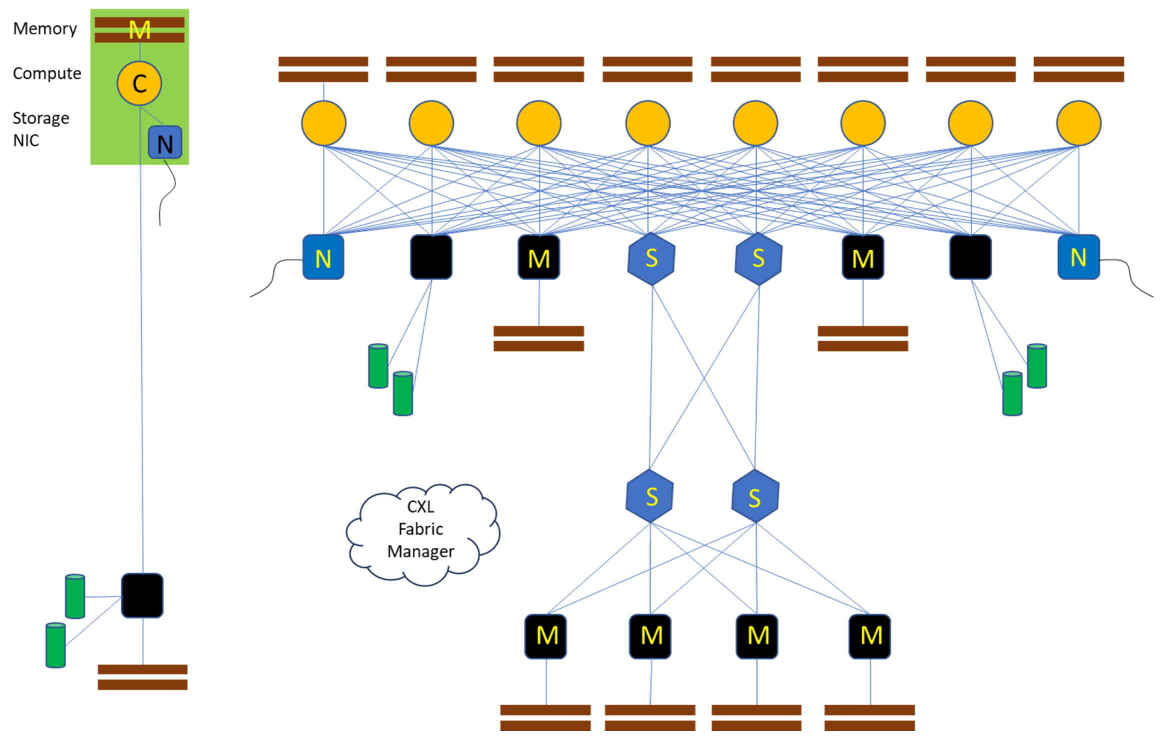


Figure 3: Example Multi-headed Memory Controller within a Switch-connected Fabric

Fabric

In contrast to the preceding deployment approaches, CXL memory buffers reside behind the CXL switch. The CXL switch connects to multiple hosts and offers the capability to dynamically provision memory to these hosts. Memory is allocated and made available to the hosts in a dynamic manner. The physical mapping and accesses to the actual CXL device are managed by the CXL switch(es). As shown in the picture below, the CXL fabric's complexity can range from a straightforward single switch with multi-host connectivity to complicated hierarchical fabric topologies. This deployment model facilitates pooling and sharing, akin to the multi-headed approach, but with enhanced fan out and improved host connectivity. The fabric manager can be integrated within the switch, implemented in the management controller or as a software agent running on any orchestrator host. In a multi-rack setup, the fabric manager enables dynamic memory provisioning based on workload demands by integrating with CXL fabric manager agents deployed throughout the data center.

