Compute Express Link™ 2.0 Specification: Memory Pooling

Mahesh Wagh, Sr. Principal Engineer, Data Center Group, Intel, Co-Chair CXL™ Consortium Protocol Working Group

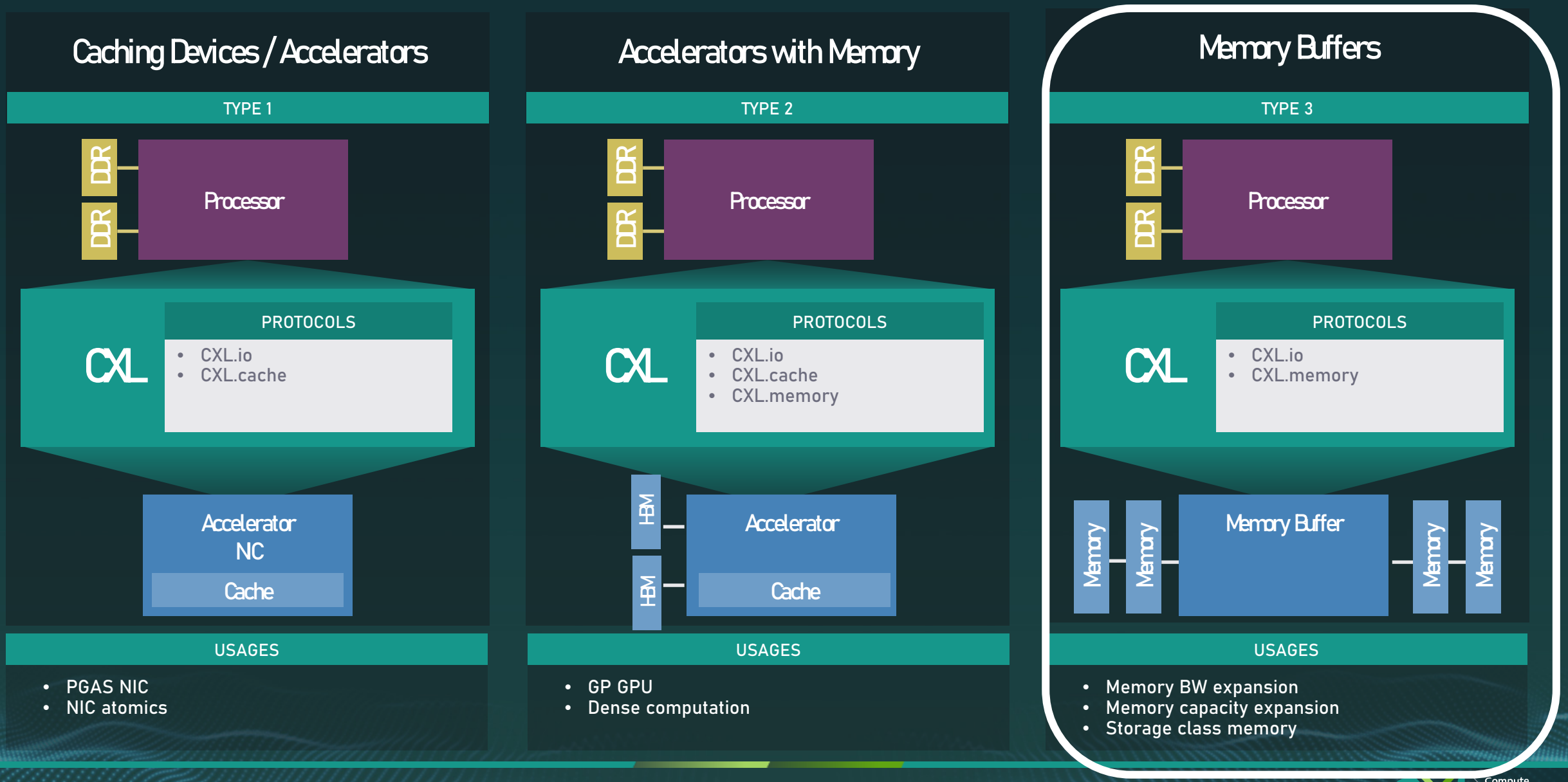
Rick Sodke, Associate Technical Fellow, Data Center Systems, Microchip

Agenda

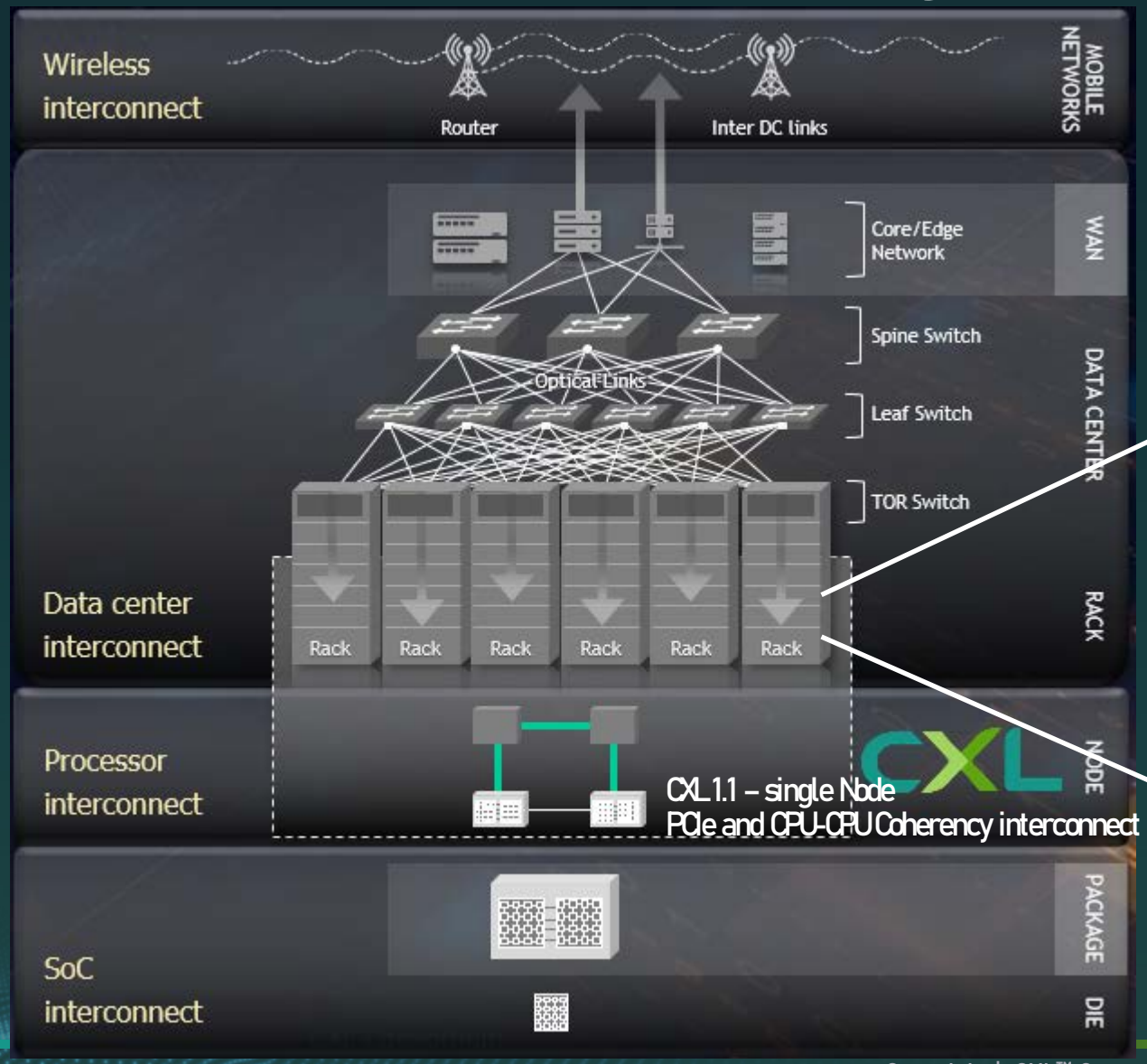
- CXL Introduction
- CXL Memory Expansion
- CXL Memory Pooling
- CXL Memory Pooling Allocation Examples

