STORAGE DEVELOPER CONFERENCE
SDC 21
BY Developers FOR Developers

Virtual Conference
September 28-29, 2021

Compute Express Link™ 2.0: A High-Performance Interconnect for Memory Pooling

Presented by Andy Rudoff, Persistent Memory SW Architect, Intel

Compute Express Link™

A New Class of Interconnect

CXL Consortium

CXL Board of Directors

Alibaba Group AMD arm cisco DELL EMC

facebook Google HUAWEI Hewlett Packard Enterprise

IBM intel MICROCHIP Microsoft XILINX

Open Industry Standard for Cache Coherent Interconnect

160+ Member Companies

Compute Express Link

- CXL 1.1
  - June 2019
- CXL 2.0
  - Nov 2020
  - 1.1 Compatible
  - Adds Pooling
  - Adds PMem

- www.computeexpresslink.org

Introducing CXL

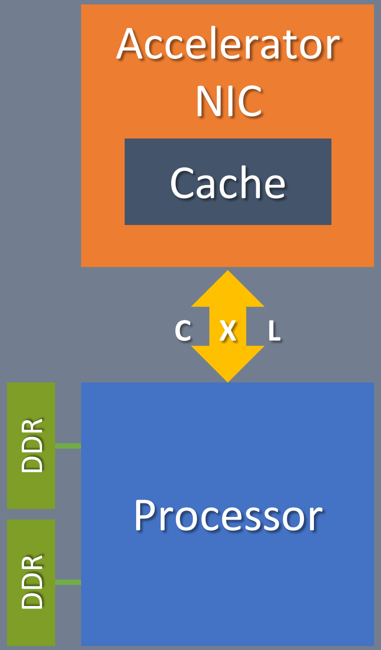
- Open industry standard for high bandwidth, low-latency interconnect
- Connectivity between host processor and accelerators/ memory device/ smart NIC
- Addresses high-performance computational workloads across AI, ML, HPC, and Comms segments
  - Heterogeneous processing: scalar, vector, matrix, spatial architectures spanning CPU, GPU, FPGA
  - Memory device connectivity
  - PCIe PHY completely leveraged with additional latency optimization
  - Dynamic multiplexing of 3 protocols
- Based on PCIe® 5.0 PHY infrastructure
  - Leverages channel, retimers, PHY, Logical, Protocols
  - CXL.io – I/O semantics, similar to PCIe - mandatory
  - CXL.cache – Caching Semantics – optional
  - CXL.memory – Memory semantics - optional

Representative CXL Usages

Caching Devices / Accelerators

Usages:
- PGAS NIC
- NIC atomics

Protocols:
- CXL.io
- CXL.cache

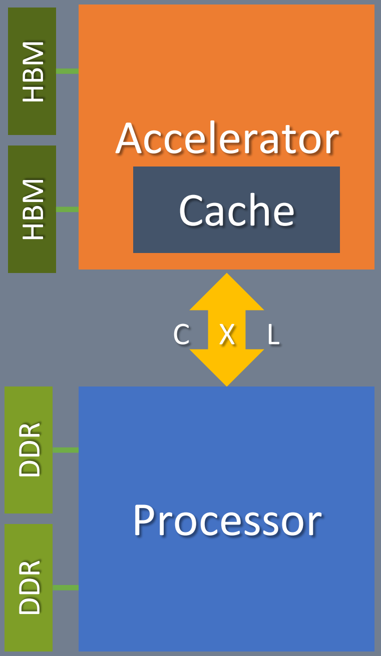


(Type 1 Device)

Accelerators with Memory

Usages:
- GPU
- FPGA
- Dense Computation

Protocols:
- CXL.io
- CXL.cache
- CXL.memory

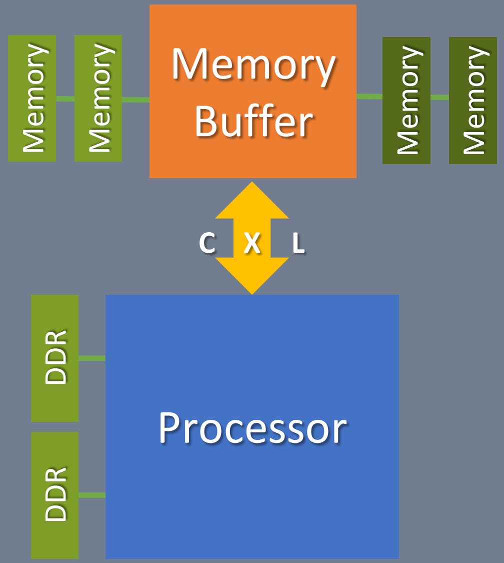


(Type 2 Device)

Memory Buffers

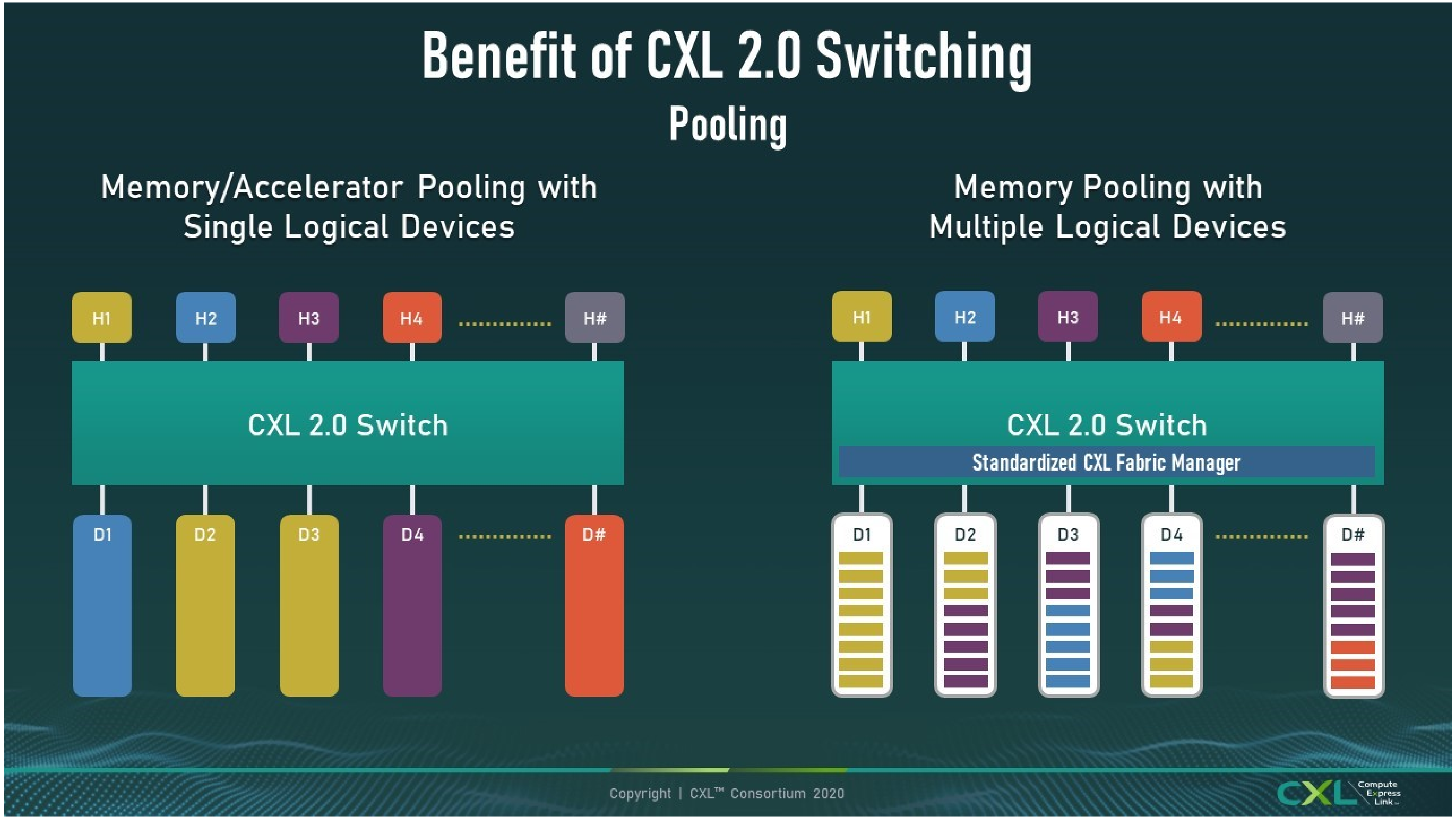
Usages:
- Memory BW expansion
- Memory capacity expansion
- 2LM

