Siamak Tavallaei
CXL Advisor to the Board of Directors, CXL™ Consortium
May 23, 2023

Compute Express Link™ (CXL™)
A Coherent Interface for Ultra-High-Speed Transfers

Alibaba Group
AMD
arm
cisco
DELL EMC
Google
Hewlett Packard Enterprise
HUAWEI
IBM
intel
Meta
Microsoft
NVIDIA
Rambus
SAMSUNG

CXL Board of Directors

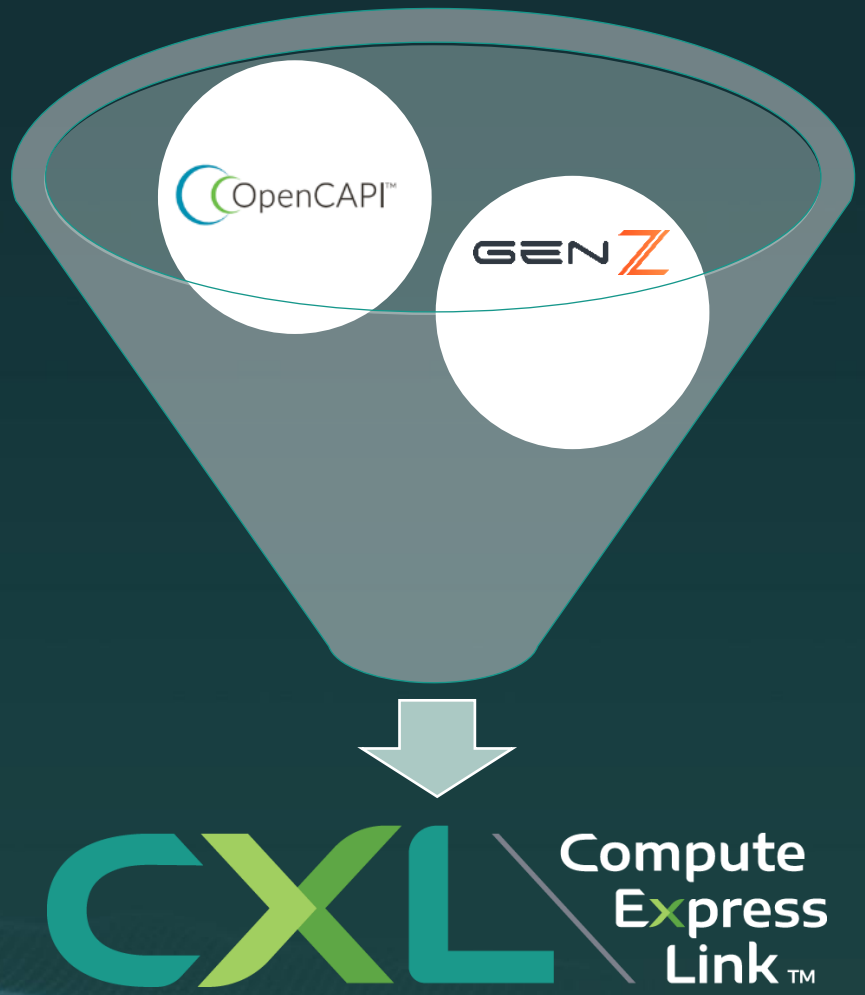
CXL Compute Express Link™

Industry Open Standard for High Speed Communications

240+ Member Companies

CXL Compute Express Link™

Industry focal point



CXL is emerging as the industry focal point for coherent data traffic

- CXL Consortium and OpenCAPI signed agreement and transferred OMI specification and OpenCAPI assets to the CXL Consortium

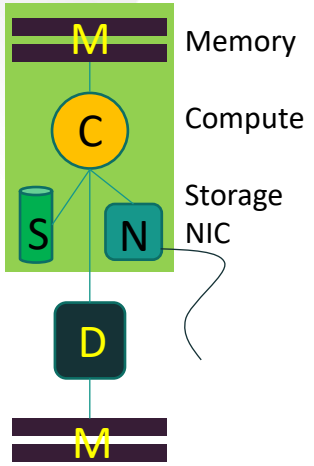
CXL Compute Express Link™ OpenCAPI™

August 1, 2022, Flash Memory Summit

CXL Consortium and OpenCAPI Consortium
Sign Letter of Intent
to Transfer OpenCAPI Assets to CXL

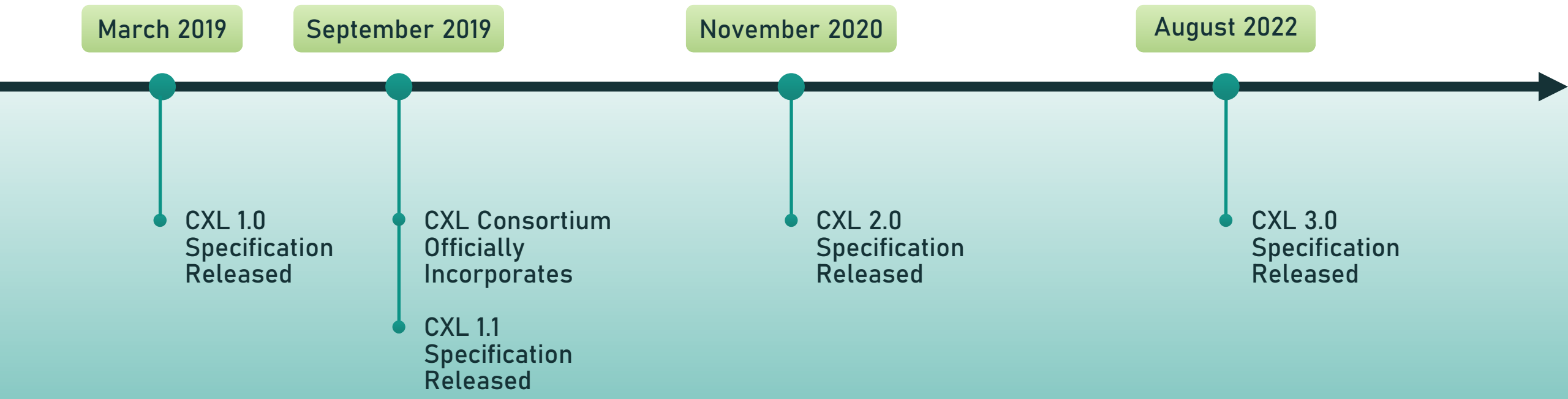
- In February 2022, CXL Consortium and Gen-Z Consortium signed agreement to transfer Gen-Z specification and assets to CXL Consortium

CXL Overview



- New breakthrough high-speed fabric
  - Enables a high-speed, efficient interconnect
    - between CPU, memory and accelerators
  - Builds upon PCI Express® (PCIe®) infrastructure
    - leveraging the PCIe® physical and electrical interface
  - Maintains memory coherency
    - between the CPU-attached memory space and memory on CXL-attached devices
    - enables fine-grained resource sharing for high performance in heterogeneous compute environments
    - enables memory disaggregation, memory pooling and sharing
    - enables persistent memory and emerging memory media
- Delivered as an open industry standard
  - CXL 3.0 specification is fully backward compatible with CXL 1.1 and CXL 2.0
  - Future CXL Specification generations will include continuous innovation to meet industry needs and support new technologies

CXL Specification Release Timeline



CXL Technology Update

CXL™ Approach

Coherent Interface

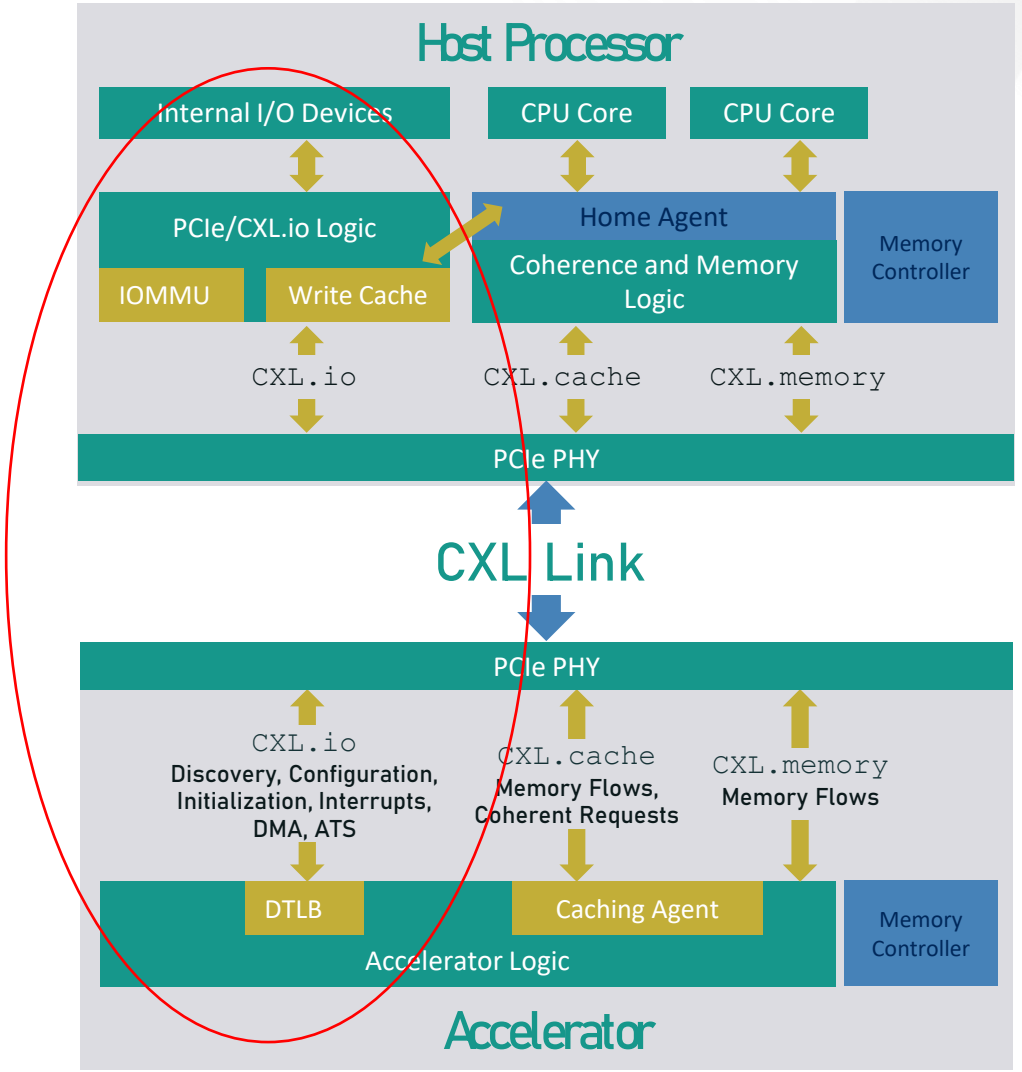
- Leverages PCIe with three multiplexed protocols
- Built on top of PCIe® infrastructure

Low Latency

- CXL.Cache/CXL.Memory targets near CPU cache coherent latency (<200ns load to use)

Asymmetric Complexity

- Eases burdens of cache coherence interface designs for devices



CXL™ Approach

Coherent Interface

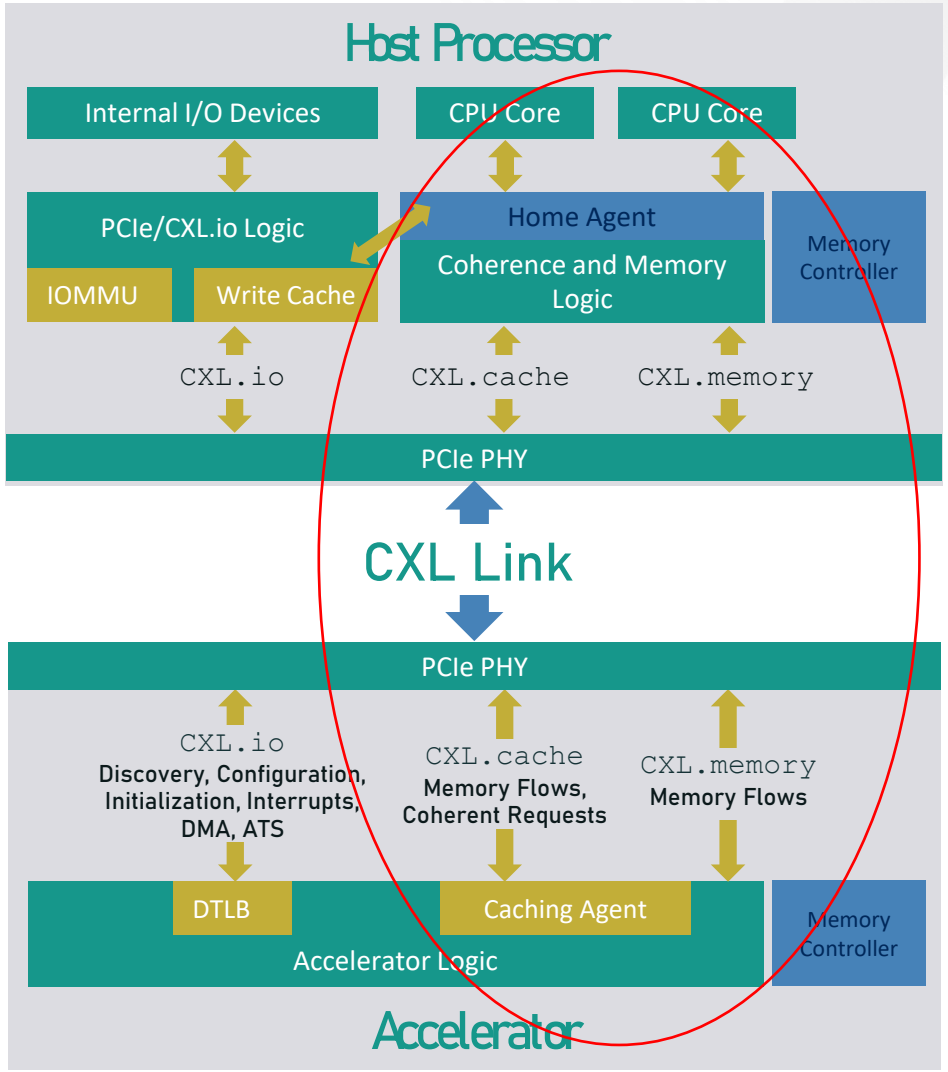
- Leverages PCIe with three multiplexed protocols
- Built on top of PCIe® infrastructure