Representative CXL Usages

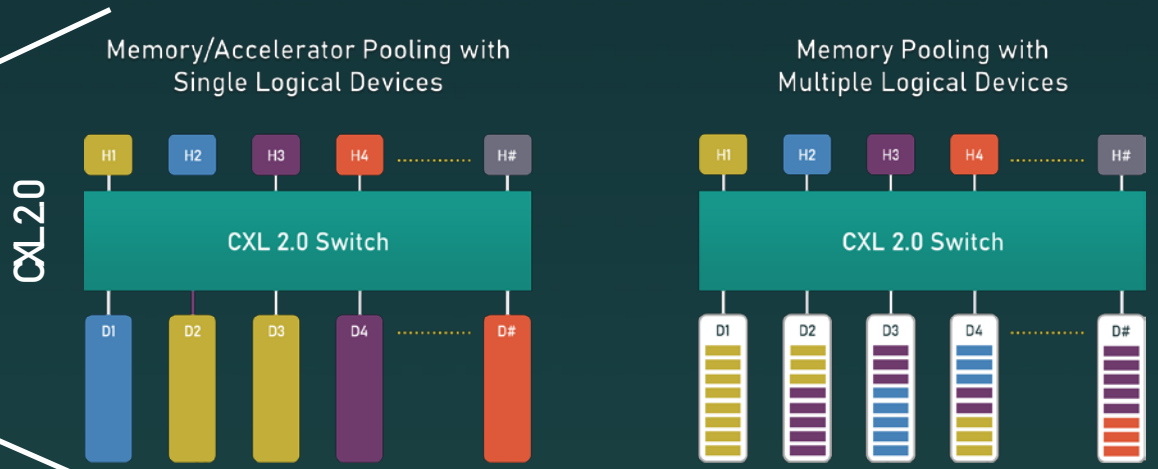
Topic will focus on Scaling & Effective Utilization of CXL Memory



Data Center: Looking Outside in: Scope of CXL 2.0

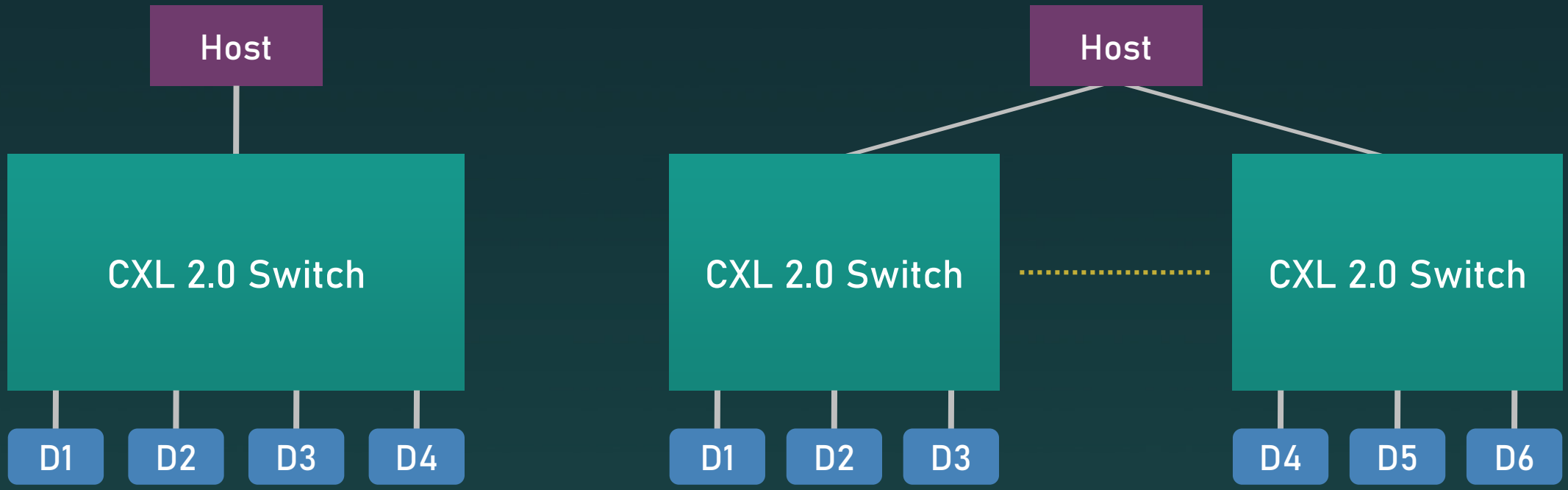


CXL 1.1 – single Node
PCIe and CPU-CPU Coherency interconnect



CXL 2.0 across Multiple Nodes inside a Rack/Chassis supporting pooling of resources

Memory Expansion with CXL 2.0 Switching



- Inclusive of Memory Interleaving
- Scaling by adding more switches

What is CXL Memory Pooling?

- A combination of Software (OS, Fabric Manager), Hardware (Platforms, Switches, Memory Devices) and Protocol (CXL) enhancements for efficient utilization of hardware resources by enabling dynamic management and allocation of resources – CXL Attached Memory.

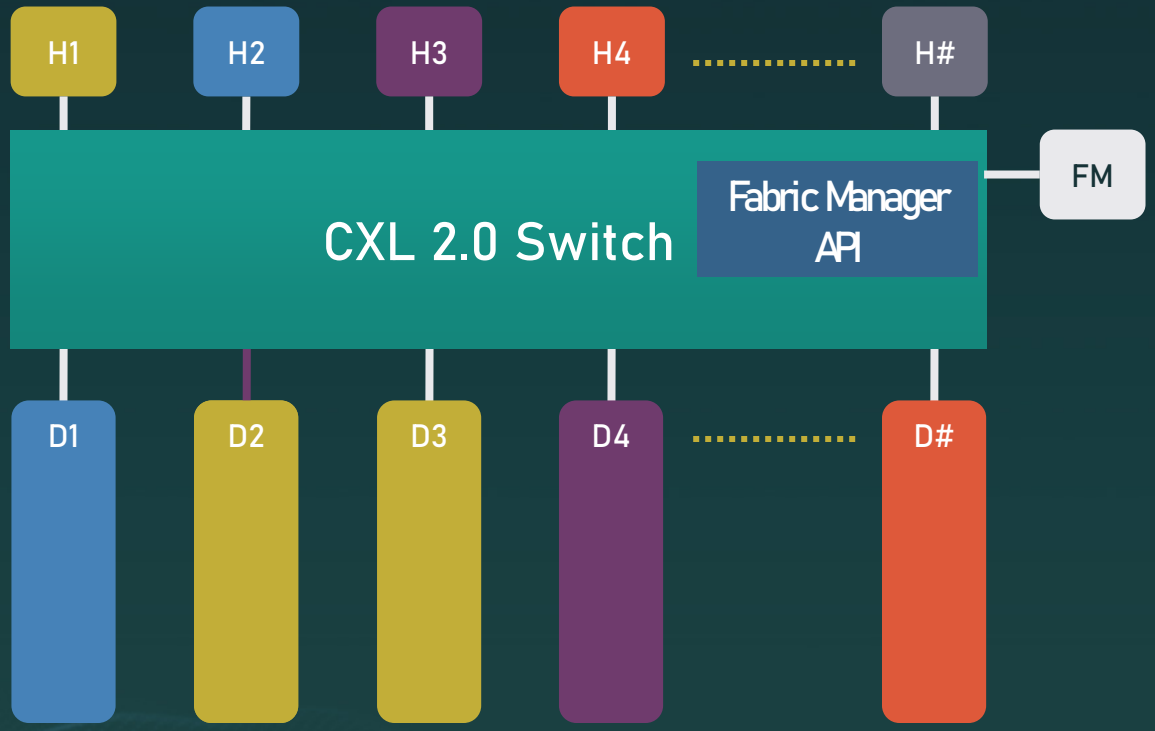
- Benefits
  - Effective utilization of memory resources within a system (Rack)
  - Dynamic Allocation/deallocation of memory resources.
  - Total Cost Of Ownership (TCO) Savings