Protocols:
- CXL.io
- CXL.mem



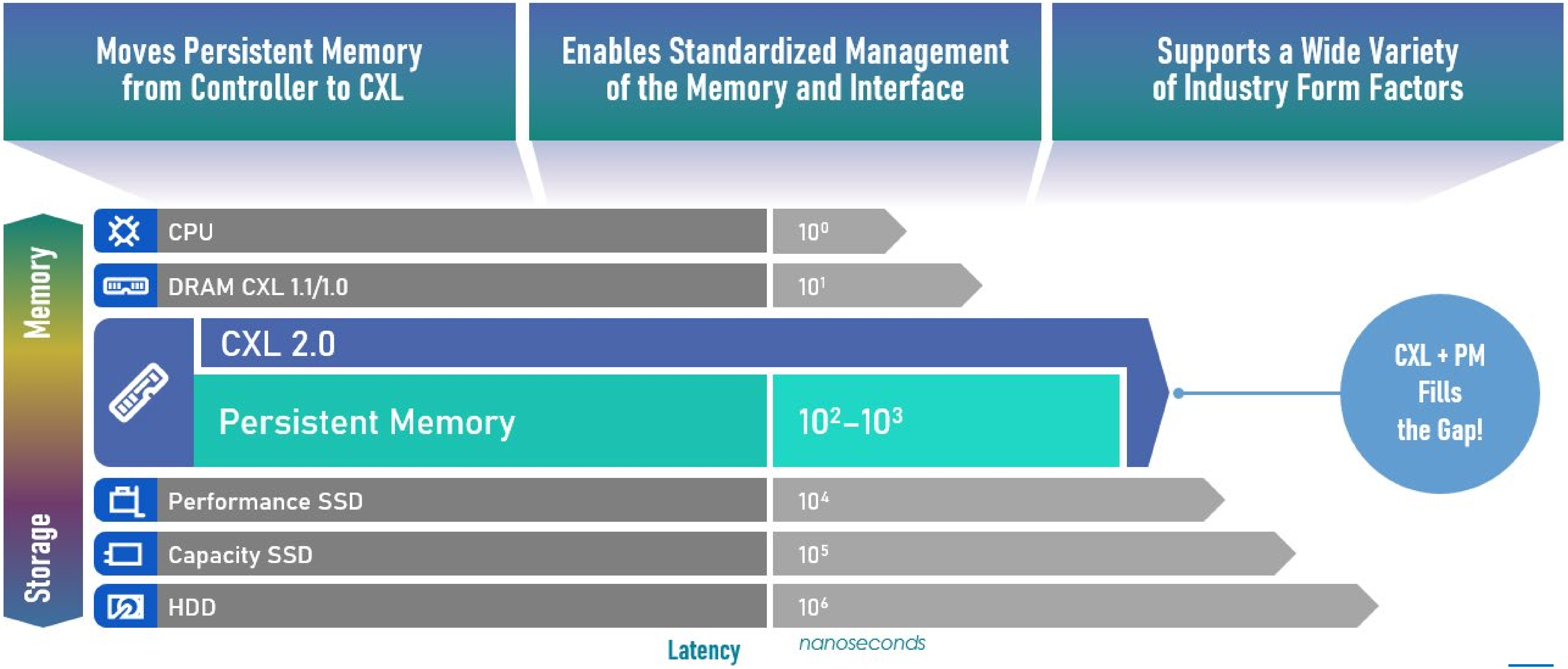
(Type 3 Device)

CXL 2.0 Memory Pooling


Benefit of CXL 2.0 Switching
Pooling
Memory/Accelerator Pooling with Single Logical Devices
Memory Pooling with Multiple Logical Devices
H1
H2
H3
H4
H#
CXL 2.0 Switch
Standardized CXL Fabric Manager
D1
D2
D3
D4
D#
Copyright | CXL™ Consortium 2020
CXL Compute Express Link