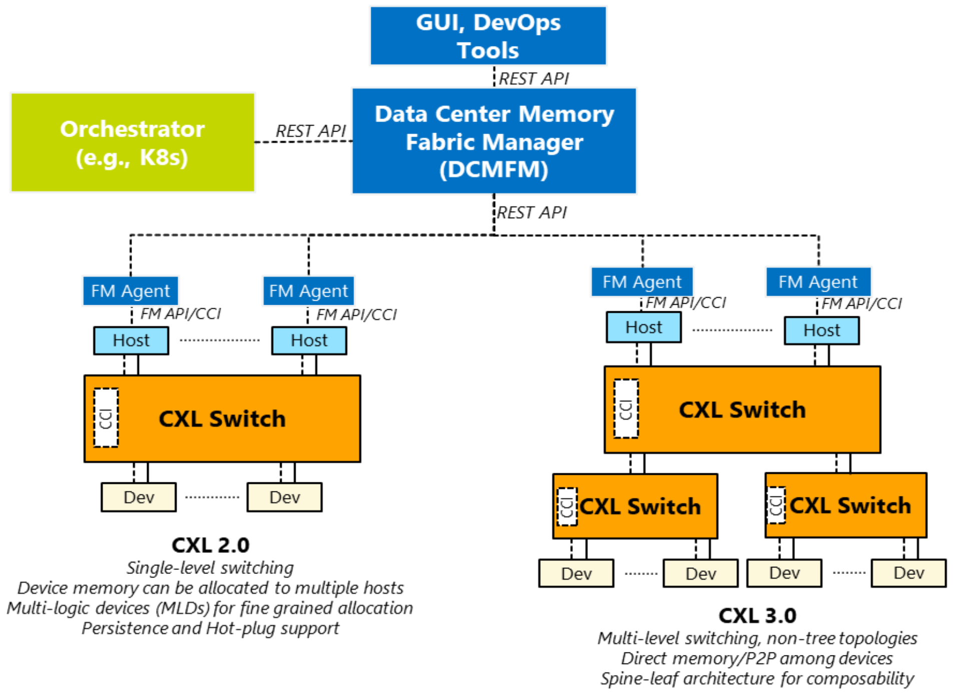

GUI, DevOps Tools
REST API
Orchestrator (e.g., K8s)
REST API
Data Center Memory Fabric Manager (DCMFM)
REST API
FM Agent
FM API/CCI
Host
CCI
CXL Switch
Dev
CXL 2.0
Single-level switching
Device memory can be allocated to multiple hosts
Multi-logic devices (MLDs) for fine grained allocation
Persistence and Hot-plug support
CXL 3.0
Multi-level switching, non-tree topologies
Direct memory/P2P among devices
Spine-leaf architecture for composability

Figure 4: Example CXL switch deployment models

Network

In this deployment configuration, CXL memory is accessed by the host via memory transport over the network. Unlike the prior deployment model, direct connectivity to the host is not established. While this model allows for greater fan-out and accessibility across multiple racks, it comes at the cost of increased latency. Software coherence models may work better in this network-connected configuration.

Data Center Memory Fabric Manager (DCMFM)

The Data Center Memory Fabric Manager (DCMFM) serves as a vital logical component for memory orchestration. Its primary role involves overseeing the life cycle of memory resources and optimizing the utilization of memory infrastructure within the data center. This software element handles various tasks, including discovery, monitoring, provisioning, and deprovisioning of memory resources within the data center environment. Illustrated in the CMS Logical Diagram earlier, the DCMFM operates as the control plane, responsible for managing critical memory infrastructure components like Memory buffers, CXL Switches/Fabric, and memory enclosures.

The DCMFM oversees and manages memory bandwidth, capacity, and dynamically adjusts the memory fabric configuration based on the workload requirements defined by the orchestrator. It has the capability to seamlessly integrate with advanced orchestration systems like Kubernetes, offering a centralized interface for administrators to establish and enforce policies, allocate resources, and govern the behavior of the memory fabric.

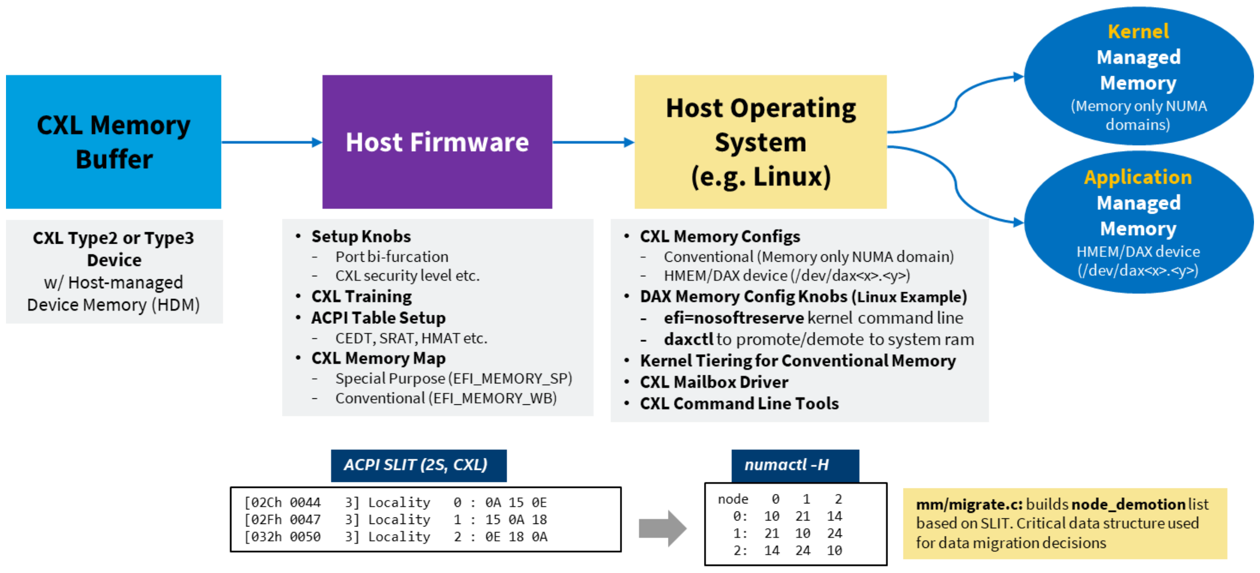
Additionally, the DCMFM holds a critical function in elevating security through the implementation of access controls and monitoring memory usage. It actively contributes to enhancing the comprehensive performance, scalability, and adaptability of the data center, in line with the evolving requirements of data center services.