Memory Pooling with CXL

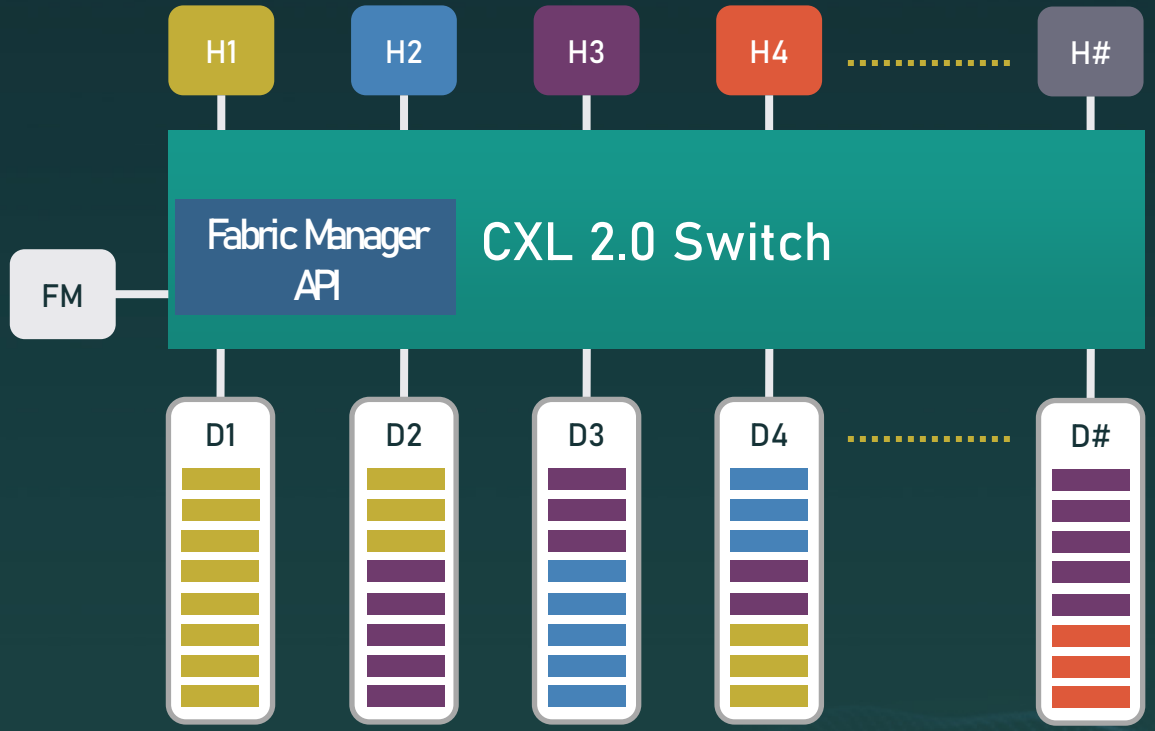
- The CXL Specification was developed with Memory Pooling as a primary use case
- Memory Pooling is supported with many different topologies including:
  - Pooling with Single Logical Devices
  - Pooling within Multi-Logical Devices
  - Pooling without a Switch
- CXL 2.0 defines a Fabric Manager Application Programming Interface that provides configuration and control capabilities supporting Pooling applications

CXL 2.0 Memory Pooling

Memory/Accelerator Pooling with Single Logical Devices



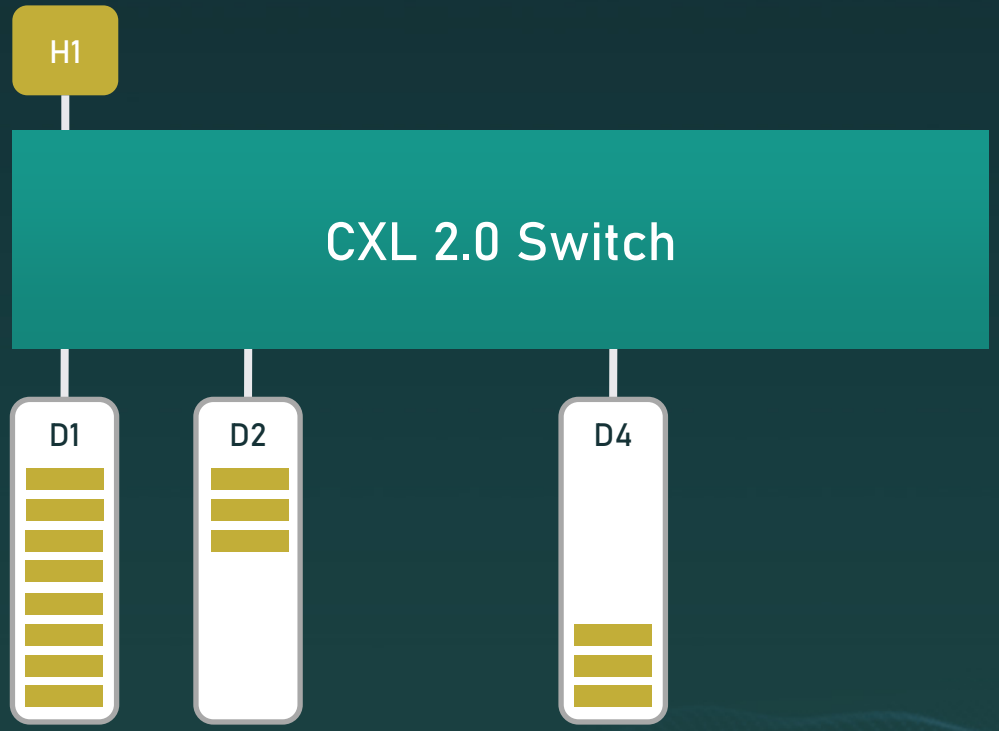
Memory Pooling with Multi-Logical Devices



Per OS/VMM View of CXL Memory

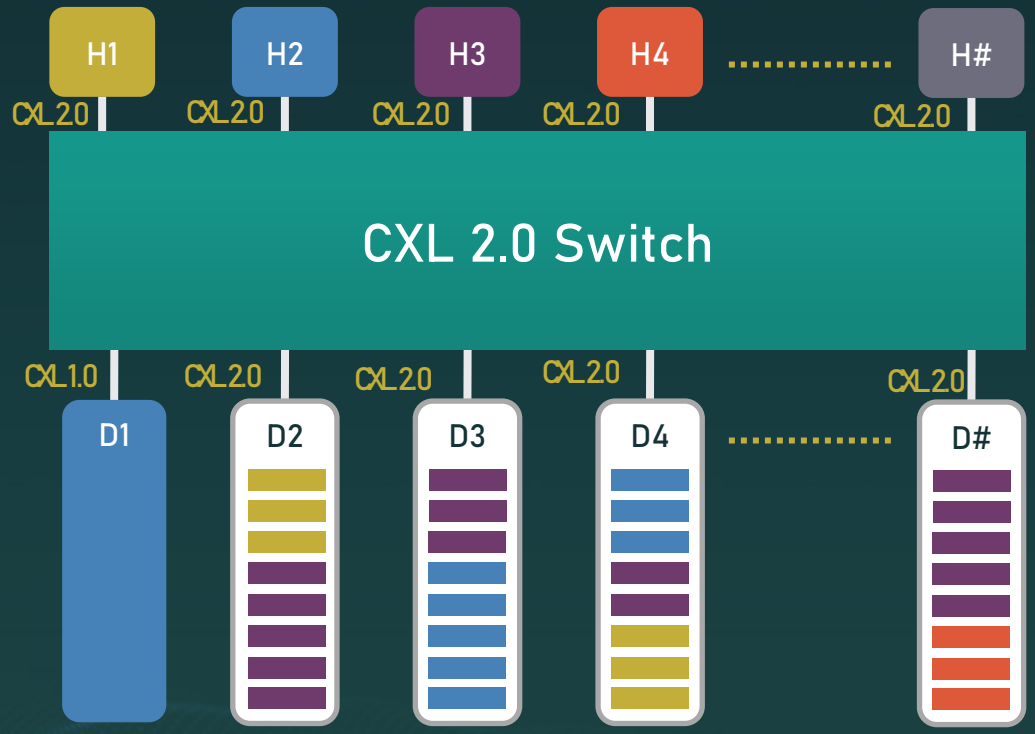
Memory/Accelerator Pooling with Single Logical Devices