STORAGE DEVELOPER CONFERENCE SDC 21

Benefits of Persistent Memory

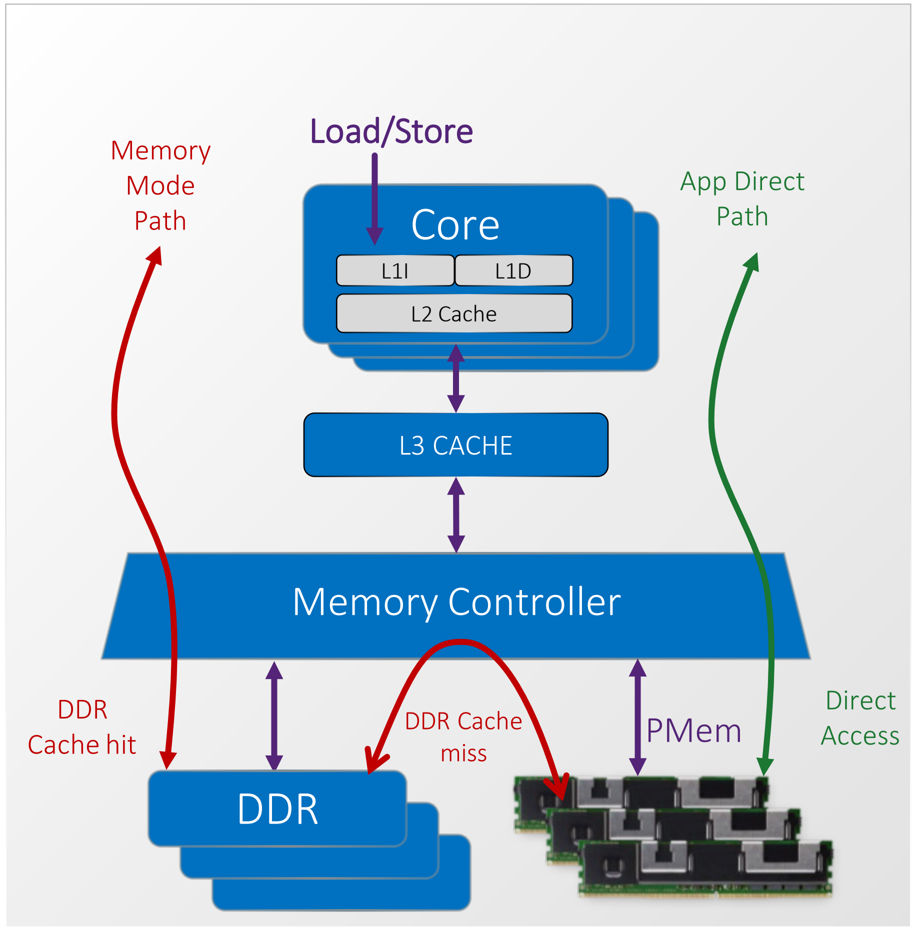


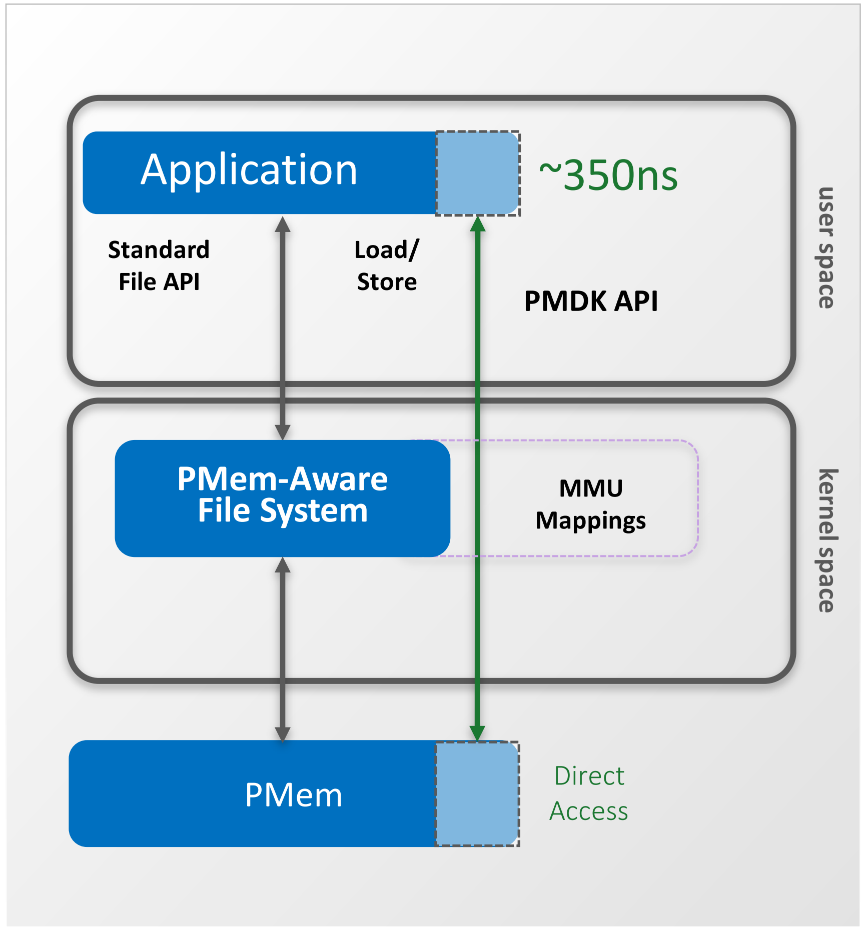
Persistent Memory Today

Connecting to the Memory Bus

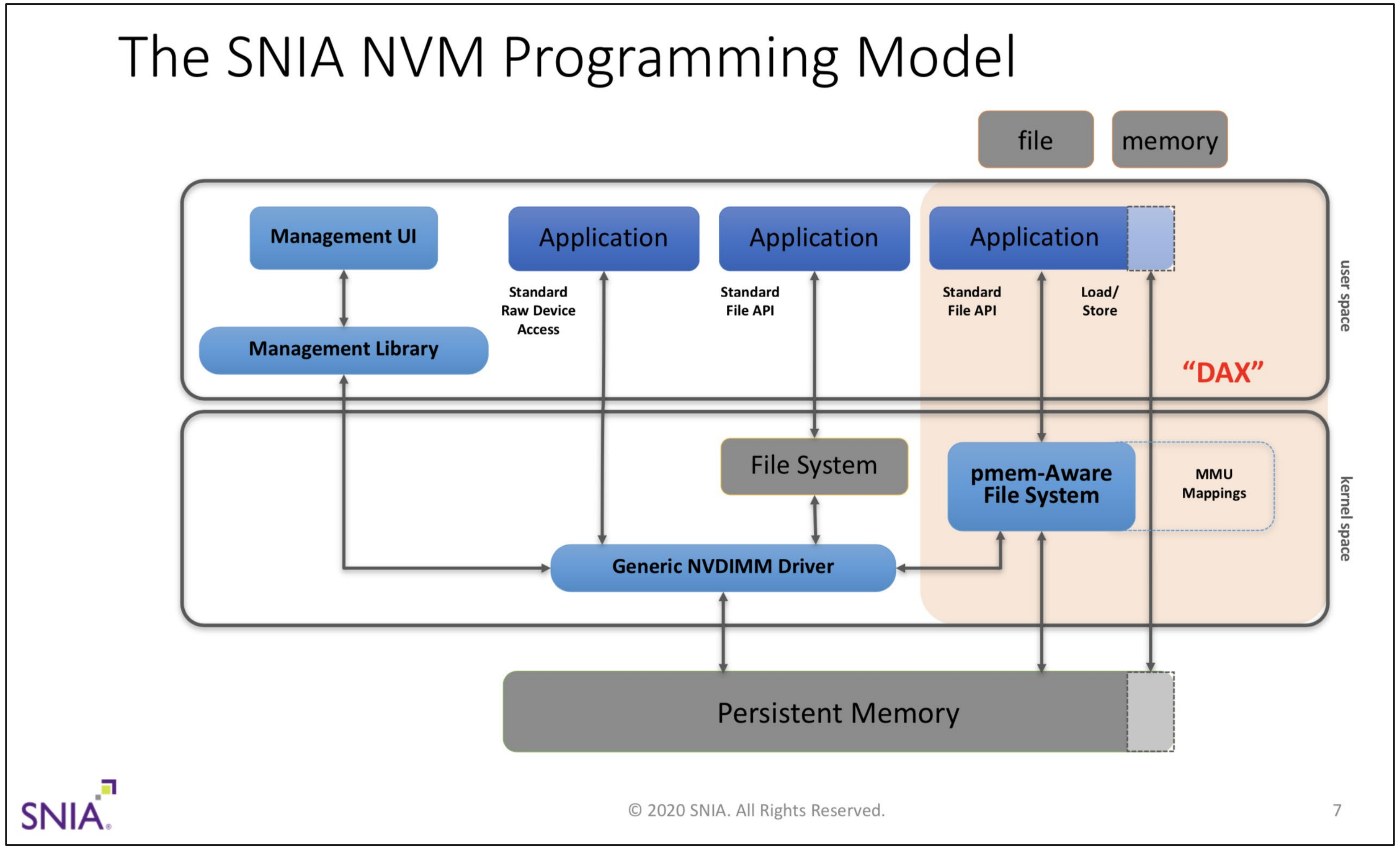
Connecting the Memory Bus

Intel's Approach for Optane PMem

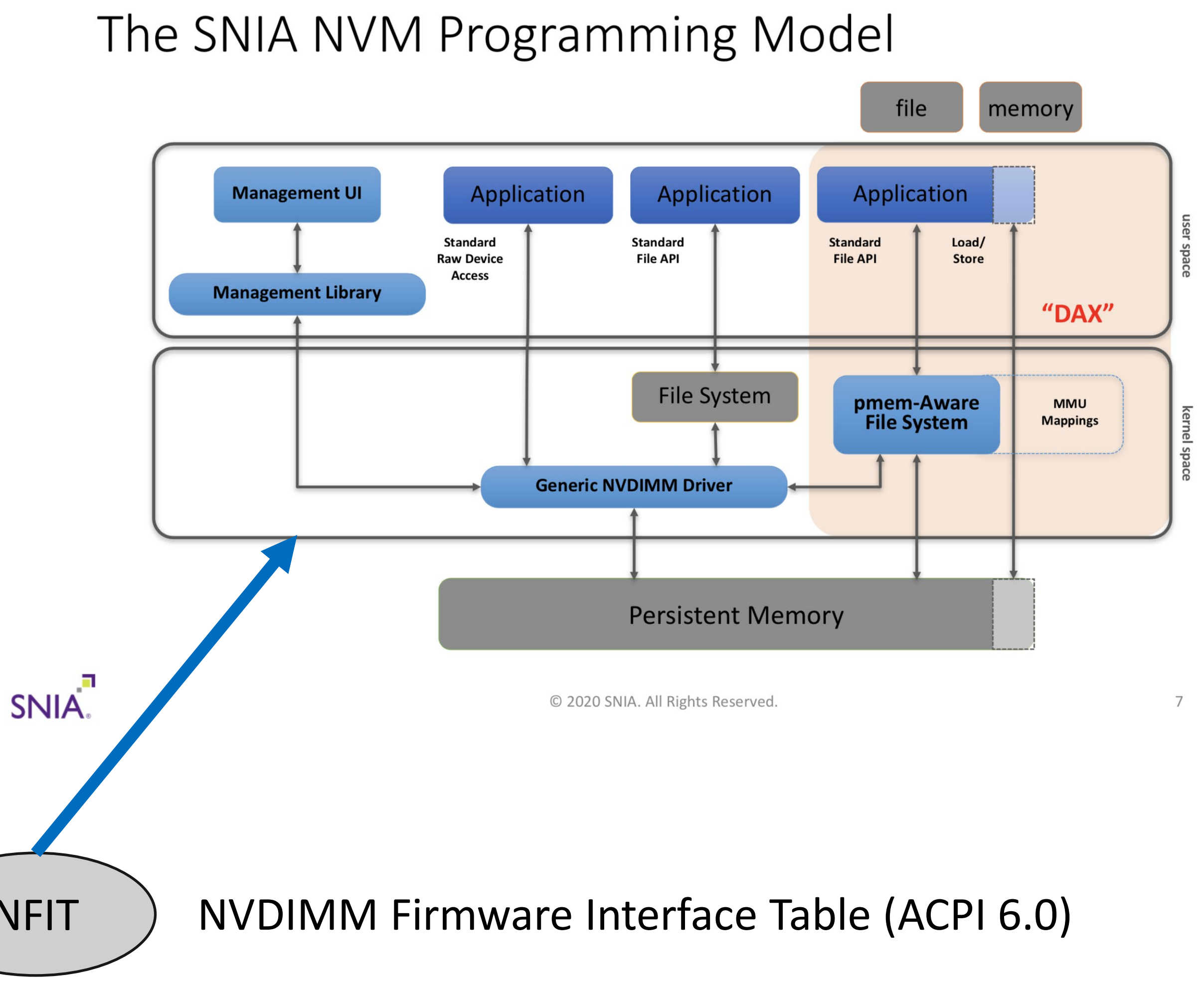




The ACPI "NVDIMM" Framework