Low Latency

- `CXL.Cache/CXL.Memory` **targets near CPU cache coherent latency (<200ns load to use)**

Asymmetric Complexity

- Eases burdens of cache coherence interface designs for devices



CXL™ Approach

Coherent Interface

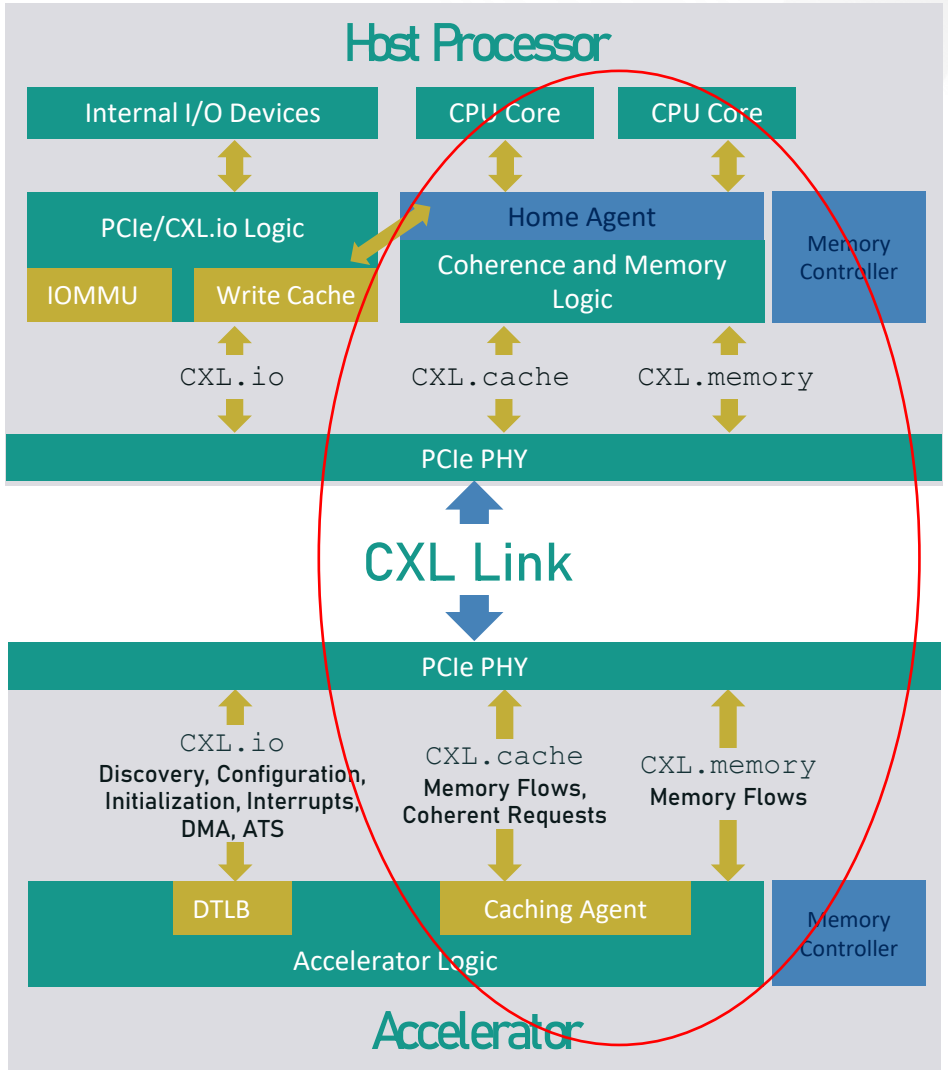
- Leverages PCIe with three multiplexed protocols
- Built on top of PCIe® infrastructure

Low Latency

- `CXL.Cache/CXL.Memory` targets near CPU cache coherent latency (<200ns load to use)

Asymmetric Complexity

- Eases burdens of cache coherence interface designs for devices



CXL™ Approach

Coherent Interface

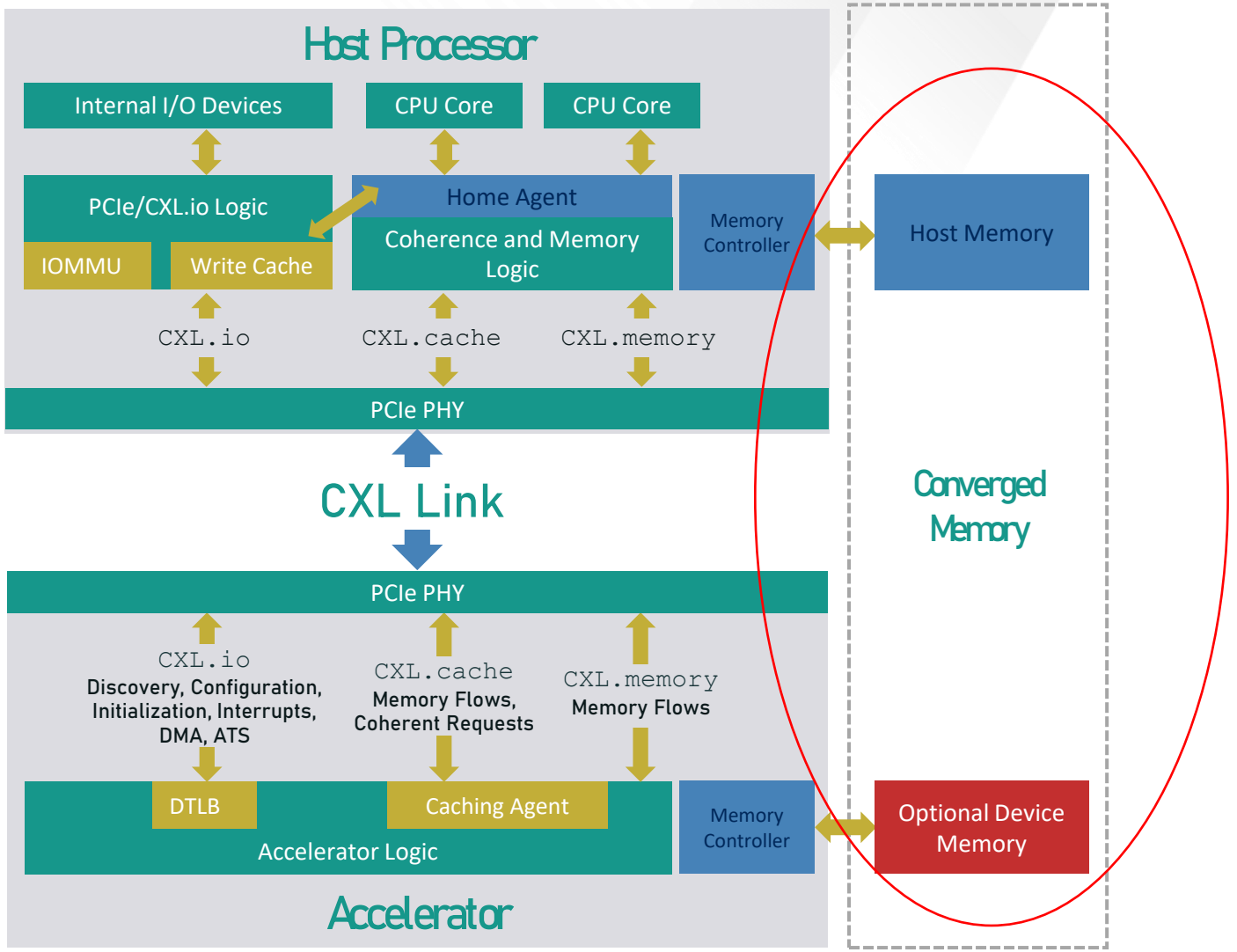
- Leverages PCIe with three multiplexed protocols
- Built on top of PCIe® infrastructure

Low Latency

- CXL.Cache/CXL.Memory targets near CPU cache coherent latency (<200ns load to use)

Asymmetric Complexity

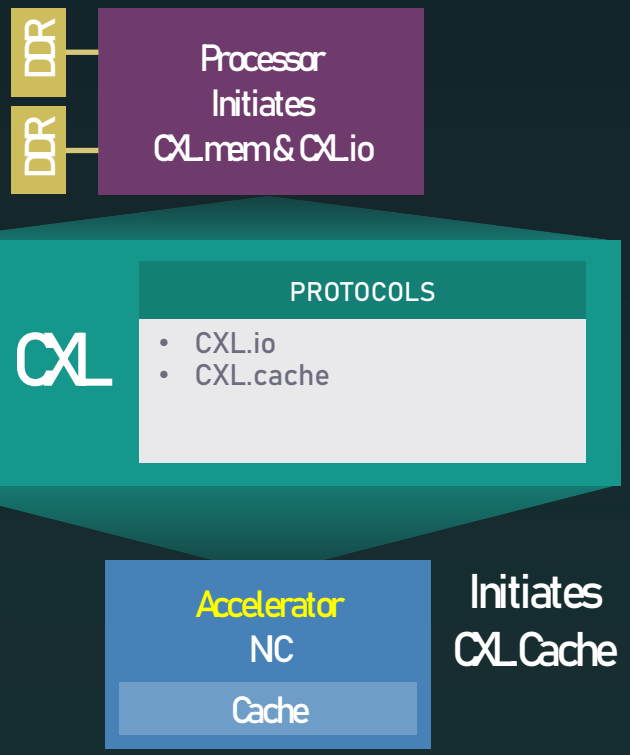
- Eases burdens of cache coherence interface designs for devices