The DCMFM communicates with physical memory devices through a DCMFM agent, which can be either software or firmware-based and supports both in-band and out-of-band modes. Interaction with the agent is facilitated through REST APIs (such as DMTF Redfish CXL models). The agent leverages the device's cxl.io and the Component Command Interface (CCI) to initiate control plane operations on the CXL device. Additionally, the DCMFM implements high availability measures to eliminate any single point of failure.

4. Memory Buffer Provisioning

This section describes the single host direct attached Single Logical Device (SLD) memory buffer provisioning to outline typical steps involved in consuming CXL memory buffer by the operating system or application. The actual implementation process may vary based on factors such as the CXL device type (memory buffer, type2 device, or switch), the capabilities of the device, support for the CXL protocol, and the capabilities of the host processor. CXL memory buffer needs to have at least one Host-managed Device Memory (HDM) so that device attached memory gets mapped to system coherent address space and accessible from host using write-back semantics. CXL memory buffer can take on any of the industry-supported form factors for memory modules and memory expansion boards.

Illustrated in Figure 5, host firmware takes on the role of configuring various settings such as PCIe bifurcation, interleaving, and DFx security related to PCIe and CXL. Additionally, it initiates link training and configures device registers following the guidelines specified in the CXL specification. Once the link training is successfully completed, the host firmware proceeds to configure ACPI tables, offering visibility into CXL devices and their configurations to the host operating system. Depending on the host firmware's configuration, the CXL memory map can be set up as either Special Purpose (EFI_MEMORY_SP) or Conventional (EFI_MEMORY_WB) memory. The operating system utilizes these memory map attributes to appropriately configure the CXL memory. Conventional memory is designated as a memory-only NUMA domain by the operating system, while special purpose memory is configured as a DAX (Direct Access) device. The ACPI SLIT table supplies NUMA distances, forming the foundation for the kernel to promote and demote CXL memory, alongside other ACPI tables like HMAT.


CXL Memory Buffer
Host Firmware
Host Operating System (e.g. Linux)
Kernel Managed Memory (Memory only NUMA domains)
Application Managed Memory HMEM/DAX device (/dev/dax<x>.<y>)
CXL Type2 or Type3 Device w/ Host-managed Device Memory (HDM)
• Setup Knobs
- Port bi-furcation
- CXL security level etc.
• CXL Training
• ACPI Table Setup
- CEDT, SRAT, HMAT etc.
• CXL Memory Map
- Special Purpose (EFI_MEMORY_SP)
- Conventional (EFI_MEMORY_WB)
• CXL Memory Configs
- Conventional (Memory only NUMA domain)
- HMEM/DAX device (/dev/dax<x>.<y>)
• DAX Memory Config Knobs (Linux Example)
- efi=nosoftreserve kernel command line
- daxctl to promote/demote to system ram
• Kernel Tiering for Conventional Memory
• CXL Mailbox Driver
• CXL Command Line Tools
ACPI SLIT (2S, CXL)
numactl -H
mm/migrate.c: builds node_demotion list based on SLIT. Critical data structure used for data migration decisions

```
[02Ch 0044   3] Locality    0 : 0A 15 0E
[02Fh 0047   3] Locality    1 : 15 0A 18
[032h 0050   3] Locality    2 : 0E 18 0A
```

```
node   0   1   2
  0:  10  21  14
  1:  21  10  24
  2:  14  24  10
```

Figure 5: CXL memory buffer provisioning

5. Memory Management

As depicted in the image below, the Kernel holds the responsibility of promoting and demoting data between conventional memory and CXL memory-only tiers based on the system admin's configured policies. Applications can utilize kernel-managed memory without requiring any modifications. Fine grained memory management can be achieved using kernel supported numa policies.

Special purpose memory is configured as a DAX device. Applications can memory-map the DAX device and directly consume the memory. Cachelib, an open-source software, is an example of an application that supports native memory tiering and effectively manages CXL memory. Virtual machines are an instance of applications capable of utilizing pass-through DAX devices to utilize memory within the guest operating system. To fulfill memory allocation requests within DAX memory regions, applications may need to implement a customized memory allocator. System admins retain the capability to promote and demote memory in and out of kernel-managed memory for DAX memory as well.