Memory Pooling with Multi-Logical Devices

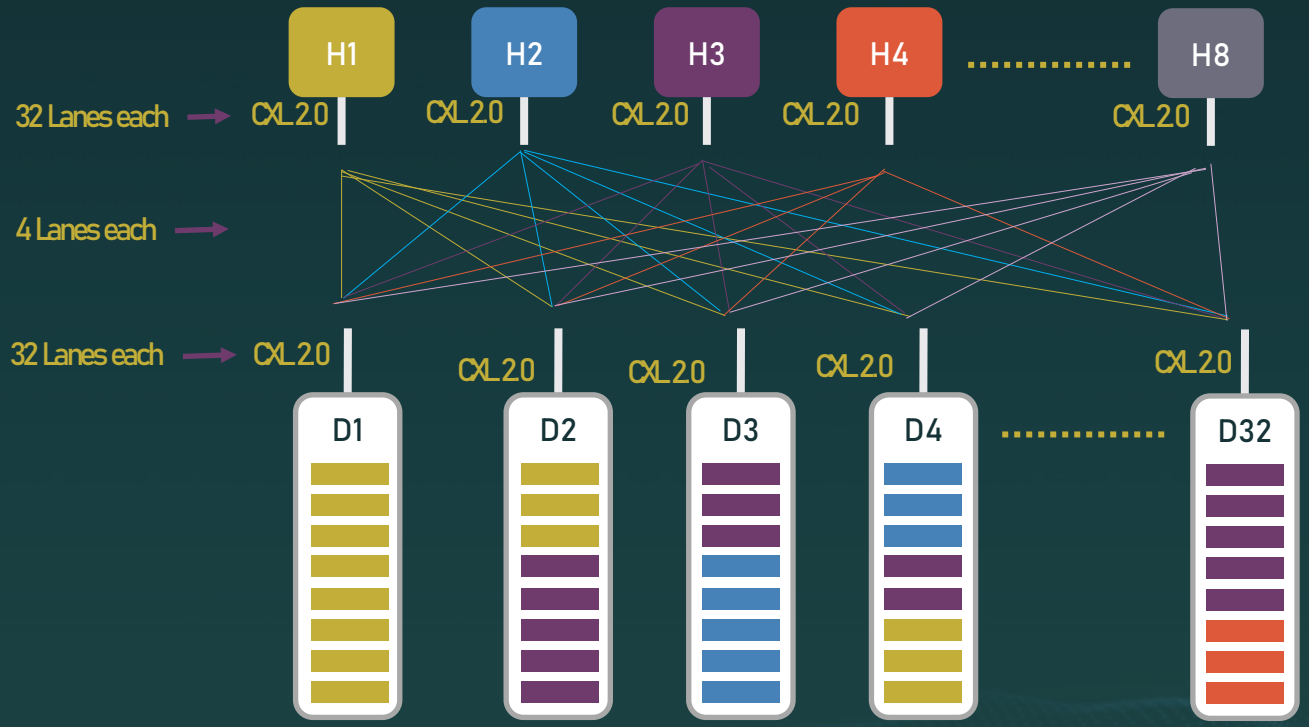


Example of CXL 2.0 Memory Pooling without a Switch

Memory Pooling with Single/ Multiple Logical Devices

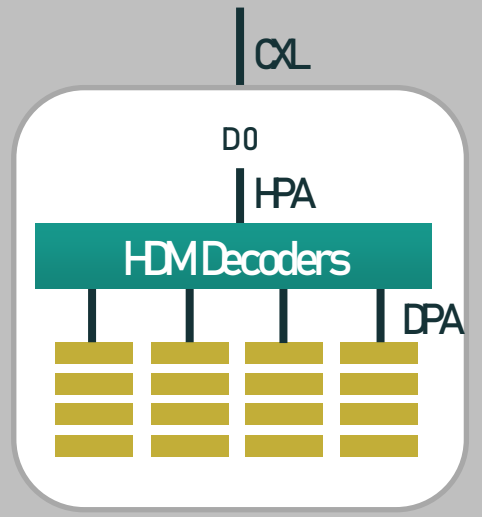


Memory Pooling with Multi Ported Device through direct connect



CXL 2.0 SLD Memory Device

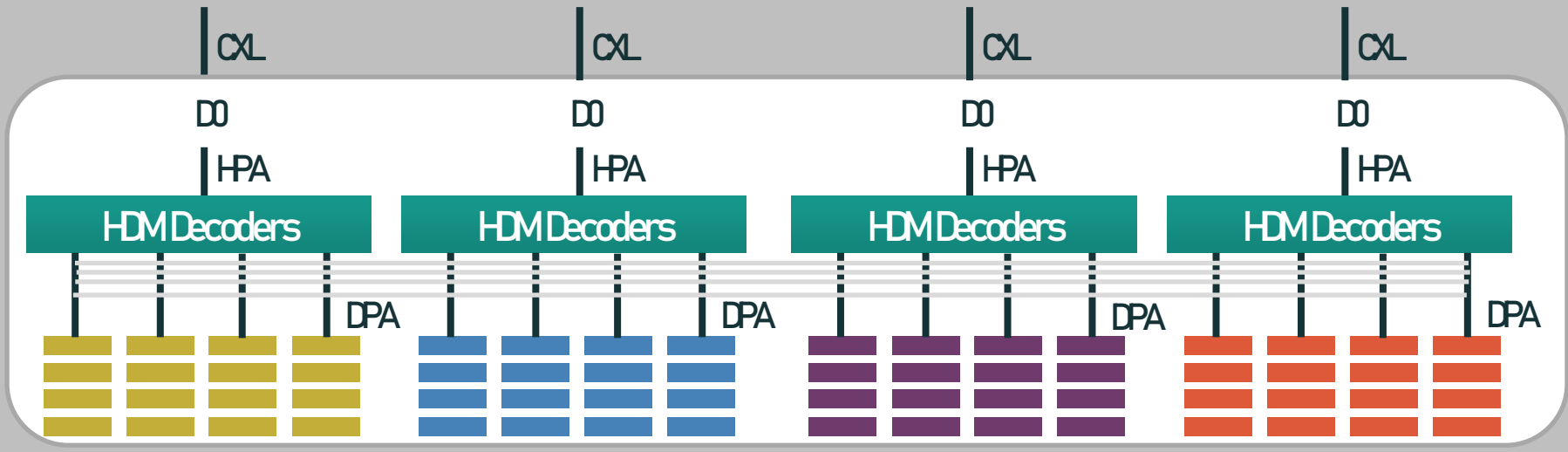
- Support one or more PCIe Endpoint Functions
    - Type 0 header in PCI Configuration space
- Primary function (device number 0, function number 0) must carry one instance of CXL DVSEC ID 0 with Revision 1 or greater.
- Non-CXL Function Map DVSEC to advertise Non-CXL functions
- Must support operating in CXL 1.1 mode
    - PCIe Endpoint → RCIEP
- Type 3 device Component Register Block includes HDM Decoder registers
- Connected to a Single Virtual Hierarchy