Recap: CXL 1.0/1.1 Representative Use Cases

Caching Devices / Accelerators

TYPE 1

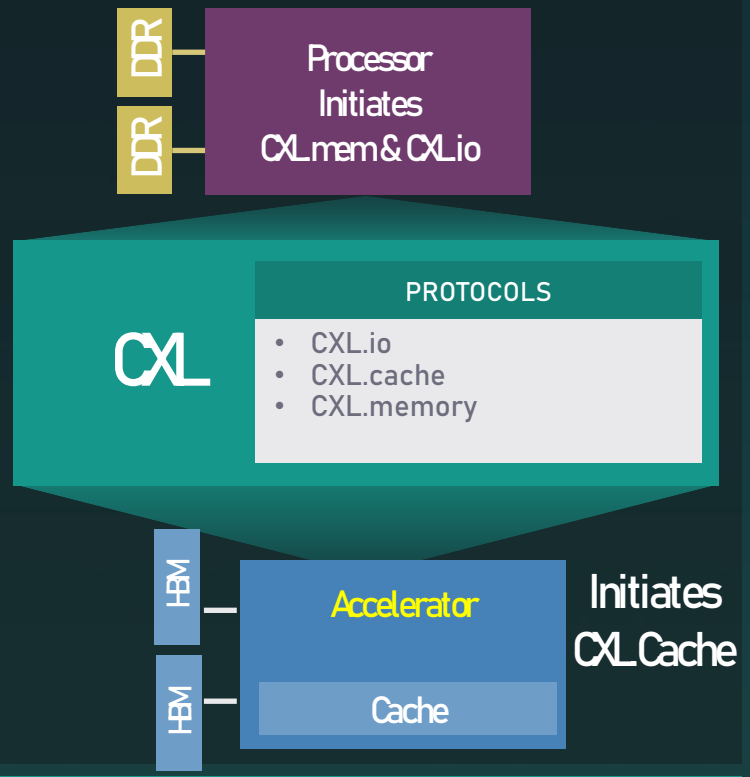


USAGES

- PGAS NIC
- NIC atomics

Accelerators with Memory

TYPE 2

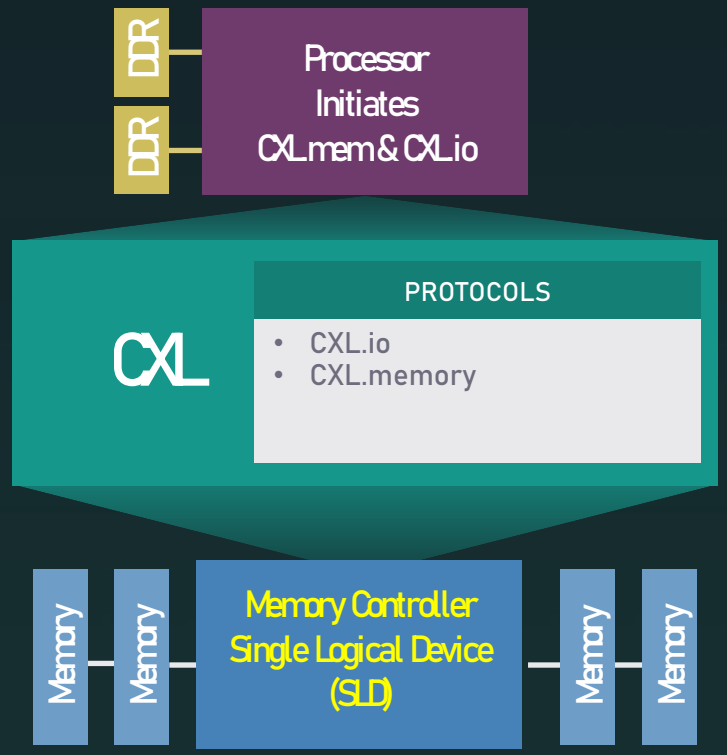


USAGES

- HDM-D: Device-Coherent HDM region type
- GP GPU
- Dense computation

Memory Buffers

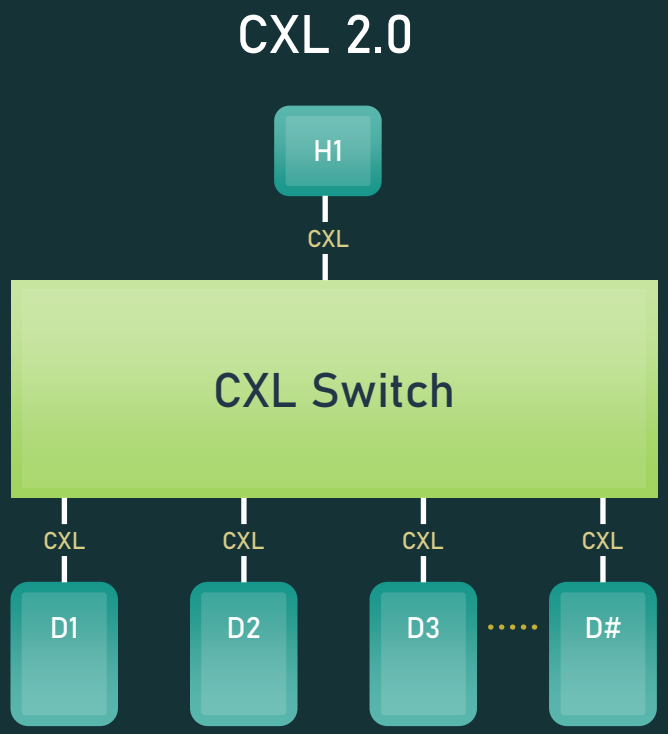
TYPE 3



USAGES

- HDM-H: Host-only Coherent HDM region type
- Memory BW expansion
- Memory capacity expansion
- Storage-class memory

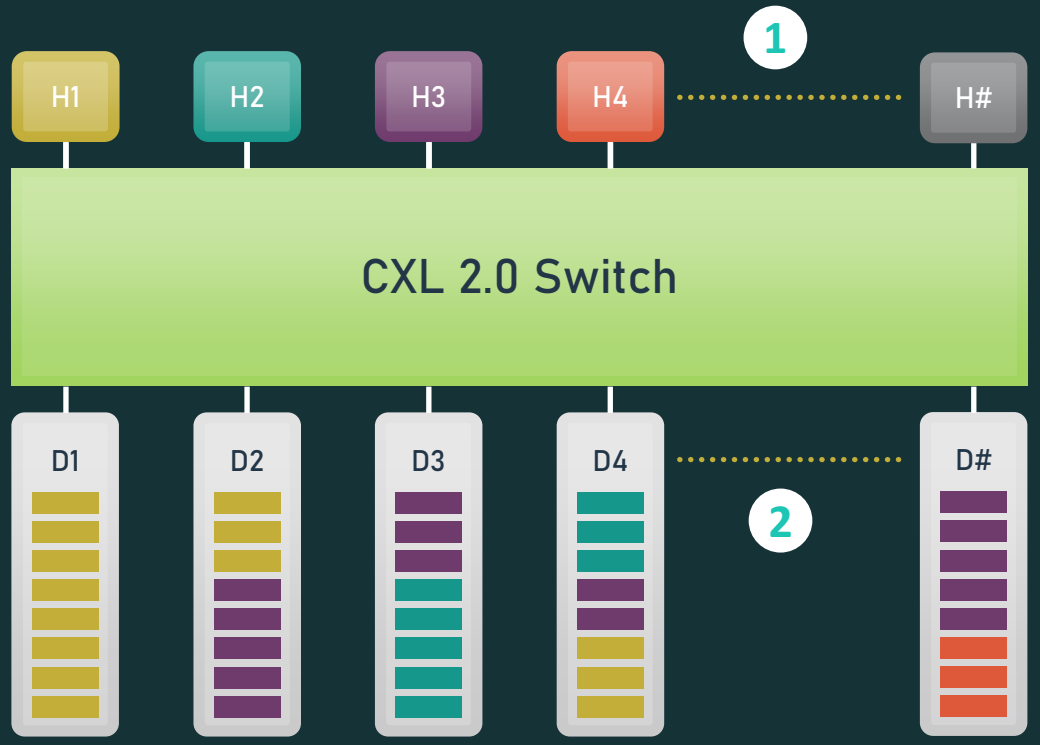
CXL 2.0 Switch Capability



- Supports single-level switching
- Enables memory expansion and resource allocation

Recap: CXL 2.0 FEATURE SUMMARY

CXL 2.0 Feature Summary
Memory Pooling w/ MLDs
Global Persistent Flush (storage)
CXL IDE (security)
Switching (single-level)



SLD MLD: MEMORY POOLING

1. Device memory can be allocated across multiple hosts
2. Multi Logical Devices (MLD) allow finer grain memory allocation
3. Persistence Flows
4. Pooling of accelerators Hot-plug flows

Industry trends

- Use cases driving need for higher bandwidth: e.g., high performance accelerators, system memory, SmartNIC etc.
- CPU capability requiring more memory capacity and bandwidth per core
- Efficient peer-to-peer resource sharing / messaging across multiple domains
- Need to overcome memory bottlenecks due to CPU pin and thermal constraints

CXL 3.0 Specification

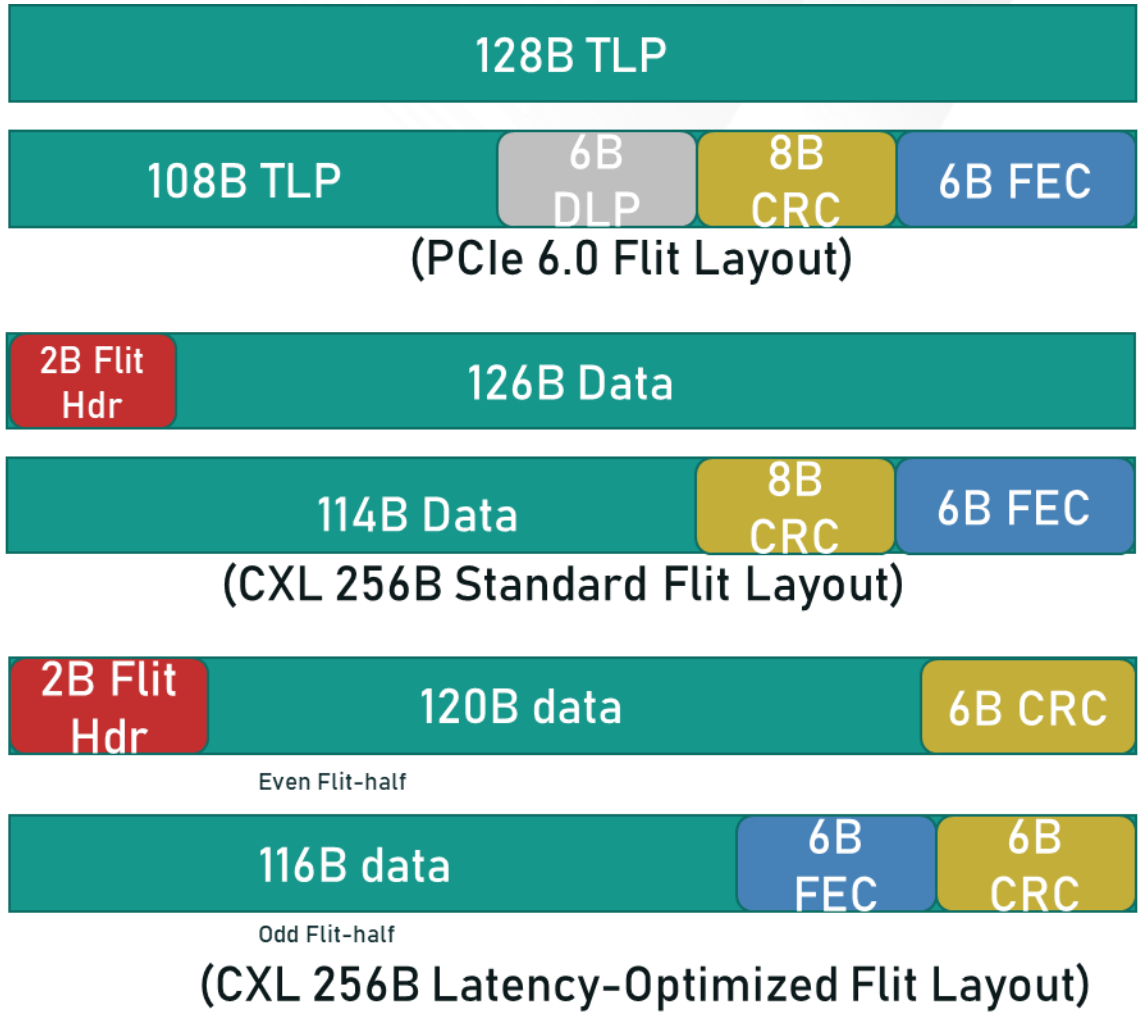
Expanded capabilities for increased scale and optimized resource utilization

- Double the bandwidth PCIe® 6.0 (Zero added latency over CXL 2.0)
- Enhanced Switching
  - Direct memory & Peer-to-Peer access by devices
  - Multiple Type 1 & Type 2 devices per root port
  - Multi-level switching
- Fabric capabilities
  - Fabric-attached memory (GFAM)
  - Enhanced fabric management framework
  - Support for composable disaggregated infrastructure
- Multi-headed devices
- New symmetric memory capabilities
- Enhanced memory pooling and sharing
- Near-memory processing
- Improved software capabilities
- Fully backward-compatible with:
  - CXL 2.0
  - CXL 1.1
  - CXL 1.0

CXL 3.0 is a huge step function with fabric capabilities while maintaining full backward compatibility with prior generations

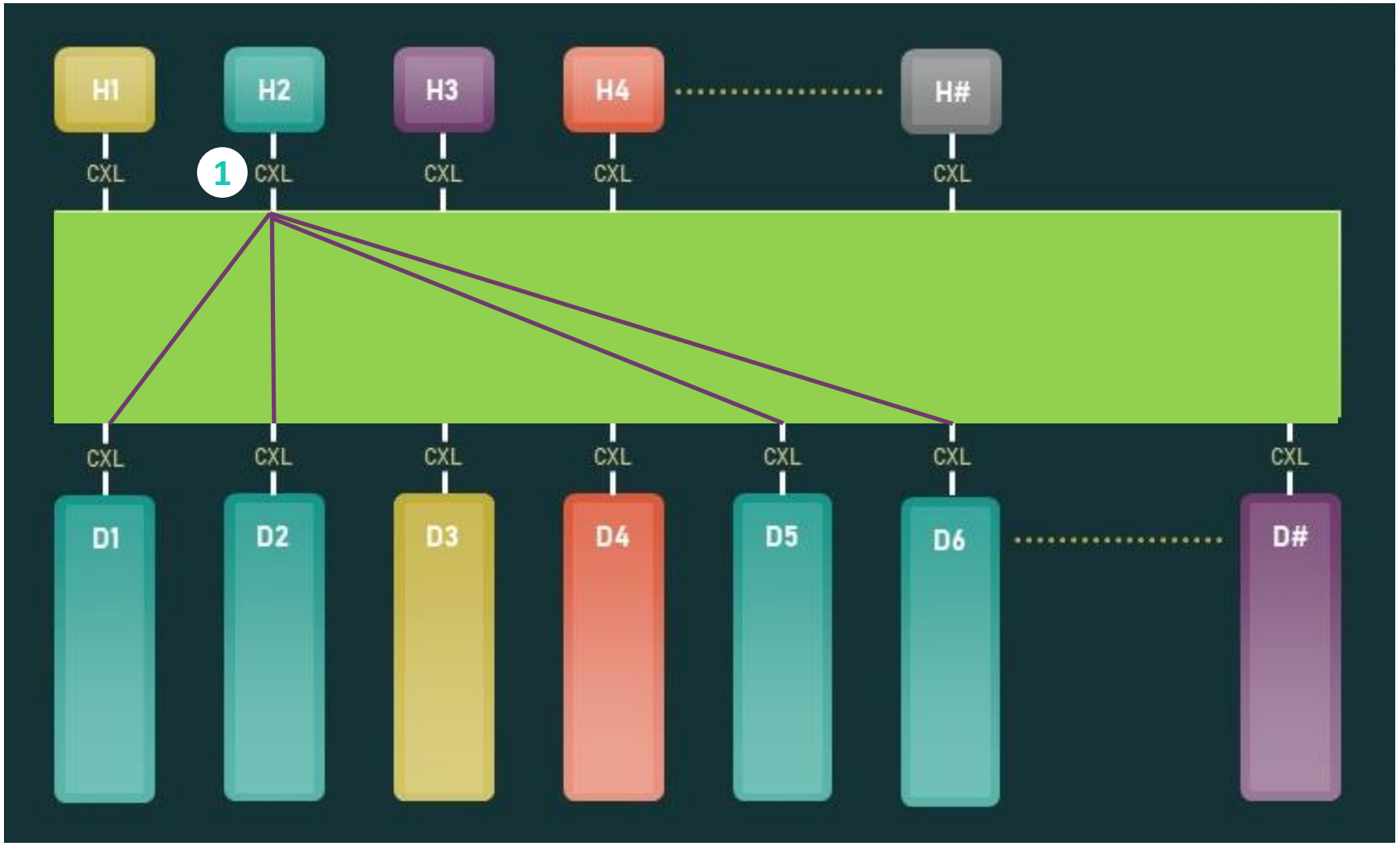
CXL 3.0: Doubles Bandwidth with Same Latency