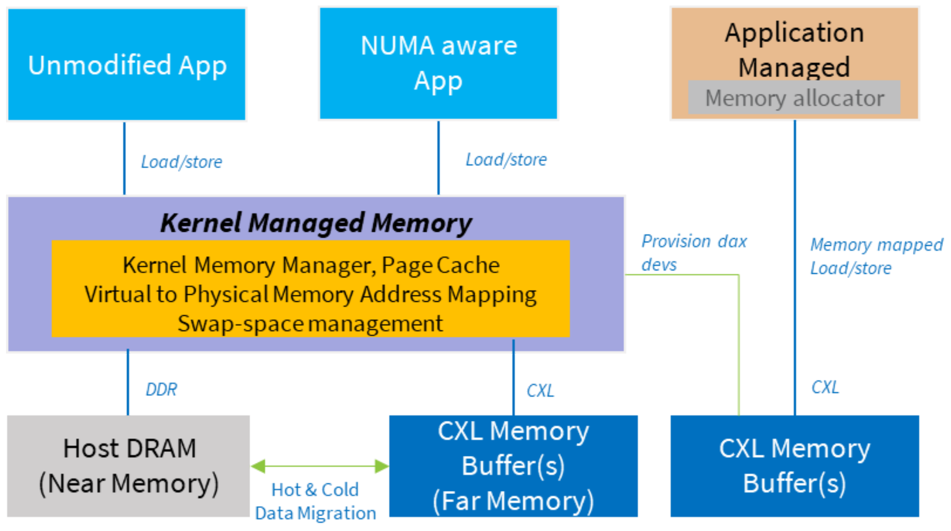


Figure 6: CXL memory management

6. Conclusion

The ever-expanding enterprise and hyperscale market have elevated the role of memory to a critical level. Data-intensive applications like Artificial Intelligence (AI), Machine Learning (ML), and databases are pushing the limits of server performance. To meet the demands of these applications, there has been a surge in newer memory technologies, interconnects, and hierarchies that promise enhanced memory capabilities such as increased bandwidth, capacity, throughput, data sharing, and scalability. Composable Memory Systems (CMS) is an emerging paradigm that facilitates dynamic and unified memory management across these diverse memory technologies, interconnects, and hierarchies. The CMS logical system architecture serves as a vital reference point for system designers, data center operators, and end customers as they navigate the landscape of commercial implementations. This entails selecting an ideal combination of host processor platforms, memory devices, chassis configurations, and rack setups.

7. Glossary

Term/Acronym	Definition
AI	Artificial Intelligence
CXL	Compute Express Link, a low-latency, high-bandwidth link that supports dynamic protocol muxing of coherency, memory access, and I/O protocols, thus enabling attachment of coherent accelerators or memory devices.
CCI	Component Command Interface
CXL.io	PCIe-based non-coherent I/O protocol with enhancements for accelerator support
CXL.mem	Memory access protocol that supports device-attached memory.
DMTF	Distributed Management Task Force
HBM	High-Bandwidth Memory
HDM	Host-managed Device Memory. Device-attached memory that is mapped to system coherent address space and accessible to the Host using standard write-back semantics.
HMAT	Heterogeneous Memory Attribute Table as defined in ACPI Specification
ML	Machine Learning
PCIe	PCI Express
SLD	Single Logical Device
SRAT	System Resource Affinity Table as defined in ACPI Specification

8. References

1. Compute Express Link (CXL) 1.1/2.0/3.0 specifications: https://www.computeexpresslink.org/
2. Compute Express Link (CXL™) Memory Module Base Standard JESD317 www.jedec.org
3. Redfish CXL Device Management Models Bundle https://www.dmtf.org/documents/redfish-spmf/redfish-cxl-device-management-models-bundle-08wip
4. Cachelib open-source software: https://cachelib.org/
5. QEMU CXL device emulation: https://www.qemu.org/docs/master/system/devices/cxl.html

9. License

10. About Open Compute Foundation

At the core of the Open Compute Project (OCP) is its Community of hyperscale data center operators, joined by telecom and colocation providers and enterprise IT users, working with vendors to develop open innovations that, when embedded in product are deployed from the cloud to the edge. The OCP Foundation is responsible for fostering and serving the OCP Community to meet the market and shape the future, taking hyperscale led innovations to everyone. Meeting the market is accomplished through open designs and best practices, and with data center facility and IT equipment embedding OCP Community-developed innovations for efficiency, at-scale operations and sustainability. Shaping the future includes investing in strategic initiatives that prepare the IT ecosystem for major changes, such as AI & ML, optics, advanced cooling techniques, and composable silicon. Learn more at www.opencompute.org.