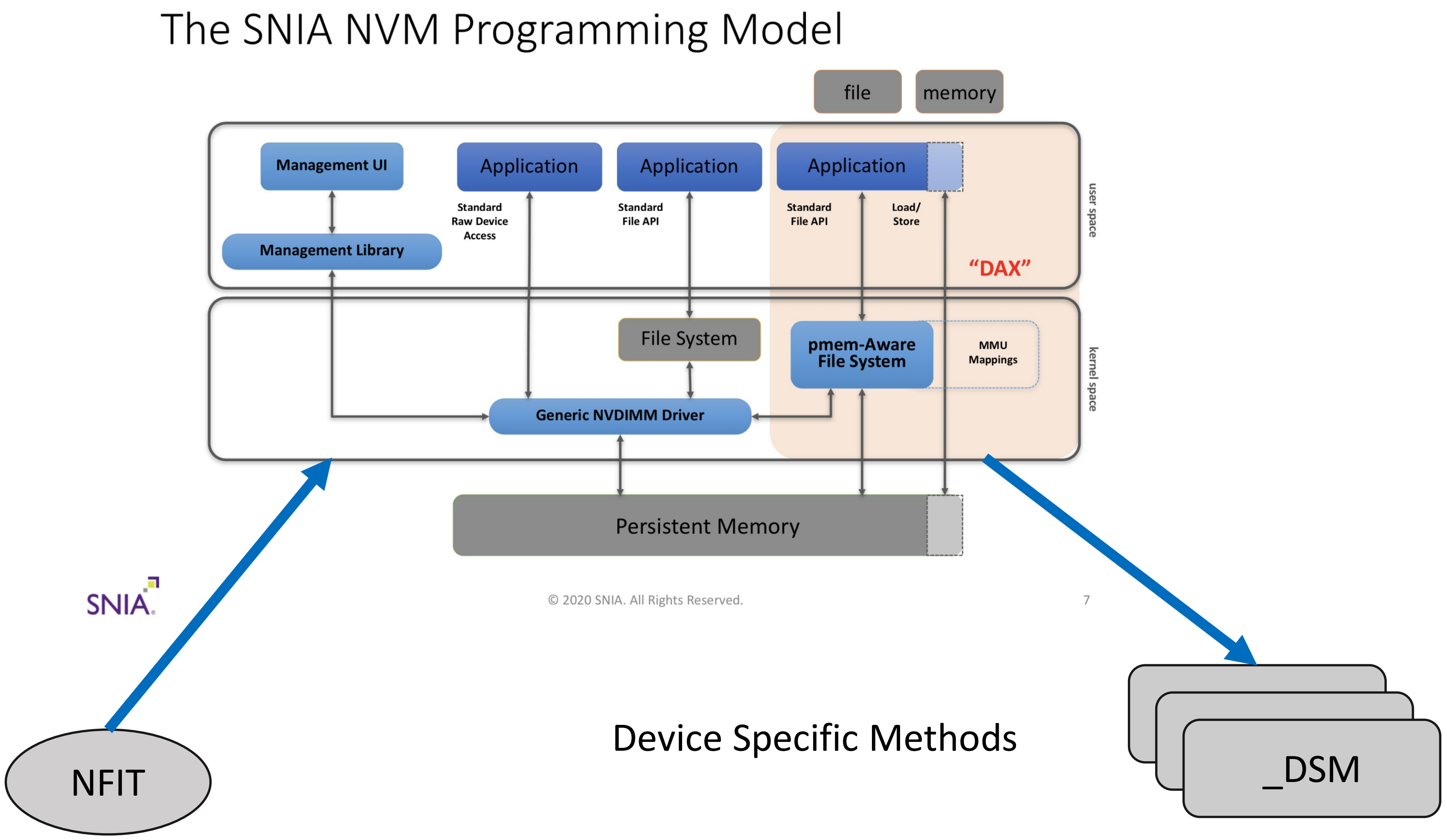


The ACPI “NVDIMM” Framework

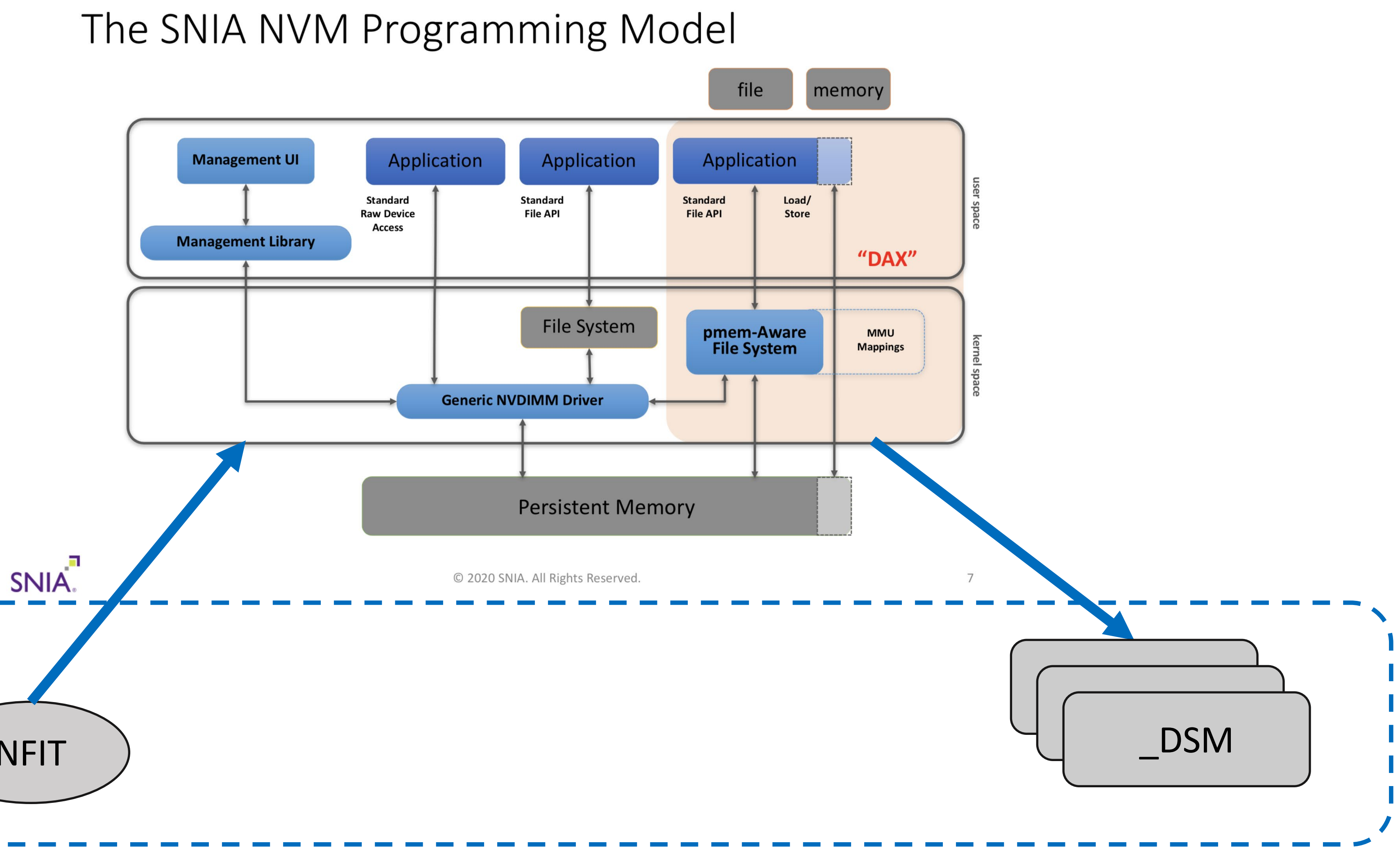


NVDIMM Firmware Interface Table (ACPI 6.0)

The ACPI “NVDIMM” Framework



The ACPI “NVDIMM” Framework



Learnings from ACPI Based Approach

Pros

- ACPI NFIT Unified NVDIMM-N and Intel's Optane PMem
  - Helped get enabling upstream early
- _DSMs allowed a generic kernel implementation
  - Differences abstracted away by _DSMs
- Mechanism has evolved gracefully
  - Fairly small additions, few errata

Cons

- ACPI dynamic support doesn't scale
  - Hot plug challenging
  - Meant for small number of empty sockets
- _DSM complexity hard to maintain
  - Bug fixes, additions logistical challenges
  - Virtually impossible to support multiple devices
  - No generic BIOS