- Uses PCIe 6.0® PHY @ 64 GT/s
- PAM-4 and high BER mitigated by PCIe 6.0 FEC and CRC (different CRC for latency optimized)
- Enables several new CXL 3 protocol enhancements with the 256B Flit format
- Standard 256B Flit along with an additional 256B Latency Optimized Flit (0-latency adder over CXL 2)
  - 0-latency adder trades off FIT (failure in time, 109 hours) from 5x10-8 to 0.026 and Link efficiency impact from 0.94 to 0.92 for 2-5ns latency savings (x16 – x4)1
- Extends to lower data rates (8, 16, and 32 GT/s)



1: D. Das Sharma, "A Low-Latency and Low-Power Approach for Coherency and Memory Protocols on PCI Express 6.0 PHY at 64.0 GT/s with PAM-4 Signaling", IEEE Micro, Mar/ Apr 2022 (https://ieeexplore.ieee.org/document/9662217)

Device to Device Comms



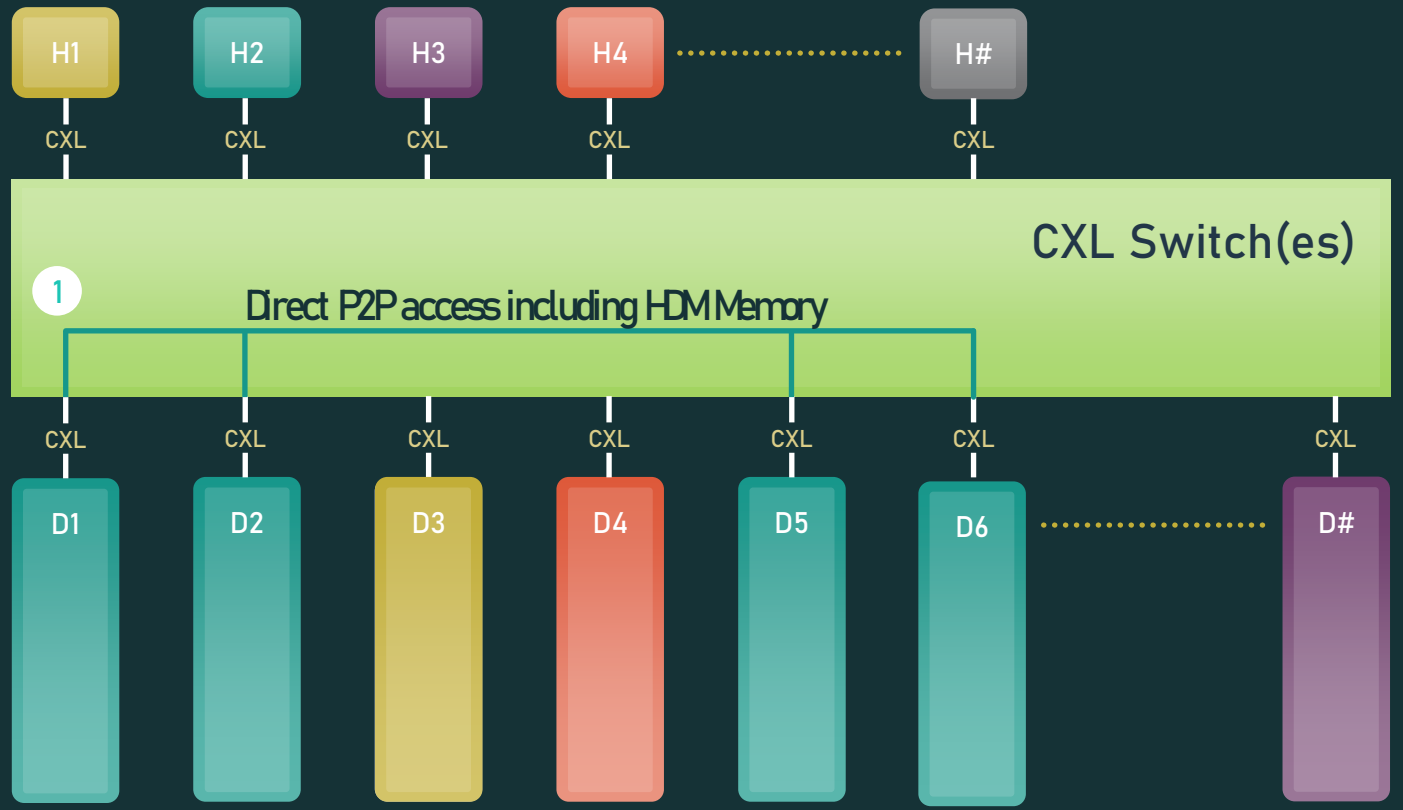
The Traditional
PCIe Tree Structure
defines a virtual hierarchy

for
Host #2 to reach
D1, D2, D5, and D6

and for
D1, D2, D5, and D6 to reach
each other via Host #2

1 Host #2's Link
is a bottleneck

CXL 3.0 Protocol Enhancements (UIO and BI) for Device to Memory Connectivity



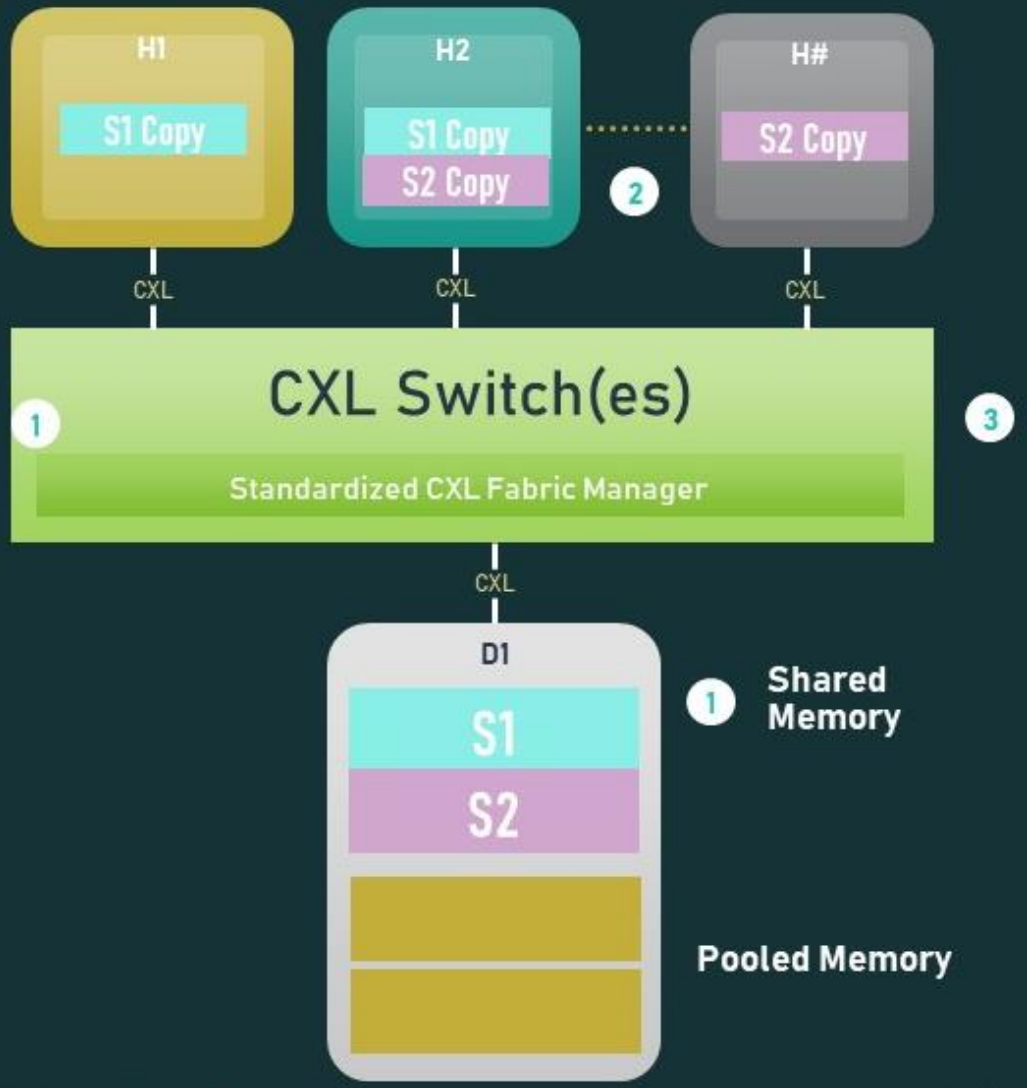
CXL 3.0 enables non-tree topologies and peer-to-peer communication (P2P) within a virtual hierarchy of devices

- Virtual hierarchies are associations of devices that maintains a coherency domain

- P2P to HDM-DB memory is I/O Coherent via a new Unordered I/O (UIO) Flow in CXL.io
- Type 2 & 3 devices that host HDM-DB memory monitor memory accesses. If coherency conflict, they generate a new Back-Invalidation flow (BI) (CXL.mem) to the host to ensure coherency

HDM-DB: Device-Coherent using Back-Invalidation HDM region type for device-attached Memory (DAMem). Can be used by Type 2 or Type 3 devices

CXL 3.0: Memory Sharing



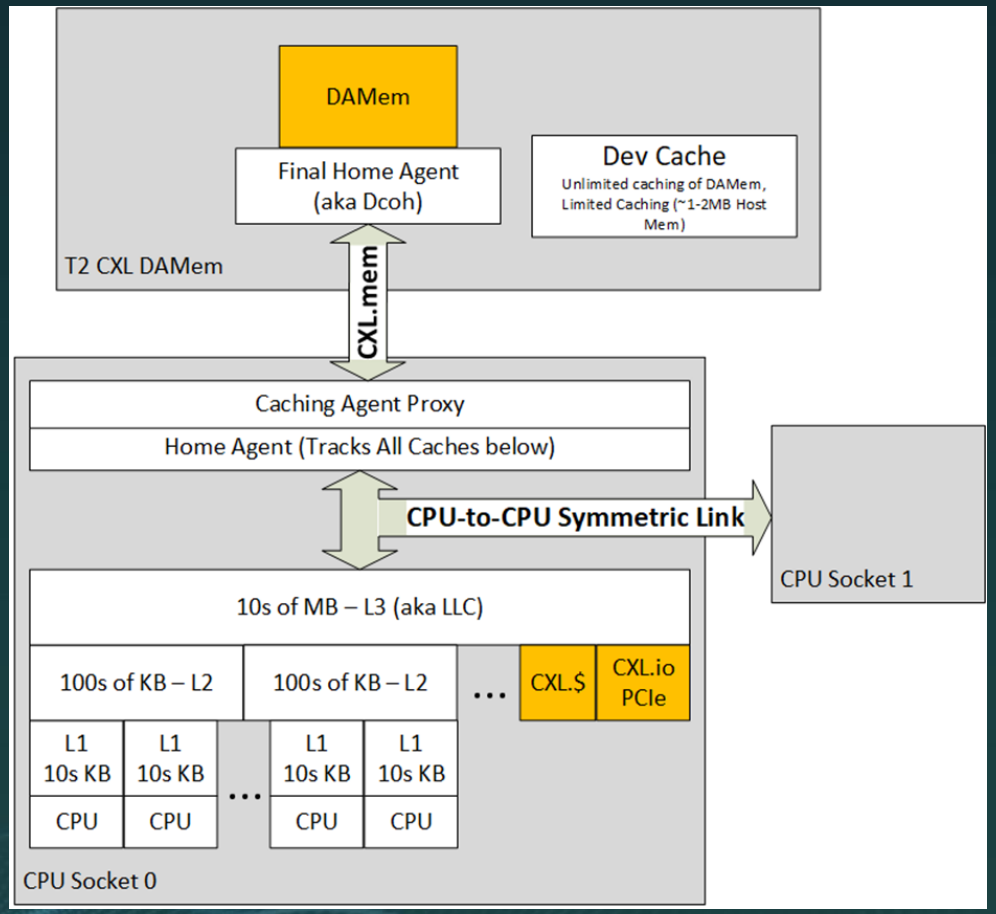
1. Device memory can be shared by all hosts to increase data-flow efficiency and improve memory utilization