HPA – Host Physical Address
DPA – Device Physical Address

CXL 2.0 Multi-Ported Device

Pooled Memory Device



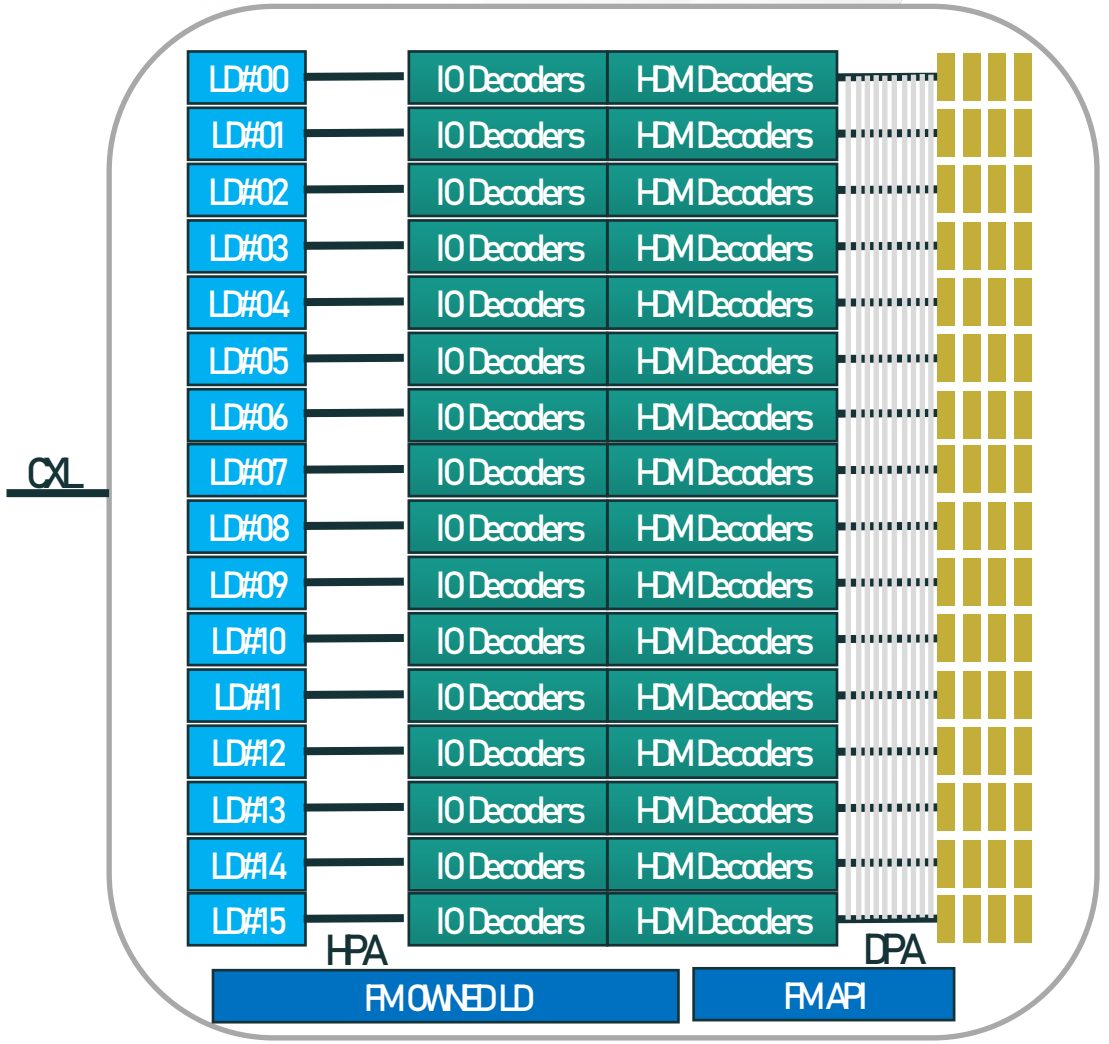
HPA – Host Physical Address
DPA – Device Physical Address

- Represents an SLD behind each CXL Port
- Device Vendor-Specific mechanisms to configure resources per SLD
- Example – 4 Ported CXL Memory Device with equal allocation of pooled resources across ports

CXL 2.0 Multi-Logical Device

Pooled Memory Device

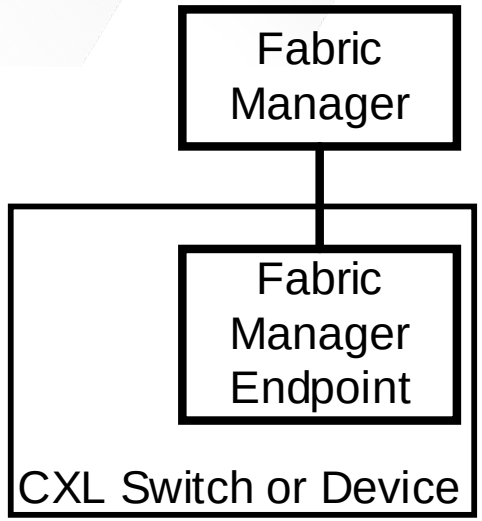
- A Pooled Type 3 device can partition its resources into Logical Devices (LD)
  - Up to 16 LDs (Type 3 only) AND
  - One Fabric Manager (FM) owned LD
- Each LD
  - Appears as Type 3 SLD device
  - Identified by LD-ID
  - FM binds each LD to a Virtual Hierarchy
- FM Owned LD
  - Accessible by FM only by using LD-ID of 0xFFFFh
  - Manage Link and Device
  - Memory resources are not assigned to LD owned LD
  - Error messages generated by LD are routed to FM
  - Does not participate in GPF Flows
- MLD Link
  - MLD Link Discovery & Link Operation configured via Alternate Protocol Negotiation



HPA – Host Physical Address DPA – Device Physical Address

Fabric Manager Endpoint and API

- Fabric Manager is a control entity that manages the CXL 2.0 Switch and the Memory Controller
  - FM can be an external BMC, a Host, or Firmware internal to the Switch
- FM Endpoint is a required feature for any switch that supports MLD ports or that supports dynamic SLD port binding
- FM API is the standardized interface for the FM to communicate with devices
- FM API uses an MCTP interface between Fabric Manager and devices
  - MCTP physical interface is switch vendor specific but could be PCIe, CXL.io VDM, SMBus, Ethernet, UART, USB, internal, …
- In general there are no real-time response requirements for the Fabric Manager so it needn't be performant