Adding PMem to CXL

The CXL 2.0 Specification

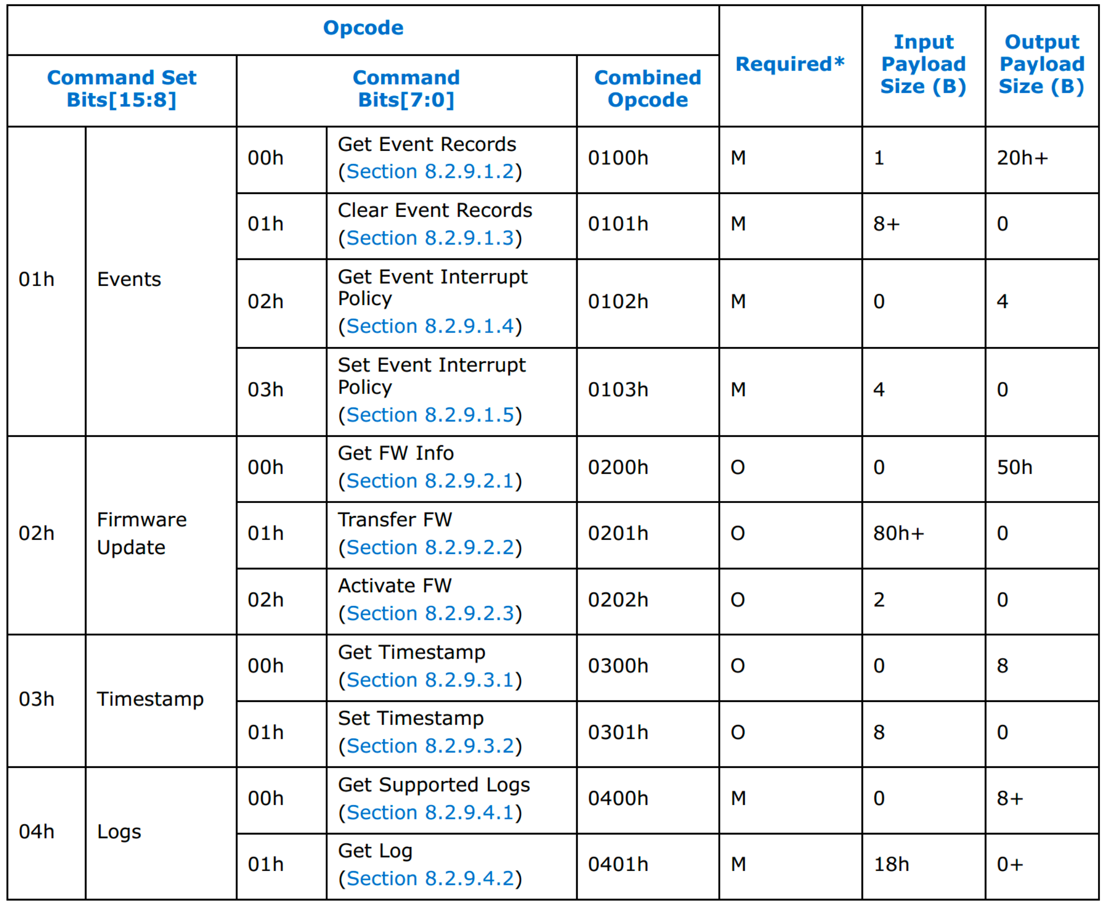
CXL 2.0 Changes for PMem

- Most changes should apply to all memory types
  - Minimize PMem-specific changes, rest apply to volatile memory too
- PCIe enumeration
  - NFIT isn't used for CXL devices (they aren't NVDIMMs!)
  - Leverage PCIe frameworks, including hot plug
- MMIO registers
  - Mailbox interface, etc.
- Command Interface
  - Was vendor-private for NVDIMMs
- SW Guide for Driver Writers
  - https://tinyurl.com/7eyje4pu
    - https://cdrdv2.intel.com/v1/dl/getContent/643805?wapkw=CXL%20memory%20device%20sw%20guide

Mailbox Commands

- Was vendor private
- Standards are a double-edged sword
  - Generic Drivers
  - Committee visit for every change
- Learnings from NVDIMMs helped
  - Leverage what worked
  - Fix pain points
- Some commands apply to all CXL →