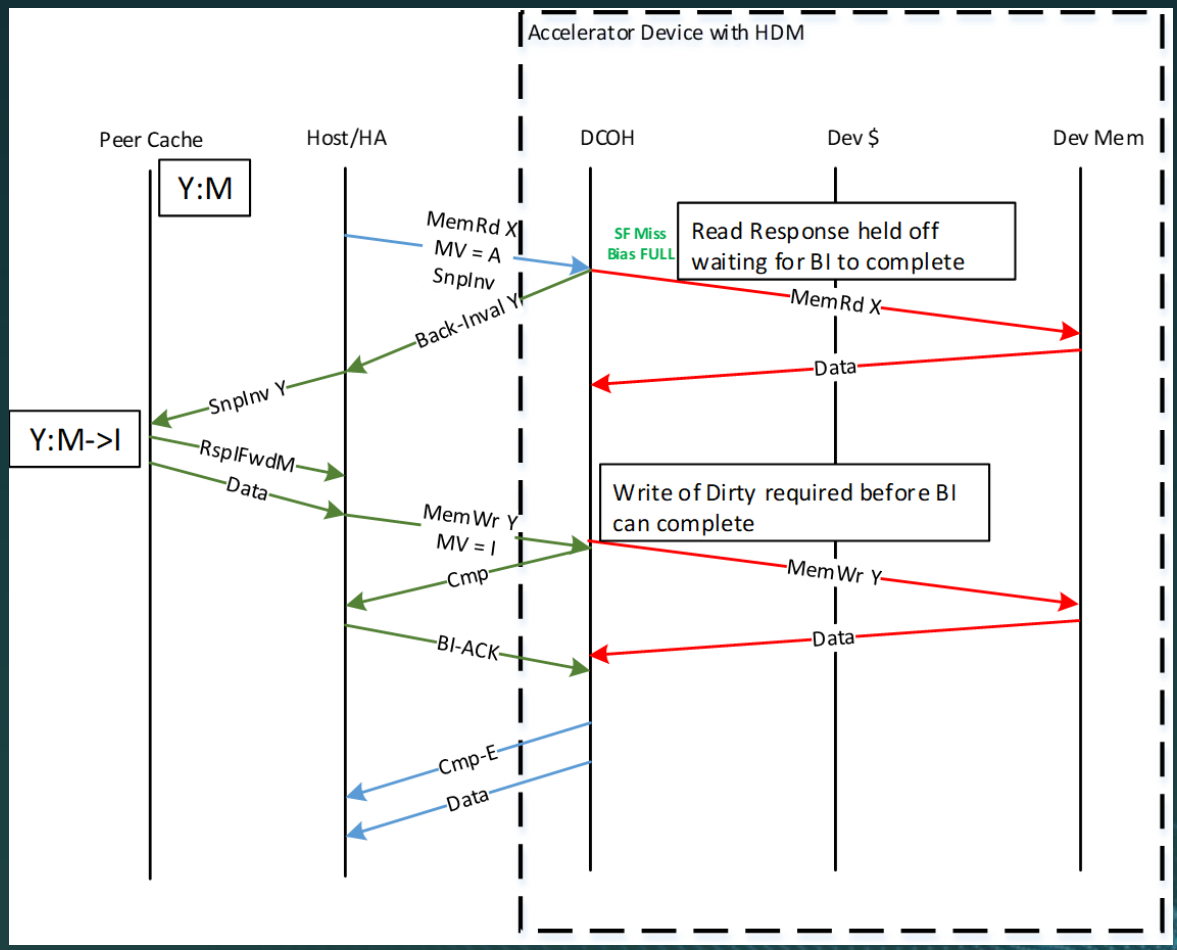
2. Host can have a coherent copy of the shared region or portions of shared region in host cache

3. CXL 3.0 defined mechanisms to enforce hardware cache-coherency

CXL 3.0 Protocol Enhancements: Mapping Large memory in Type-2 Devices to HDM with Back Invalidate

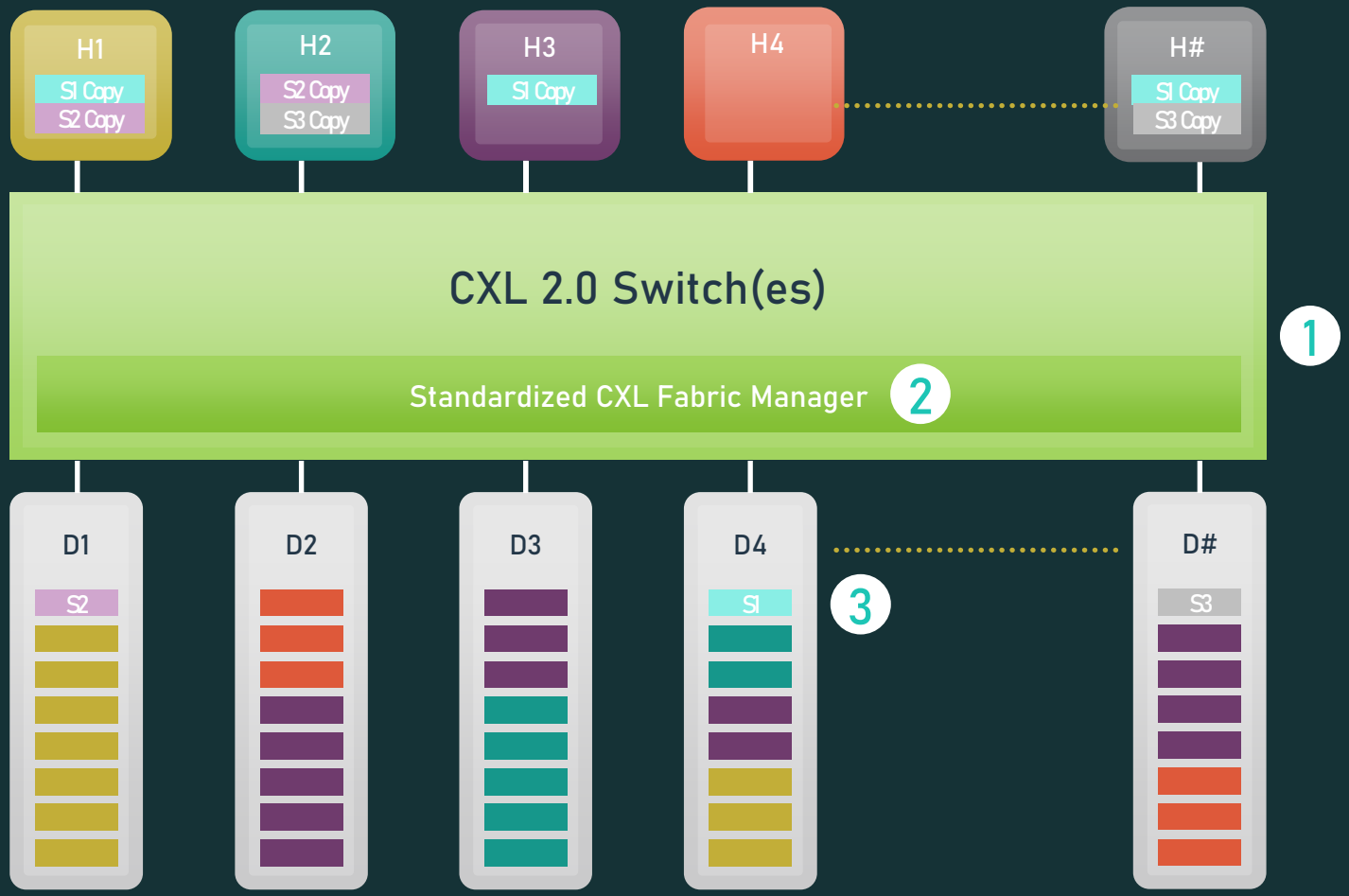
Existing Bias – Flip mechanism needed HDM to be tracked fully since device could not back snoop the host. Back Invalidate with CXL 3.0 enables snoop filter implementation resulting in large memory that can be mapped to HDM





CXL 3.0: Memory Pooling & Sharing

A Fabric for Disaggregated and Compostable Architectures



1. Expanded use case showing memory sharing and pooling

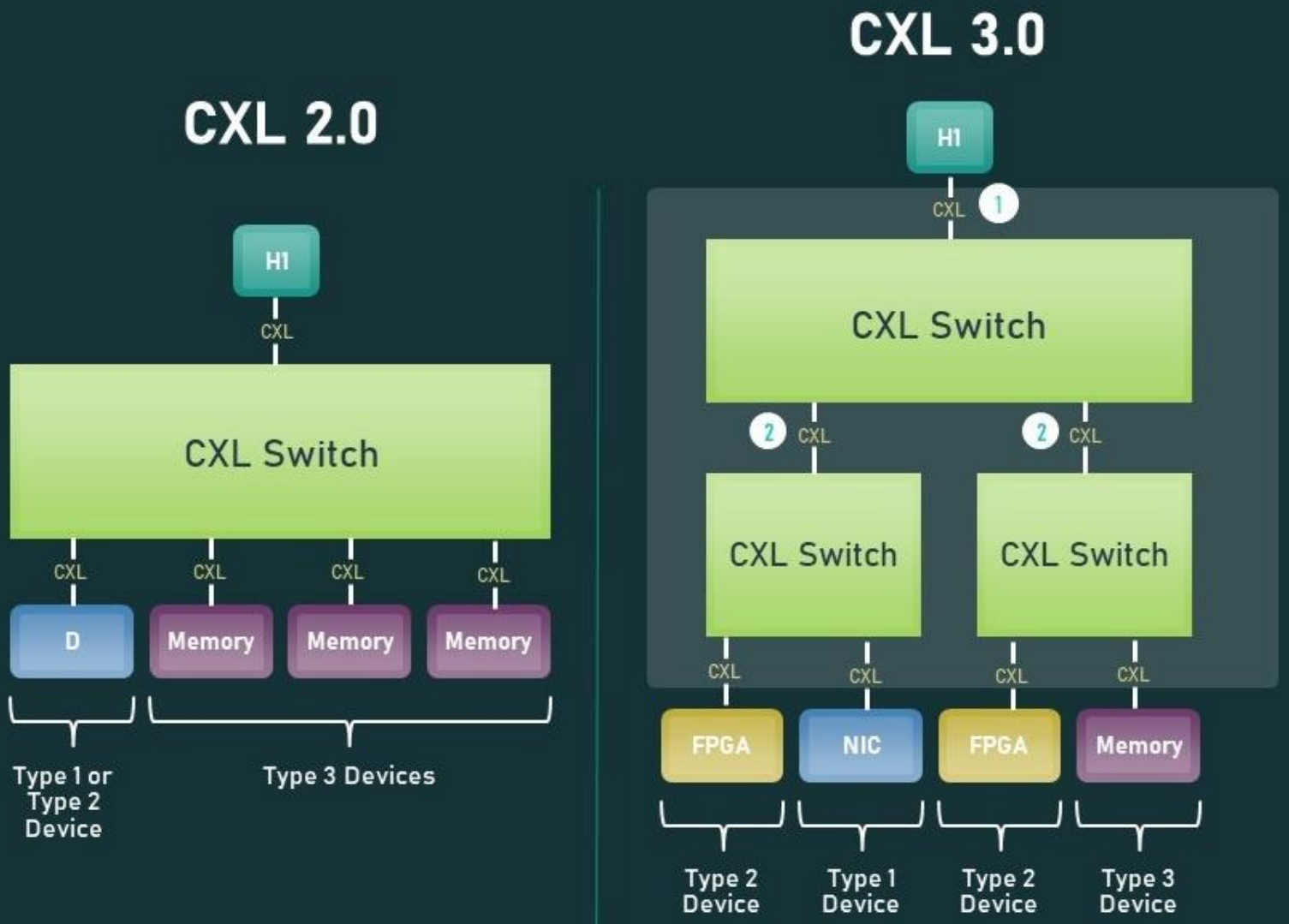
2. CXL Fabric Manager is available to setup, deploy, and modify the environment

3. Shared Coherent Memory across hosts using hardware coherency (directory + Back-Invalidate Flows) (BI).

   Allows one to build large clusters to solve large problems through shared memory constructs.