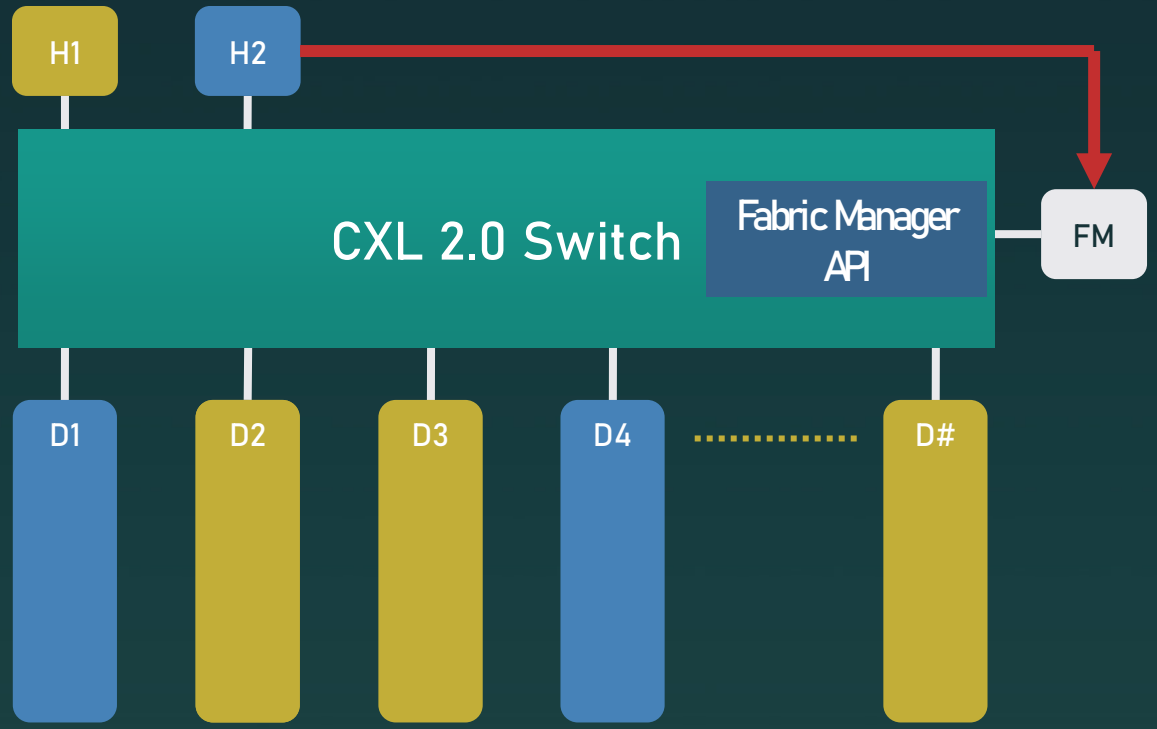


Fabric Manager

- Fabric Manager plays a critical role in CXL for systems supporting Memory Pooling
- The Fabric Manager enables dynamic system changes supporting Memory disaggregation
- Some examples:
  - Managing all devices that support traffic from multiple Hosts including:
    - Downstream ports connected to MLD ports
    - FM-owned Logical Device within an MLD component
    - Unbinding and rebinding of Logical Devices within an MLD between Hosts
  - Unbinding and rebinding of an SLD
  - Re-allocation of memory within an MLD
  - Re-allocation of memory within a multi-port SLD

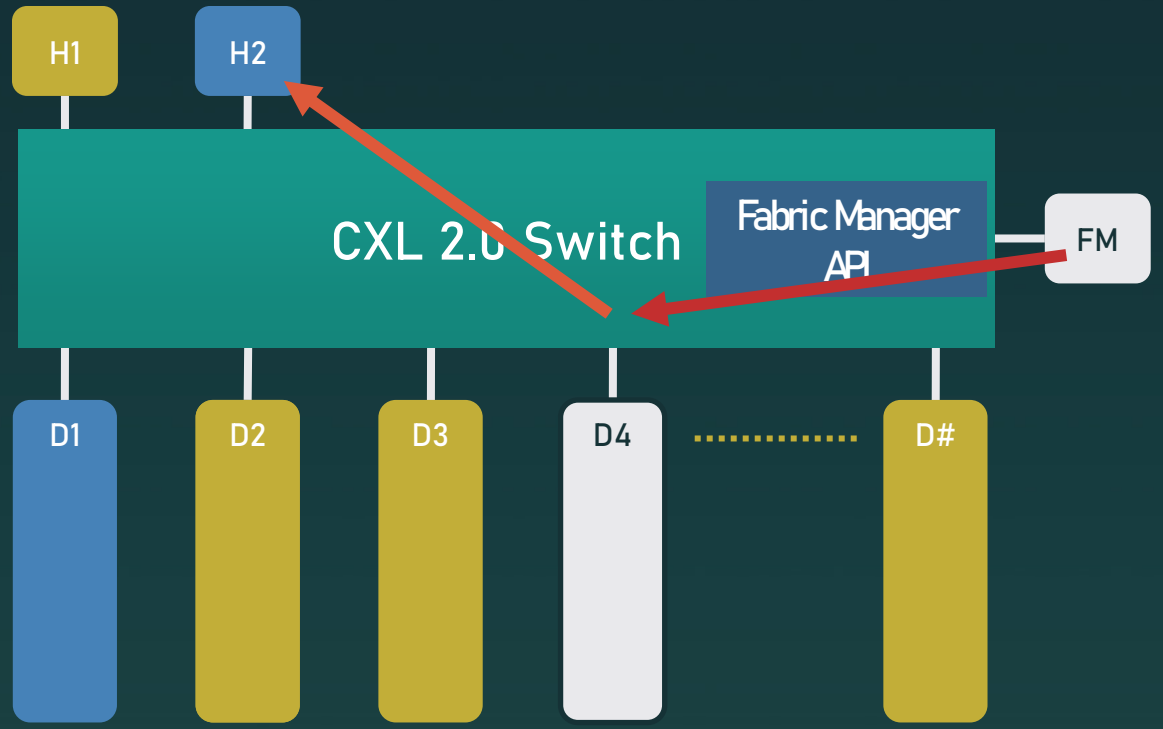
Memory Pooling with Single Logical Devices

H2 notifies FM that D4 memory is no longer needed



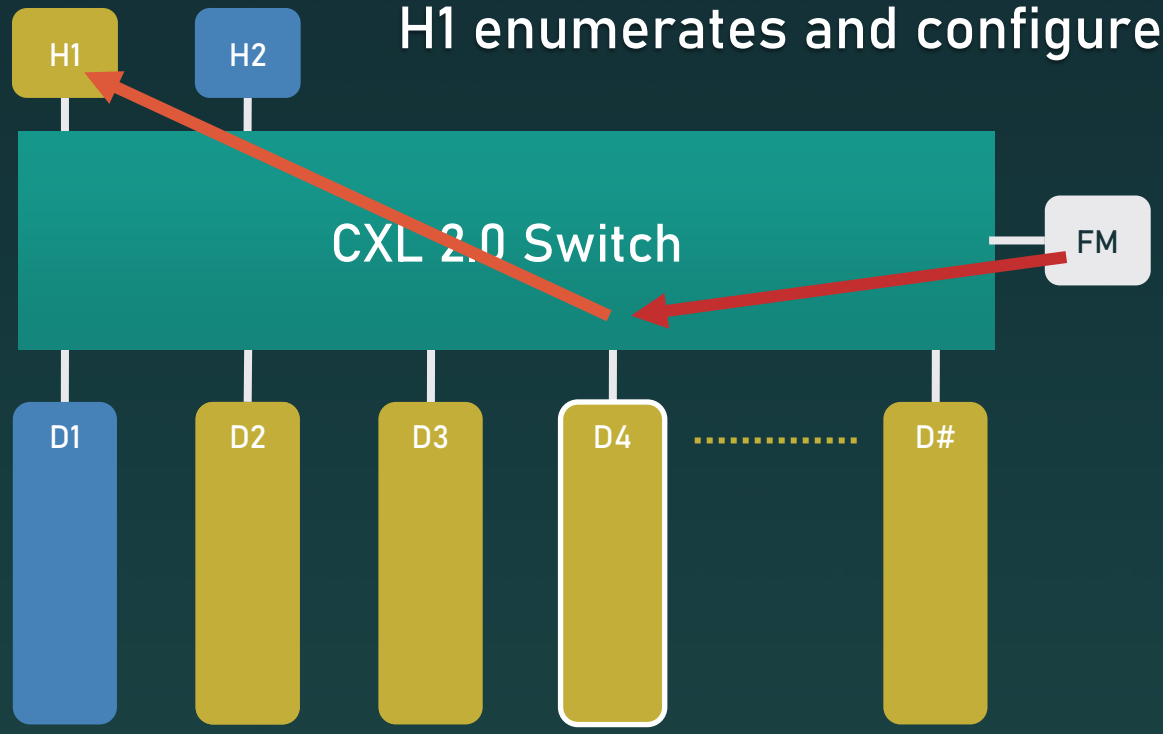
Memory Pooling with Single Logical Devices