CXL Device Command Opcodes

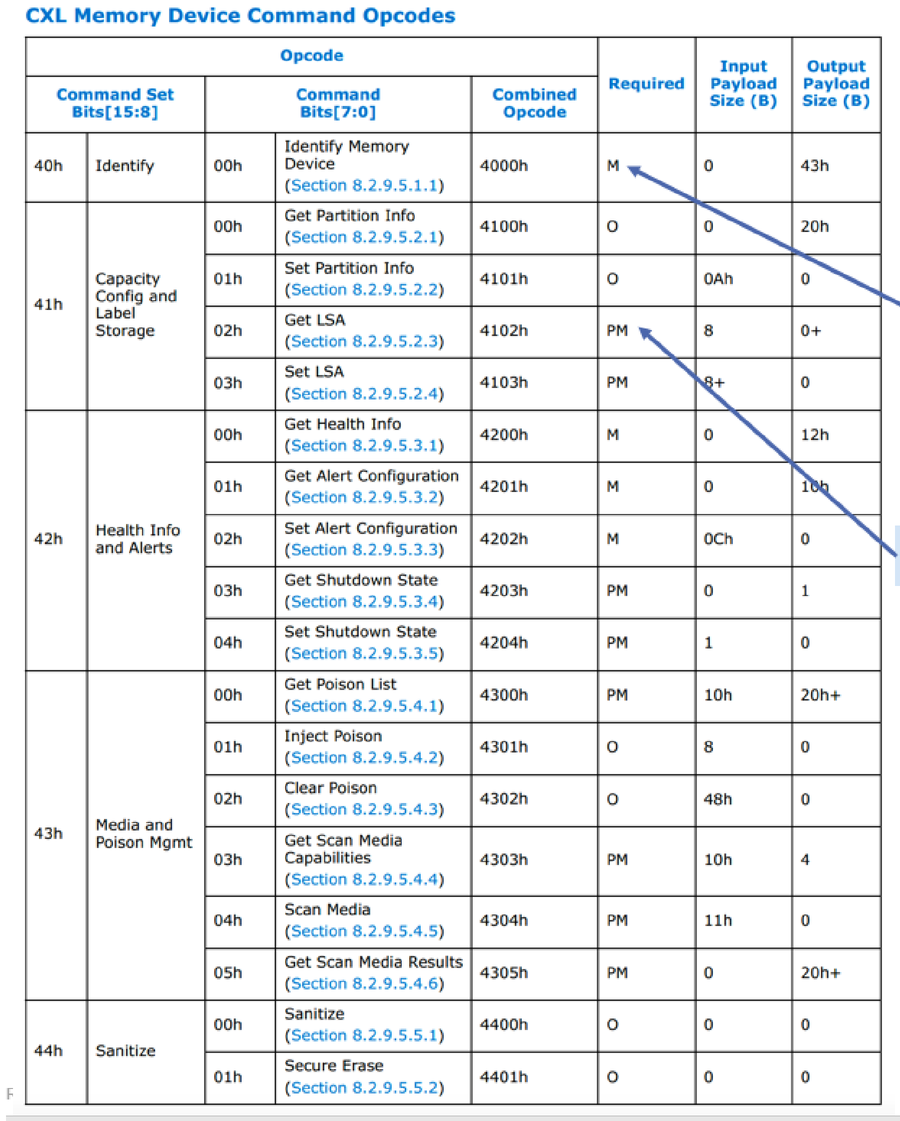


Opcode					Required*	Input Payload Size (B)	Output Payload Size (B)
Command Set Bits[15:8]		Command Bits[7:0]		Combined Opcode			
01h	Events	00h	Get Event Records (Section 8.2.9.1.2)	0100h	M	1	20h+
		01h	Clear Event Records (Section 8.2.9.1.3)	0101h	M	8+	0
		02h	Get Event Interrupt Policy (Section 8.2.9.1.4)	0102h	M	0	4
		03h	Set Event Interrupt Policy (Section 8.2.9.1.5)	0103h	M	4	0
02h	Firmware Update	00h	Get FW Info (Section 8.2.9.2.1)	0200h	O	0	50h
		01h	Transfer FW (Section 8.2.9.2.2)	0201h	O	80h+	0
		02h	Activate FW (Section 8.2.9.2.3)	0202h	O	2	0
03h	Timestamp	00h	Get Timestamp (Section 8.2.9.3.1)	0300h	O	0	8
		01h	Set Timestamp (Section 8.2.9.3.2)	0301h	O	8	0
04h	Logs	00h	Get Supported Logs (Section 8.2.9.4.1)	0400h	M	0	8+
		01h	Get Log (Section 8.2.9.4.2)	0401h	M	18h	0+

STORAGE DEVELOPER CONFERENCE SDC21

Memory Device Commands

- Most added commands →
- _DSMs are gone
  - OS uses mailbox directly
  - BIOS too
- Much complexity moved:
  - From BIOS to OS
- Allows generic:
  - BIOS
  - OS Drivers



CXL Memory Device Command Opcodes

Opcode					Required	Input Payload Size (B)	Output Payload Size (B)
Command Set Bits[15:8]		Command Bits[7:0]		Combined Opcode			
40h	Identify	00h	Identify Memory Device (Section 8.2.9.5.1.1)	4000h	M	0	43h
41h	Capacity Config and Label Storage	00h	Get Partition Info (Section 8.2.9.5.2.1)	4100h	O	0	20h
		01h	Set Partition Info (Section 8.2.9.5.2.2)	4101h	O	0Ah	0
		02h	Get LSA (Section 8.2.9.5.2.3)	4102h	PM	8	0+
		03h	Set LSA (Section 8.2.9.5.2.4)	4103h	PM	8+	0
42h	Health Info and Alerts	00h	Get Health Info (Section 8.2.9.5.3.1)	4200h	M	0	12h
		01h	Get Alert Configuration (Section 8.2.9.5.3.2)	4201h	M	0	10h
		02h	Set Alert Configuration (Section 8.2.9.5.3.3)	4202h	M	0Ch	0
		03h	Get Shutdown State (Section 8.2.9.5.3.4)	4203h	PM	0	1
		04h	Set Shutdown State (Section 8.2.9.5.3.5)	4204h	PM	1	0
43h	Media and Poison Mgmt	00h	Get Poison List (Section 8.2.9.5.4.1)	4300h	PM	10h	20h+
		01h	Inject Poison (Section 8.2.9.5.4.2)	4301h	O	8	0
		02h	Clear Poison (Section 8.2.9.5.4.3)	4302h	O	48h	0
		03h	Get Scan Media Capabilities (Section 8.2.9.5.4.4)	4303h	PM	10h	4
		04h	Scan Media (Section 8.2.9.5.4.5)	4304h	PM	11h	0
		05h	Get Scan Media Results (Section 8.2.9.5.4.6)	4305h	PM	0	20h+
44h	Sanitize	00h	Sanitize (Section 8.2.9.5.5.1)	4400h	O	0	0
		01h	Secure Erase (Section 8.2.9.5.5.2)	4401h	O	0	0