   Defines a Global Fabric-Attached Memory (GFAM) which can provide access to up to 4095 entities

CXL 3.0: Multiple Level Switching, Multiple Type-1/2/3

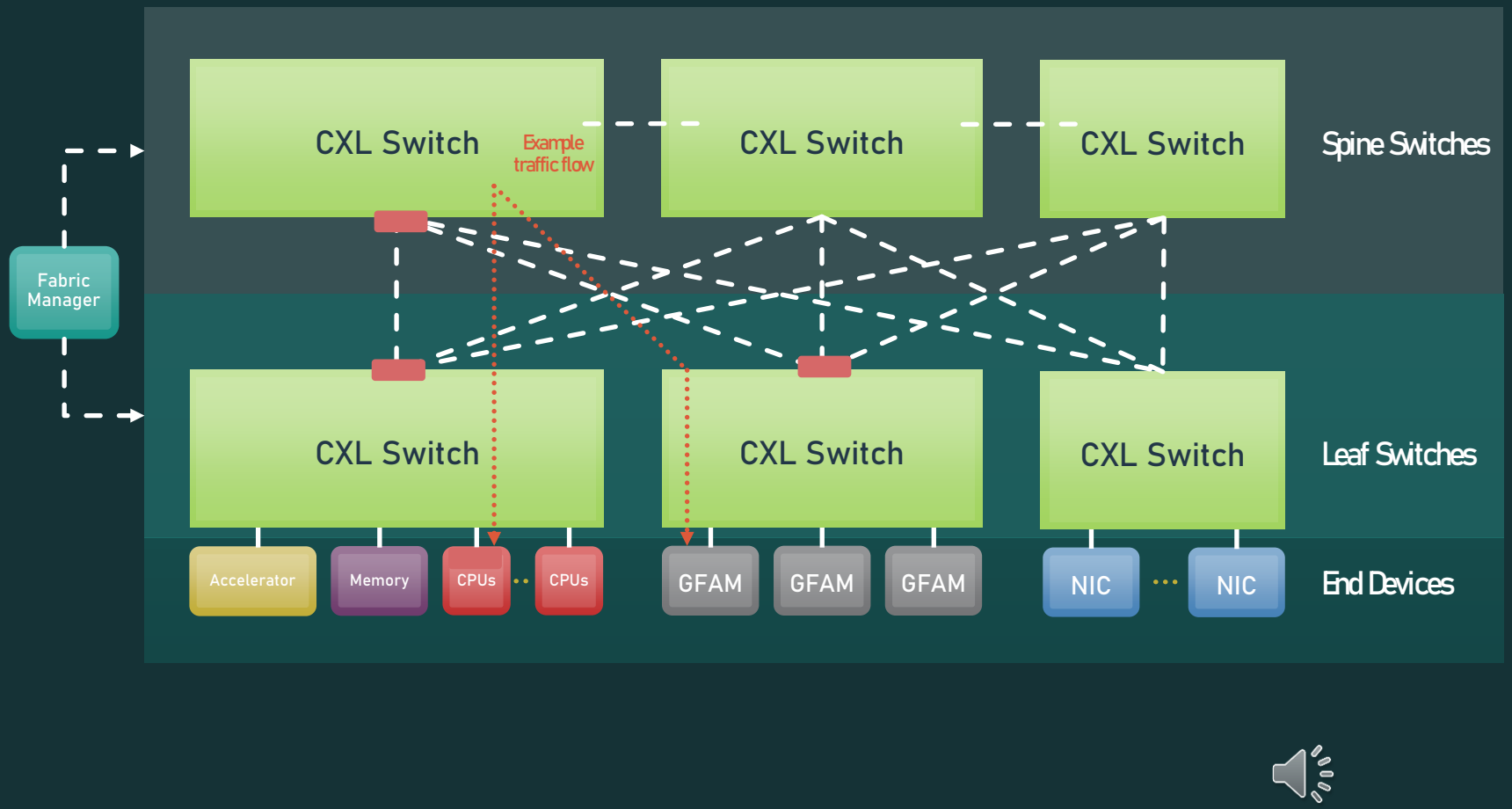


1. Each host's root port can connect to more than one device type
   (up to 16 CXL.cache devices)

2. Multiple switch levels (aka cascade)
   - Supports fanout of all device types

CXL 3.0 Fabrics

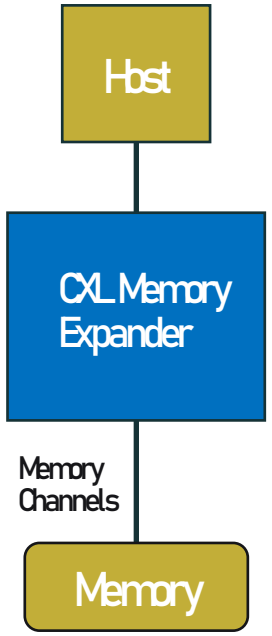
Composable Systems with Spine/Leaf Architecture



CXL 3.0 Fabric Architecture

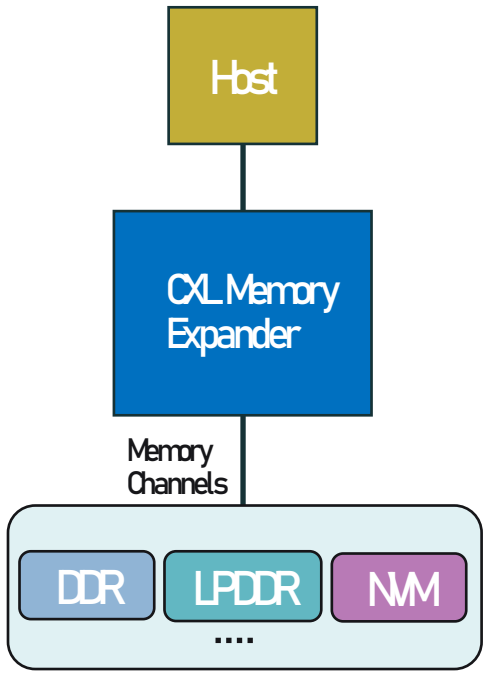
- Interconnected Spine Switch System
- Leaf Switch NIC Enclosure
- Leaf Switch CPU Enclosure
- Leaf Switch Accelerator Enclosure
- Leaf Switch Memory Enclosure

Current Use Cases



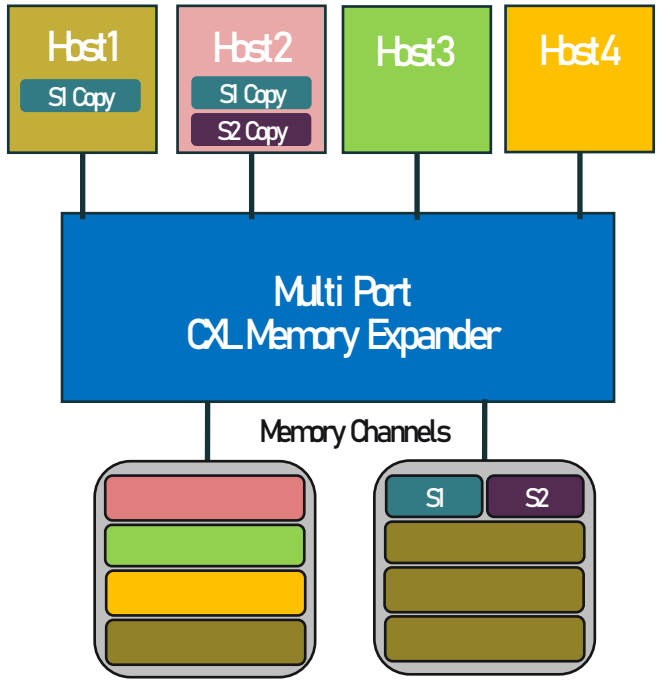
CXL Memory Expansion

- Improves processor efficiency
- Increases Capacity
- Improves Bandwidth
- Lowers TCO



CXL-Enabled Tiered Expansion

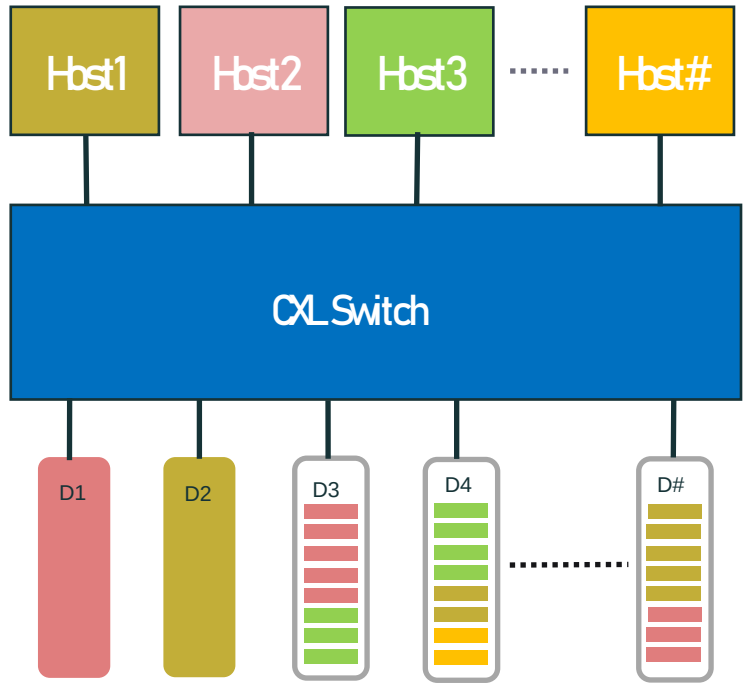
- Flexible support for faster or slower Memory Tier
- Lowers TCO- higher memory capacity for lower $/GB
- Improves Bandwidth



Pooled Memory

Shared Memory

- Reduces Memory Stranding
- Improves Data Flow Efficiency
- Improves Memory Utilization
- Lowers TCO



CXL Switch with SLD and MLD

- Scales to large datasets
- Flexibility across the fabric
- Improves Data Flow Efficiency
- Improves Memory Utilization
- Lowers TCO

CXL 3.0 Spec Feature Summary

Features	CXL 1.0 / 1.1	CXL 2.0	CXL 3.0
Release date	2019	2020	Aug 2022
Max link rate	32GT/s	32GT/s	64GT/s
68-byte Flit (up to 32 GT/s)	✓	✓	✓
Type 1, Type 2, and Type 3 Devices	✓	✓	✓
Memory Pooling w/ MLDs		✓	✓
Global Persistent Flush		✓	✓
CXL IDE		✓	✓
Switching (Single-level)		✓	✓
Switching (Multi-level)			✓
Multiple Type 1 & Type 2 Devices per root port			✓
Direct memory access for peer-to-peer			✓
256-byte Flit (up to 64 GT/s)			✓
256-byte Flit (Enhanced coherency)			✓
256-byte Flit (Memory sharing)			✓
256-byte Flit (Fabric capabilities)			✓

Not supported

✓ Supported

CXL Ecosystem & Demos and Current Use Cases

CXL Ecosystem

Growth of CXL Ecosystem since it's inception

- Ecosystem that meets the ever-increasing performance and scale requirements
- Fully backwards compatible
- Lowers overall system cost
- Comprehensive compliance and testing support

CXL Consortium members for showcasing demos at SC'22!

Conclusions

CXL 3.0 features

- Full fabric capabilities and fabric management
- Expanded switching topologies
- Enhanced coherency capabilities
- Peer-to-peer resource sharing
- Double the BW and zero added latency compared to CXL 2.0
- Full backward compatibility with CXL 2.0, CXL 1.1, & CXL 1.0

Enabling new usage models

- Memory sharing between hosts and peer devices
- Support for multi-headed devices
- Expanded support for Type-1 and Type-2 devices
- Global Fabric-attached Memory (GFAM) provides expansion capabilities for current and future memory

Call to Action

- Download the CXL 3.0 specification
- Support future specification development by joining the CXL Consortium as a Contributor
- Take a system-level view (Hardware-Software co-Design)
- Follow us on Twitter and LinkedIn for updates!

CXL Resources

Webinars:

- Webinar: A look into the CXL device ecosystem and the evolution of CXL use cases
- Webinar: Introducing CXL 3.0: Enabling composable systems with expanded fabric capabilities
- Webinar: CXL 1.1 vs. CXL 2.0 – What's the difference?
- Webinar Archive: https://www.computeexpresslink.org/webinars

Blogs:

- Upcoming Webinar: A Look into the CXL™ Device Ecosystem and the Evolution of CXL Use Cases
- CXL 3.0 Webinar Q&A Recap
- CXL™ Consortium Member Spotlight: UnifabriX
- Blog Archive: https://www.computeexpresslink.org/blog

White Papers:

- CXL 3.0 specification
- An Overview of Reliability, Availability, and Serviceability (RAS) in Compute Express Link™ 2.0

Thank You

www.ComputeExpressLink.org

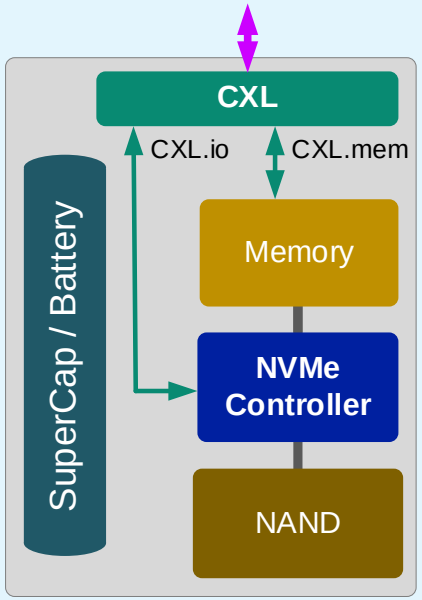
Supplemental Use Cases targeting Storage

(NVMe, Page Prediction, Pooling, Expanded Memory)

Courtesy: Peter Onufryk peter.onufryk@intel.com

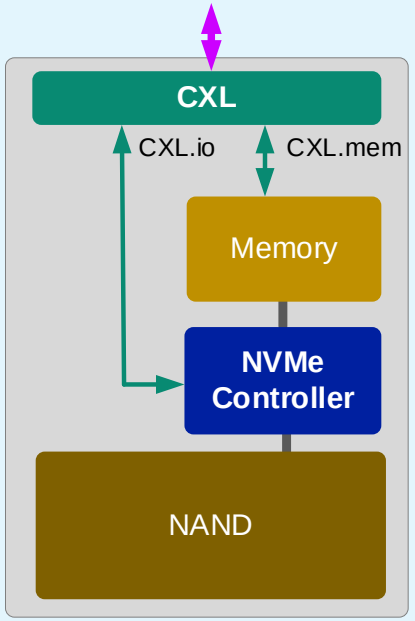
Possible Combined CXL & NVMe Applications