FM tells switch to UNBIND D4
Switch notifies H2 of the managed hot remove

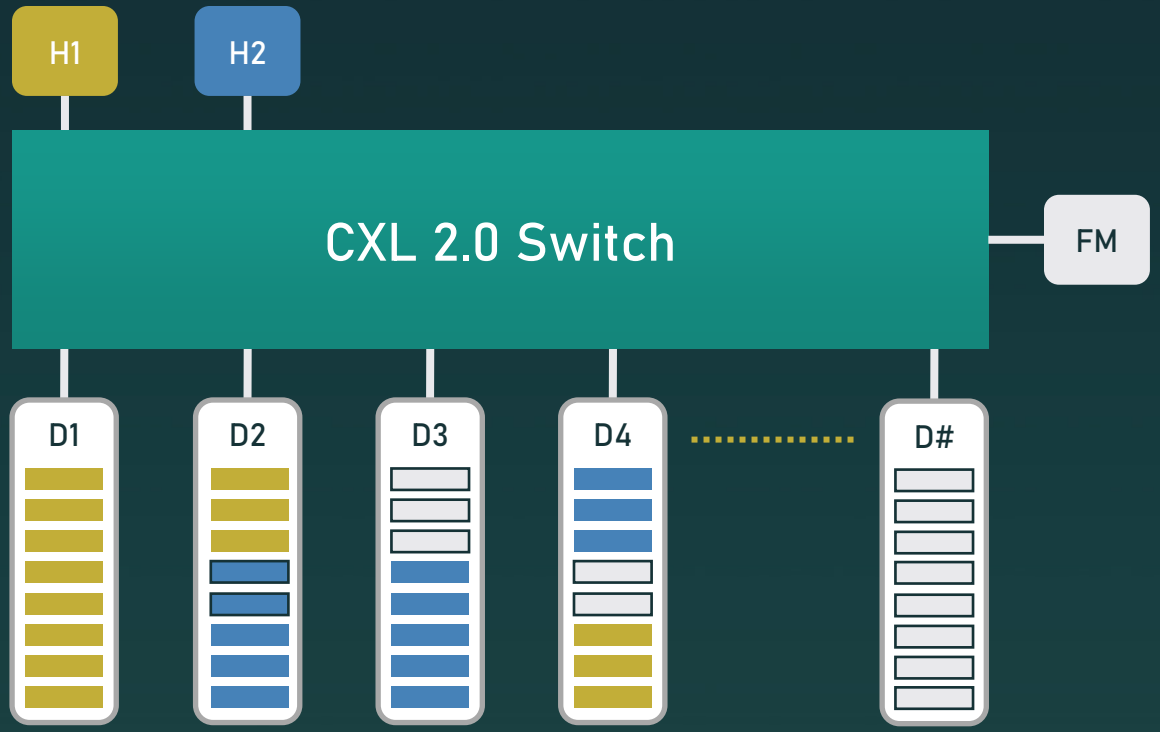


Memory Pooling with Single Logical Devices

FM tells switch to BIND D4 to H1
Switch notifies H1 using managed hot add
H1 enumerates and configures accesses to D4

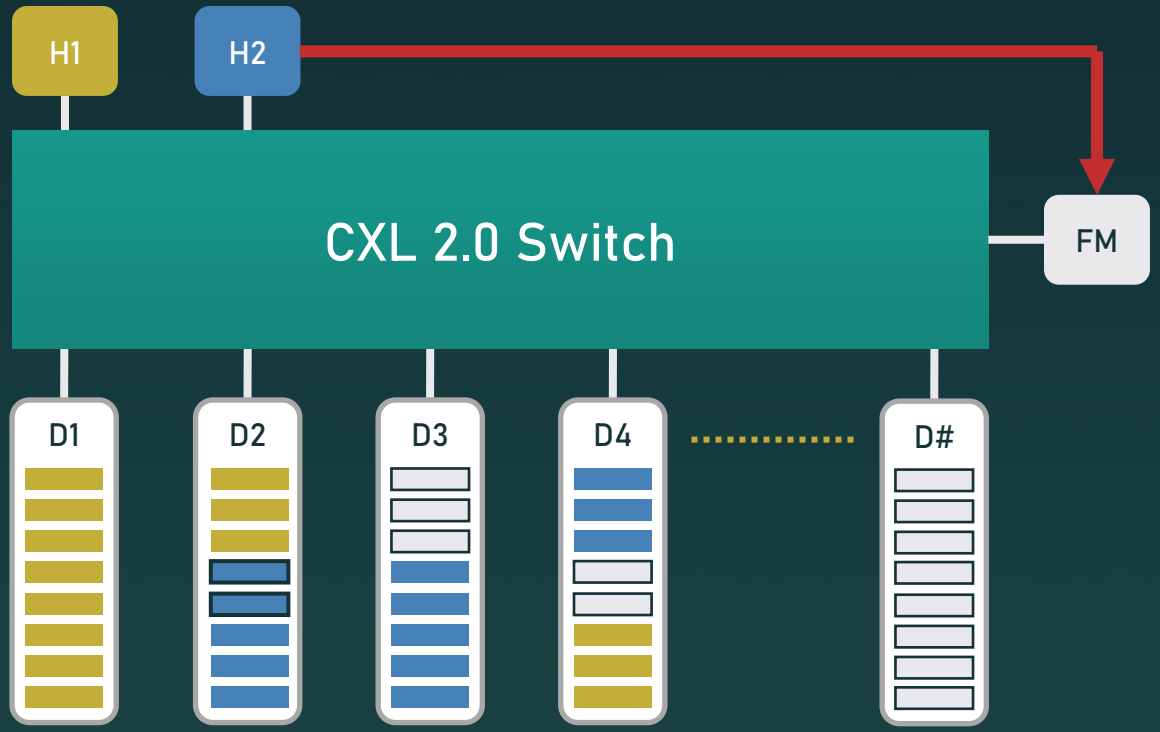


Memory Pooling with Multi-Logical Devices



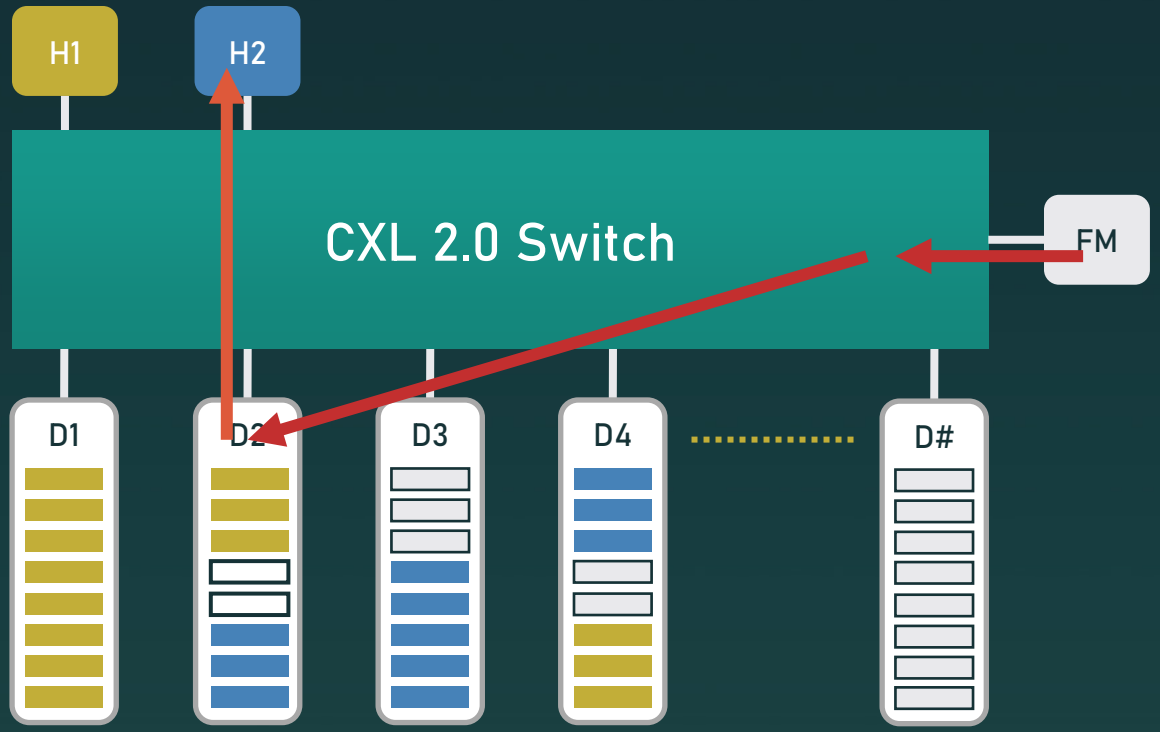
Memory Pooling with Multi-Logical Devices

H2 notifies FM that some D2 memory is no longer needed



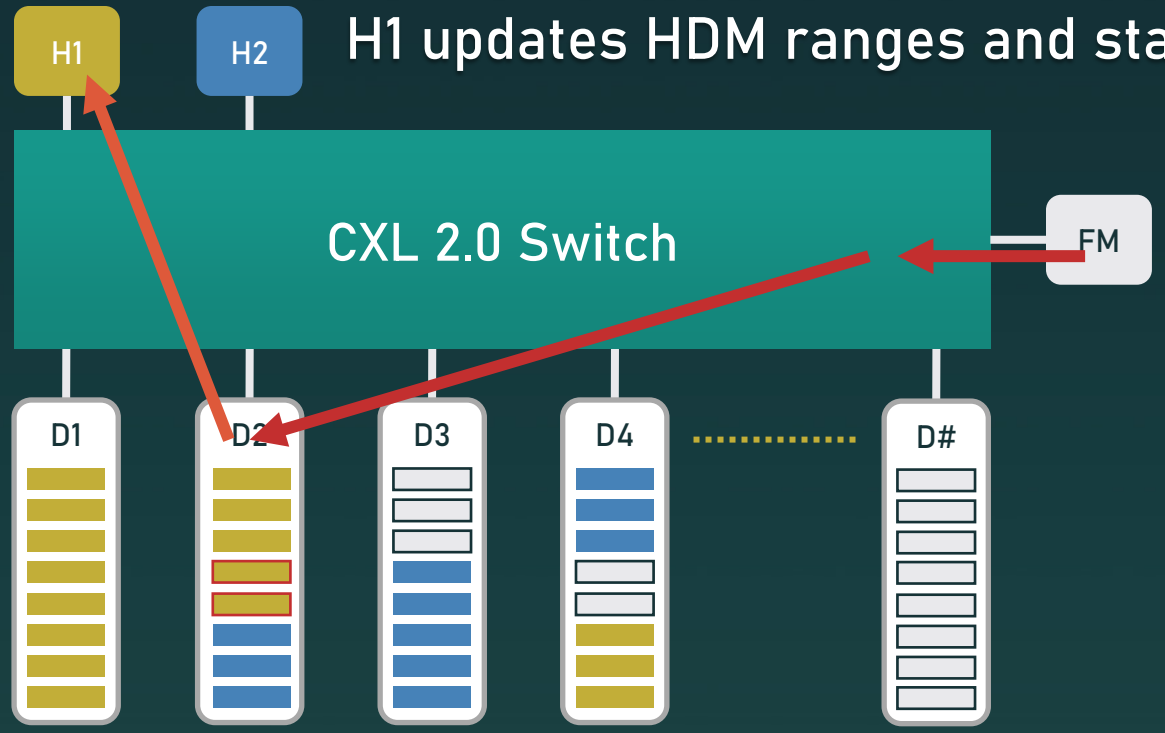
Memory Pooling with Multi-Logical Devices

FM tells D2 to de-allocate some blue memory
D2 notifies H2



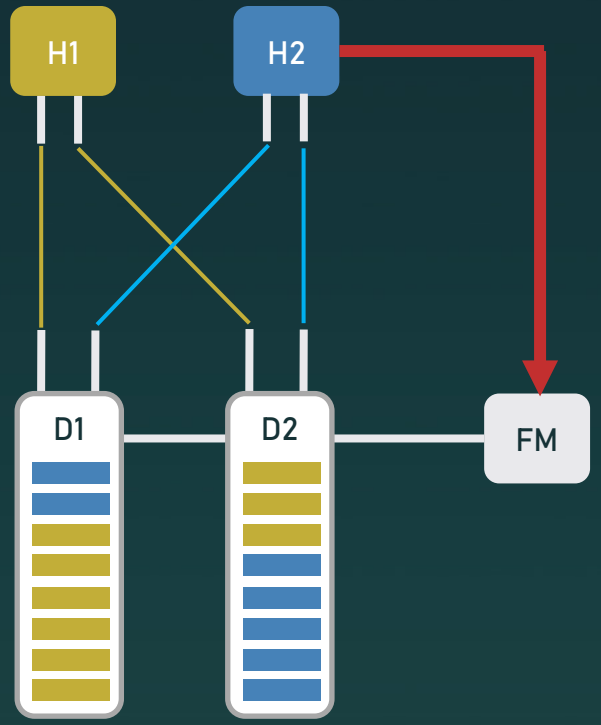
Memory Pooling with Multi-Logical Devices

FM tells D2 to allocate some yellow memory
D2 notifies H1
H1 updates HDM ranges and starts using memory



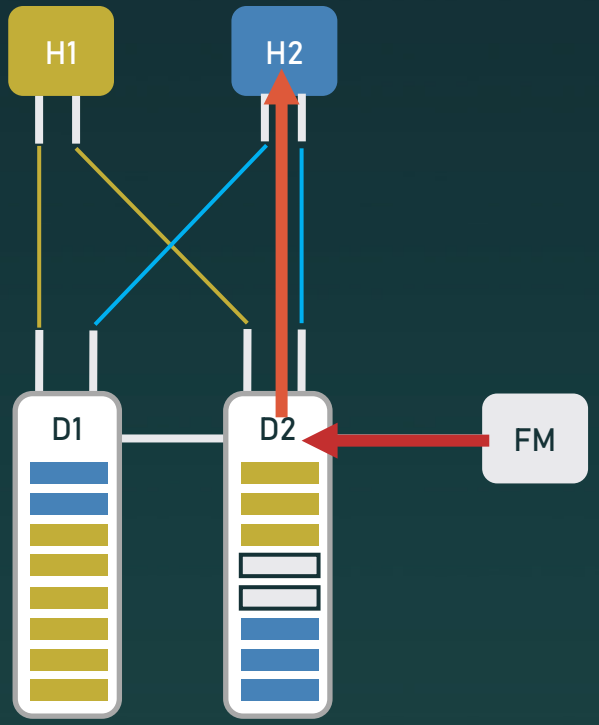
Memory Pooling without a Switch

H2 notifies FM that some D2 memory is no longer needed



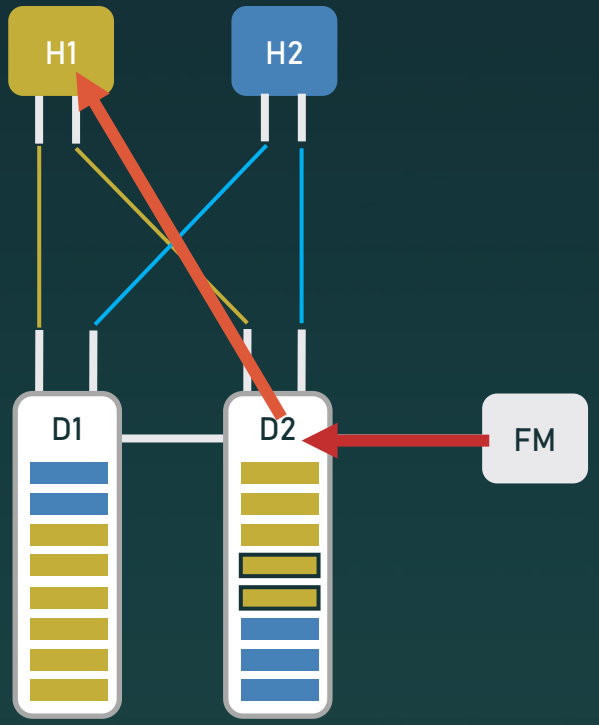
Memory Pooling without a Switch

FM tells D2 to de-allocate some blue memory
D2 notifies H2



Memory Pooling without a Switch

FM tells D2 to allocate some yellow memory
D2 notifies H1, H1 updates HDM ranges



Memory Pooling with CXL

- Other FM API features beyond BIND, UNBIND, and SET LD ALLOCATIONS:
  - Switch Discovery including capacity, capabilities, and connected devices
  - Event notification such as switch link events and Advanced Error Reporting for FM owned resources
  - Manage MLD QoS parameters

- Summary: Benefits of Memory Pooling
  - Effective utilization of memory resources within a system
  - Dynamic Allocation/deallocation of memory resources
  - Total Cost Of Ownership (TCO) savings

Thank You