Persistent Memory



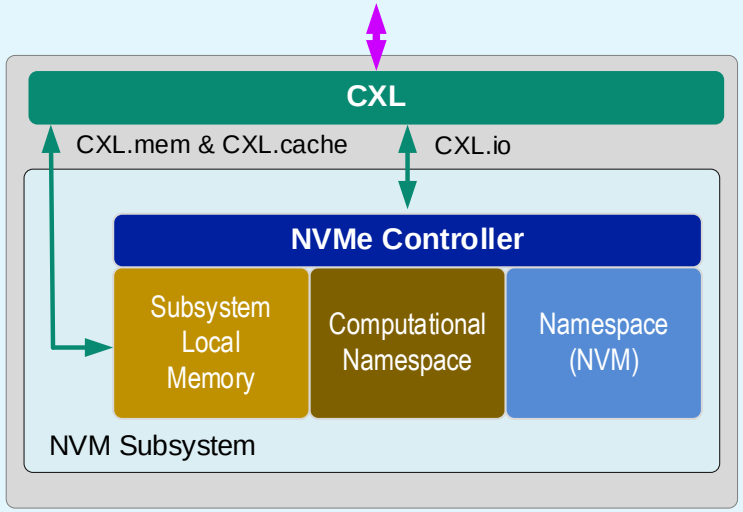
- Byte-addressable memory using CXL memory semantics (CXL.mem)
- Auto-save to NAND on power fail
- Runtime save to NAND using NVMe block commands

Byte-addressable SSD or Memory Expansion

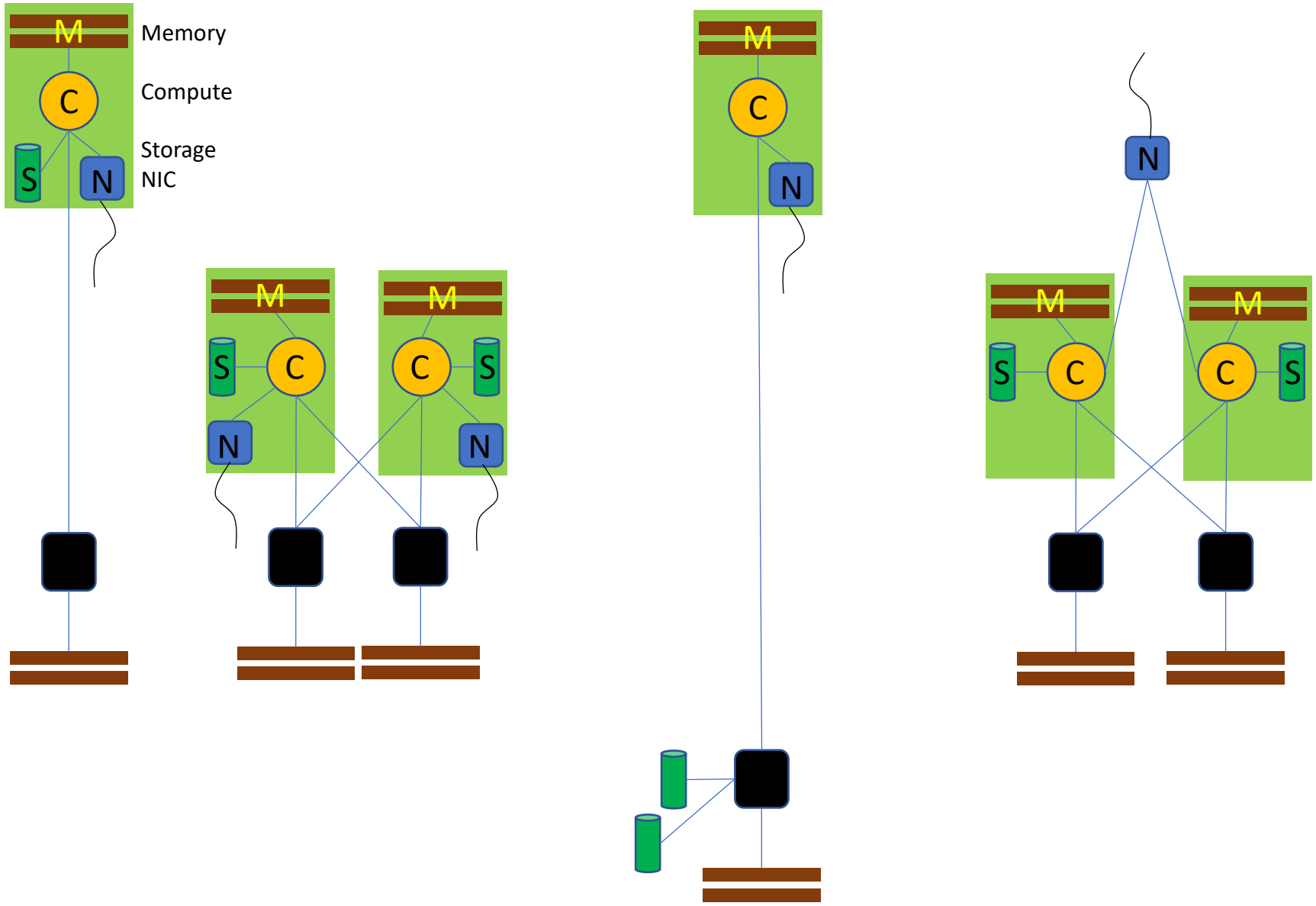


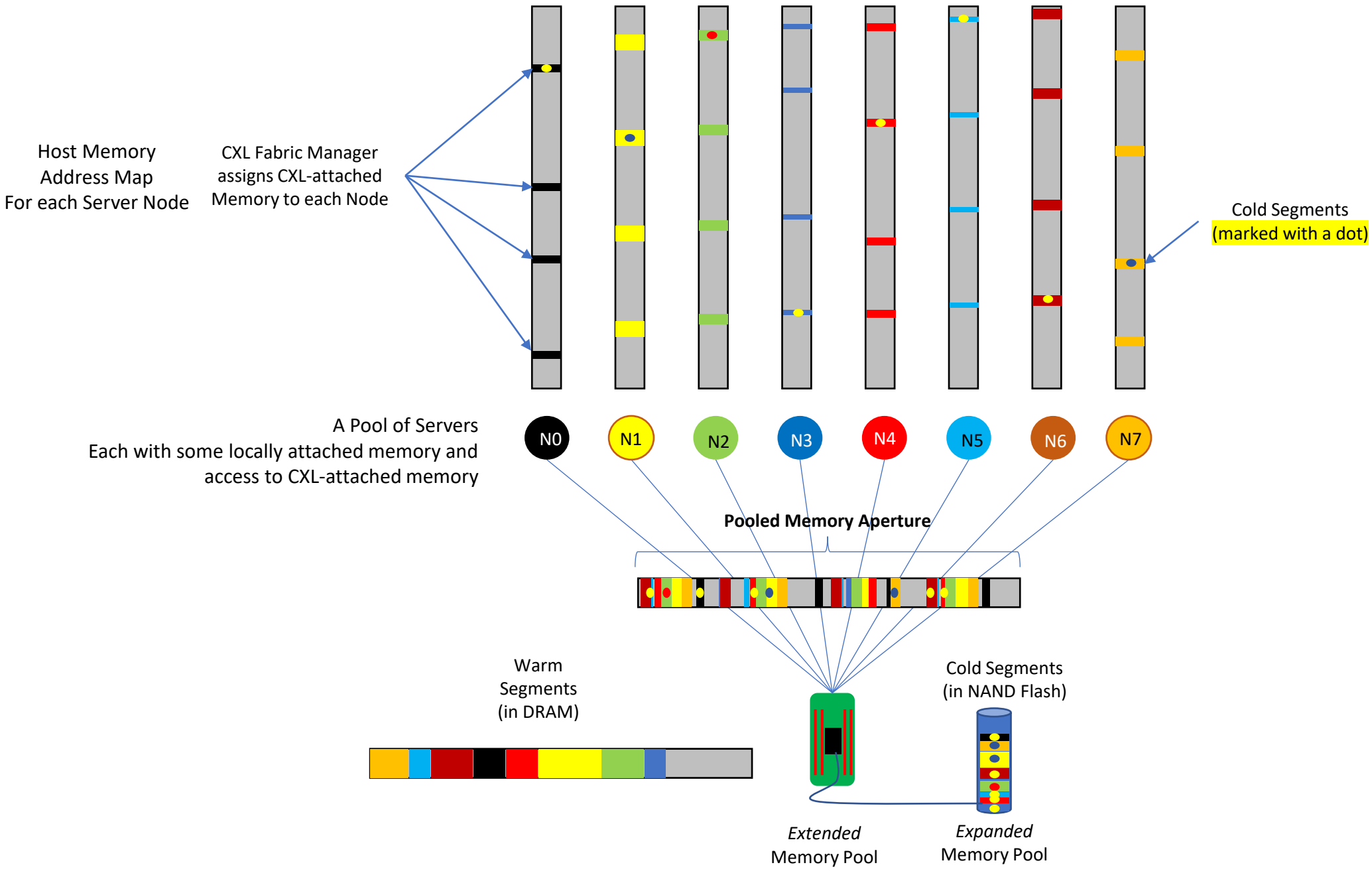
- Read/write NAND using NVMe block commands to/from memory
- Access memory using CXL memory semantics (CXL.mem)
- Host controlled page swapping between memory and NAND

Computational Storage



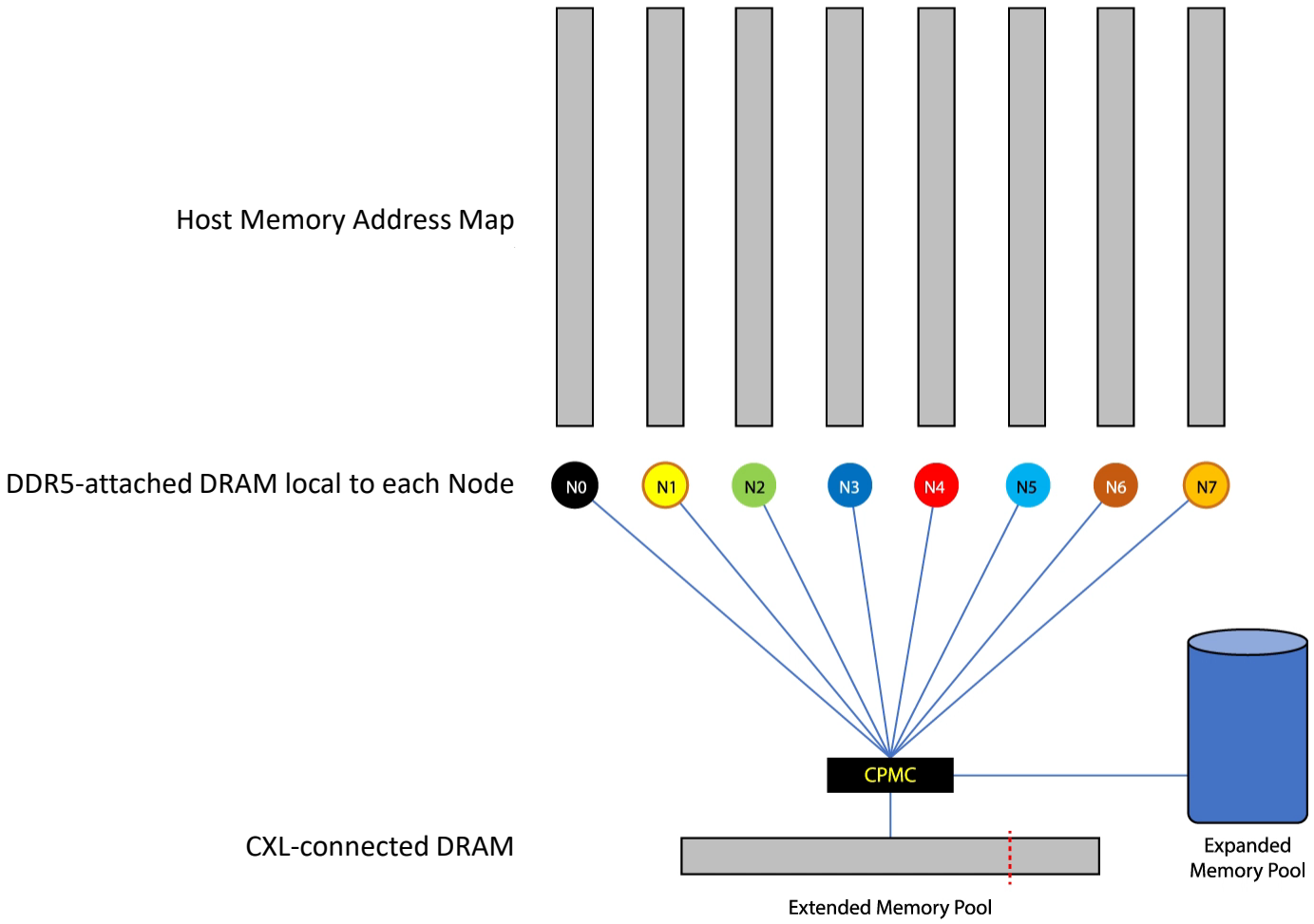
- NVMe is standardizing computational storage (i.e., NVMe Computational Programs)
- NVMe computation programs operate on “Subsystem Local Memory”
- Subsystem Local Memory may use CXL to be implemented as local memory accessible by host or host memory using cache operation


M
Memory
C
Compute
S
Storage
N
NIC


Host Memory
Address Map
For each Server Node
CXL Fabric Manager
assigns CXL-attached
Memory to each Node
Cold Segments
(marked with a dot)
A Pool of Servers
Each with some locally attached memory and
access to CXL-attached memory
N0
N1
N2
N3
N4
N5
N6
N7
Pooled Memory Aperture
Warm
Segments
(in DRAM)
Cold Segments
(in NAND Flash)
Extended
Memory Pool
Expanded
Memory Pool

Memory Serving Concept (*with Bulk Memory Expansion*)

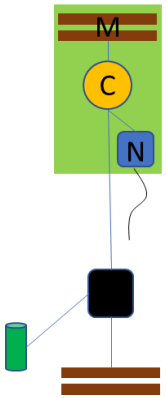
Animation by: Leili Arai Tavallaei (linkedin.com/in/leili-arai-tavallaei-3bb948160)

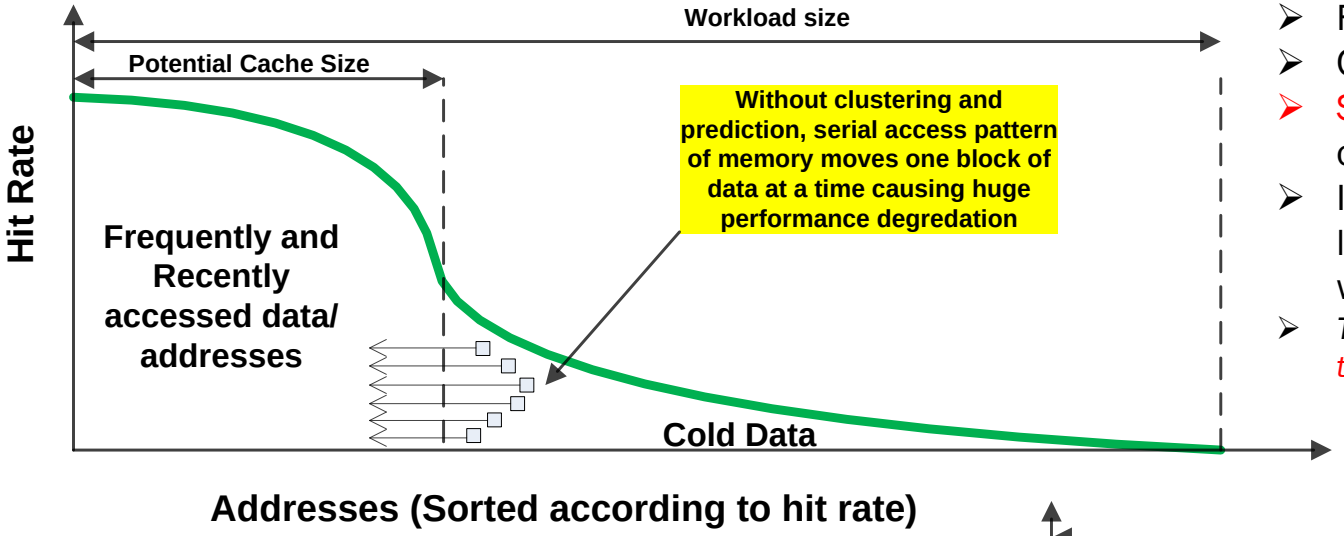


Reduce Long Tail Latency with Prediction when using NAND Flash

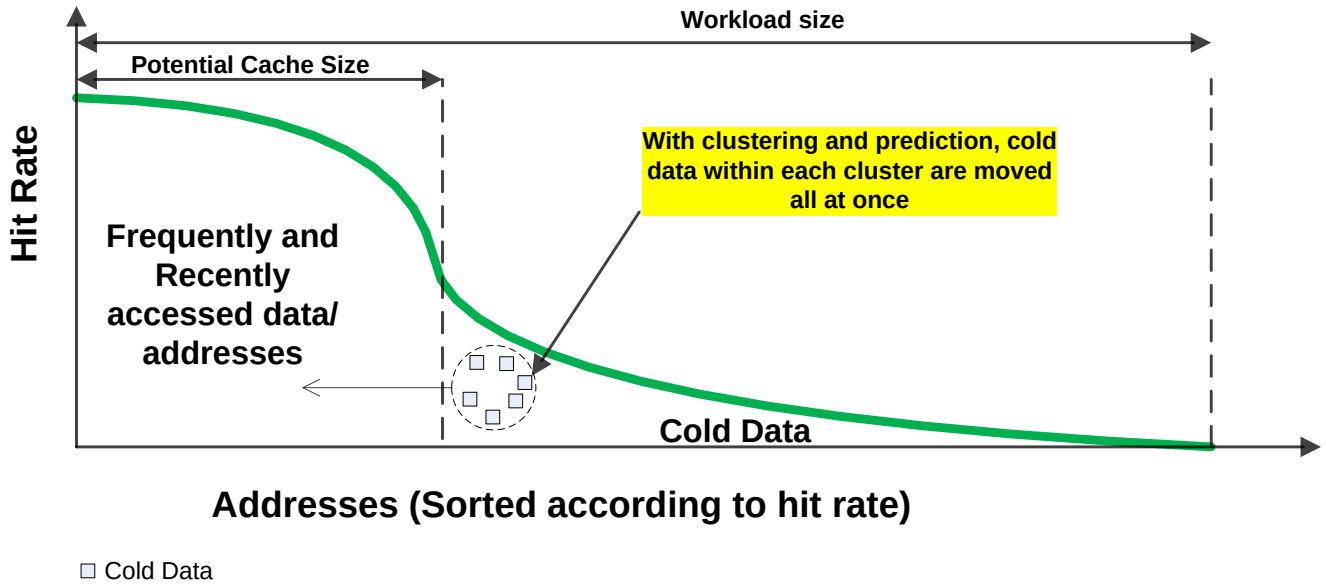
Courtesy: Maher Amer (linkedin.com/in/maheramerbionym)

- Traditional Cache Management algorithms use recency and frequency to manage data placement
- Frequently & recently accessed data goes to the fast mem tier
- Cold data goes to the slow memory tier
- Single-thread memory access is a serial process (accesses one address at a time)
- If the latency differential between the fast and slow tier is large (e.g., between DRAM and Flash), the cache performance will significantly suffer due to the long tail of cold data.
- *Traditional cache management algorithms don't have a solution to long tail access distribution problem!*





- Use AI to detect patterns and create clusters amongst cold data
- Cluster formation allows for the parallelization access of cold data
- Flash latency is amortized across each cluster size due to parallelization of access
- *Cluster learning and pattern detection addresses the long tail distribution of cold data and significantly reduces the impact of latency differential of multi-tier memory system*

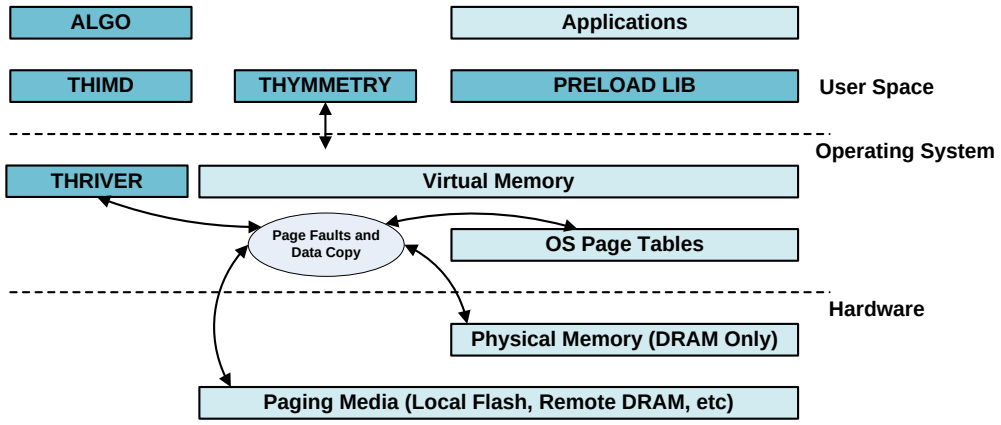


Courtesy: Maher Amer (linkedin.com/in/maheramerbionym)

Prediction Challenges and Prototype Overview: THRIVER

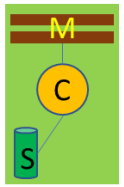
- Number of unique addresses is in the millions making collecting statistical information per address unrealistic/impossible
- Memory access patterns change so fast making it impossible to buffer the patterns and perform traditional clustering algorithms that rely on data being relatively static
- In multi-threaded environment, it is impossible to isolate patterns from one thread to another (at the HW level). As a result, traditional pattern detection algorithms can't perform data cleaning phase to isolate relevant data from irrelevant ones. (*NEED A PREDICTION ALGORITHM THAT IS PURELY DEPENDANT ON TEMPORAL LOCALITY*)

Prototype Overview



- THRIVER is a kernel module custom page fault handler
- THIMD is abstraction layer that connects THRIVER to ALGO
- ALGO performs prediction algorithm
- THYMMETRY is HOT/COLD tiering algorithm

HW Spec:



- 28 Cores Dual Socket Intel(R) Xeon(R) CPU E5-2683 v3 @ 2.00GHz
- Memory = 160 GB
- Paging Device NVMe = 1.5 TB INTEL SSDPEDME016T4

Results Summary

Workload	DRAM/ Flash	Faults Reduction: with/without Prediction	Prediction Rate	Prediction Accuracy	Performance No Prediction**	Performance With Prediction**
OpenFoam (HPC)	50/50	5.7x	82%	96%	0.55	0.90
OpenFoam (HPC)	25/75	10.2x	90%	95%	0.26	0.66
OPM (Oil and Gas) (HPC)	50/50	8.9x	92%	98%	0.43	0.77
ML Training	35/65	5.8x	80%	95%	0.29	0.59
SQLite TPCH (Data Analytics)	50/50	10.1x	90%	93%	0.14	0.63

Fault Reduction = #Faults without Prediction / #Faults with Prediction
Prediction Rate = #Predictions / (#Faults + #Predictions)
Performance No Prediction = Baseline Runtime* / No Prediction Runtime
Performance With Prediction = Baseline Runtime* / With Prediction Runtime

***Baseline = All data in DRAM**
****Baseline performance = 1**

Call to action (Storage Topics)

- The CXL and NVMe combined applications in the earlier slide can be built using existing capabilities in the CXL and NVMe specifications.

- For NVMe information refer to www.nvmexpress.org
- For CXL information refer to www.computeexpresslink.org

- If you have ideas about how CXL and NVMe could work better together, contact Peter Onufryk peter.onufryk@intel.com

- For page access prediction topics, reach out to Maher Amer (linkedin.com/in/maheramerbionym)

Backup

Taxonomy

- **Pooling** (dividing a resource to multiple non-overlapping units and assigning each to a different server/host
- **Sharing**
  - Serialized Sharing (a device may be fully mapped to a server at one time and to a different server at a different time)
  - Concurrent Sharing (multiple servers/hosts are assigned to the same portion of a device at the same time; coherence and access ordering may be enforced by hardware or software)
- **Borrowing** (Server B may get permission to access a portion of server A's resource. This portion of the resource leaves the coherence domain of server A to join that of server B.)
- **Local Disaggregation**
  - Multiple servers in one Chassis accessing shared, pooled, or borrowed set of resources
  - Using a multi-host capable **Switch** or via multi-ported End Devices
- **Logical Disaggregation** (**composing** several servers via a **Fabric** to provide access to shared, pooled, or borrowed resources)
- **Physical Disaggregation** (interconnected Chassis: Server Head-node, Expansion Chassis such as JBOD, JBOF, JBOG, JBOM)
- **Extended** Memory via increased memory **capacity** & **distance** of the same type of medium (e.g., DRAM with <3x latency)
- **Expanded** Memory via managing multiple **tiers** of memory (e.g., slow bulk memory & paging/swapping techniques)
- **Coherence** (enforced by HW or maintained by SW via access ordering sequences and appropriate flush mechanisms_
- **Scale-up** (single-host, homogeneous computing, scale via the same type of interconnect protocol)
- **Scale-ou**t (networked-based or via changing interconnect protocol, heterogeneous or distributed multi-server computing)

Fundamental Requirements Persist

We still need to deliver integrated hardware and software solutions which are

Useful
- Desirable

High Quality
- Secure
- Safe
- Reliable
- Available

Manageable
- Serviceable
- Diagnosable

Performant
Efficient (power, space, cost, time, complexity, …)

Especially when driving the solutions into large Datacenters

Enablers (Software and Firmware Ingredients)

CXL Fabric Manager

- Secure composability, allocation, on-lining/off-lining

Pre-boot Environment

- Discovery, enumeration, setup, …

CXL Bus/Class Driver

- Configuration, Resource Allocation

CXL Memory Device Driver

- Interactions with Bus/Class Driver, Fabric Manager, VMM, …
- RAS, Security, Fault-isolation, On-lining, Off-lining, …
- Error Isolation, Telemetry, Performance Monitoring

OS-specific Software

- VMM, Hypervisor
- VM Allocation, Orchestration, Fault-isolation & Recovery