Mandatory

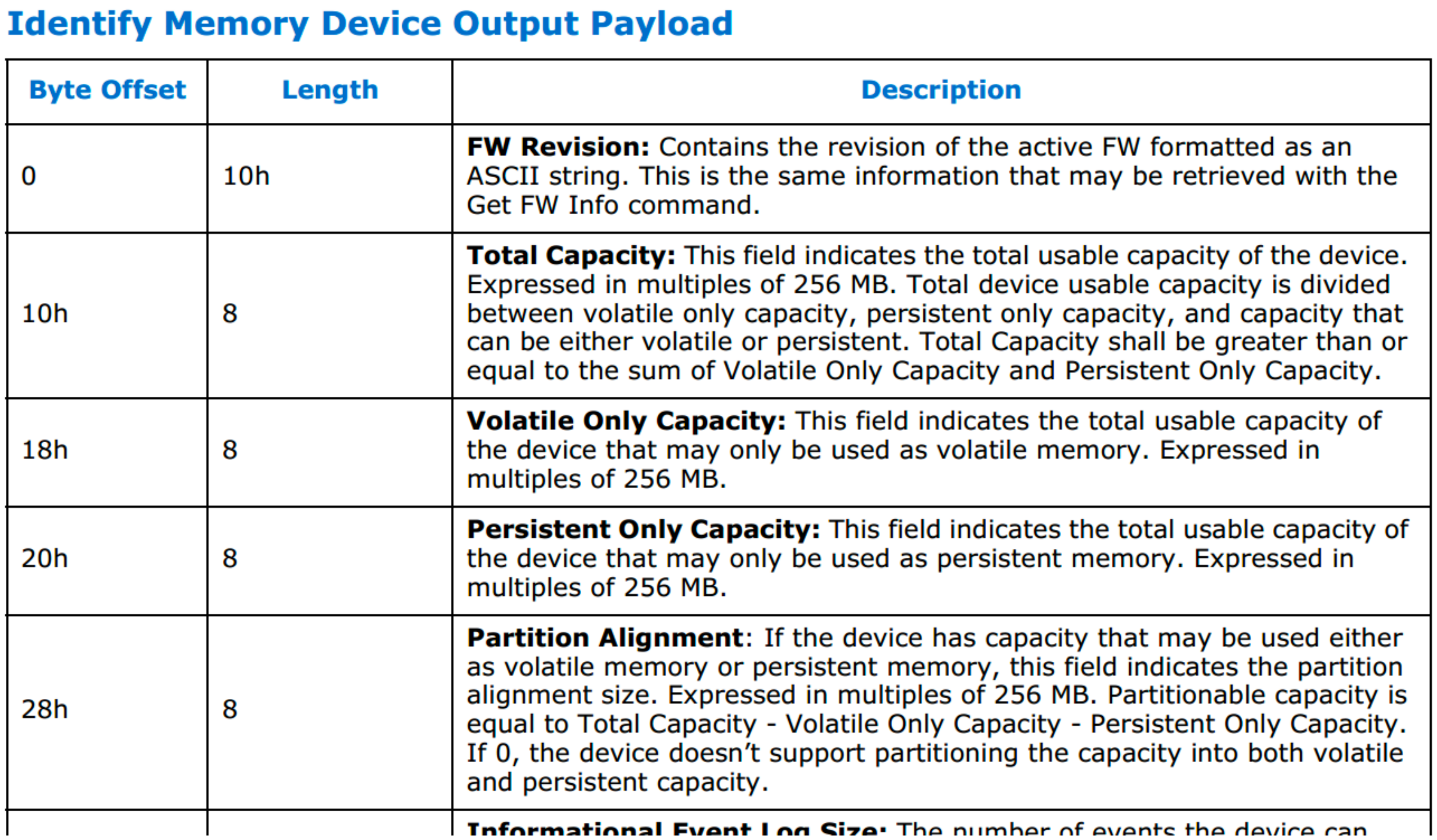
Mandatory for PMem

…not the full table

16

STORAGE DEVELOPER CONFERENCE SDC 21

Example: Identify Memory Device



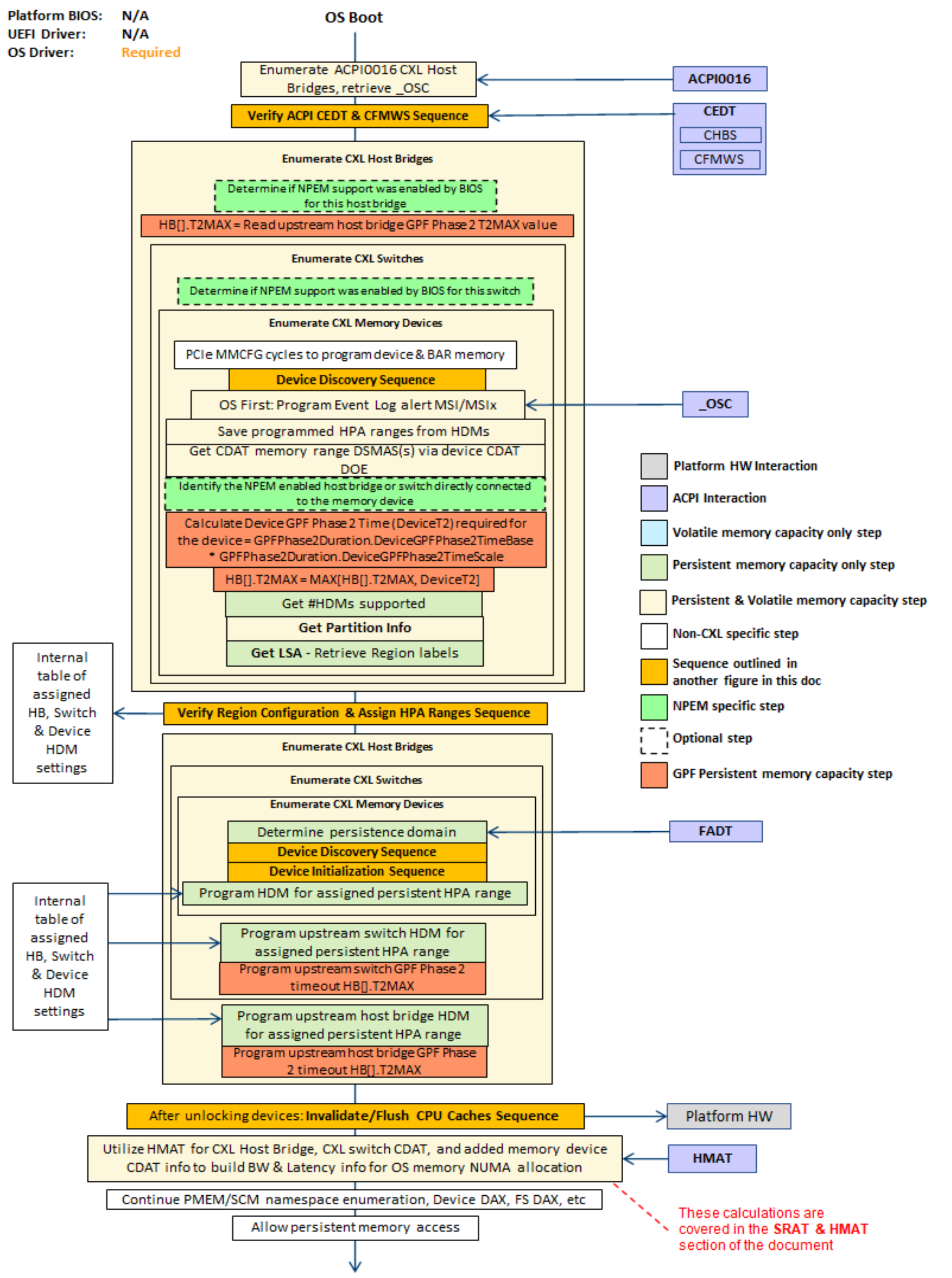
Identify Memory Device Output Payload

Byte Offset	Length	Description
0	10h	**FW Revision:** Contains the revision of the active FW formatted as an ASCII string. This is the same information that may be retrieved with the Get FW Info command.
10h	8	**Total Capacity:** This field indicates the total usable capacity of the device. Expressed in multiples of 256 MB. Total device usable capacity is divided between volatile only capacity, persistent only capacity, and capacity that can be either volatile or persistent. Total Capacity shall be greater than or equal to the sum of Volatile Only Capacity and Persistent Only Capacity.
18h	8	**Volatile Only Capacity:** This field indicates the total usable capacity of the device that may only be used as volatile memory. Expressed in multiples of 256 MB.
20h	8	**Persistent Only Capacity:** This field indicates the total usable capacity of the device that may only be used as persistent memory. Expressed in multiples of 256 MB.
28h	8	**Partition Alignment**: If the device has capacity that may be used either as volatile memory or persistent memory, this field indicates the partition alignment size. Expressed in multiples of 256 MB. Partitionable capacity is equal to Total Capacity - Volatile Only Capacity - Persistent Only Capacity. If 0, the device doesn’t support partitioning the capacity into both volatile and persistent capacity.
		Informational Event Log Size: The number of events the device can

STORAGE DEVELOPER CONFERENCE SDC 21

Memory Device SW Guide

- CXL 2.0 Spec defines commands
  - Can be terse, spec language
- Platforms decide some of the details
  - Example: BIOS on Intel platforms
- Document flows, algorithms

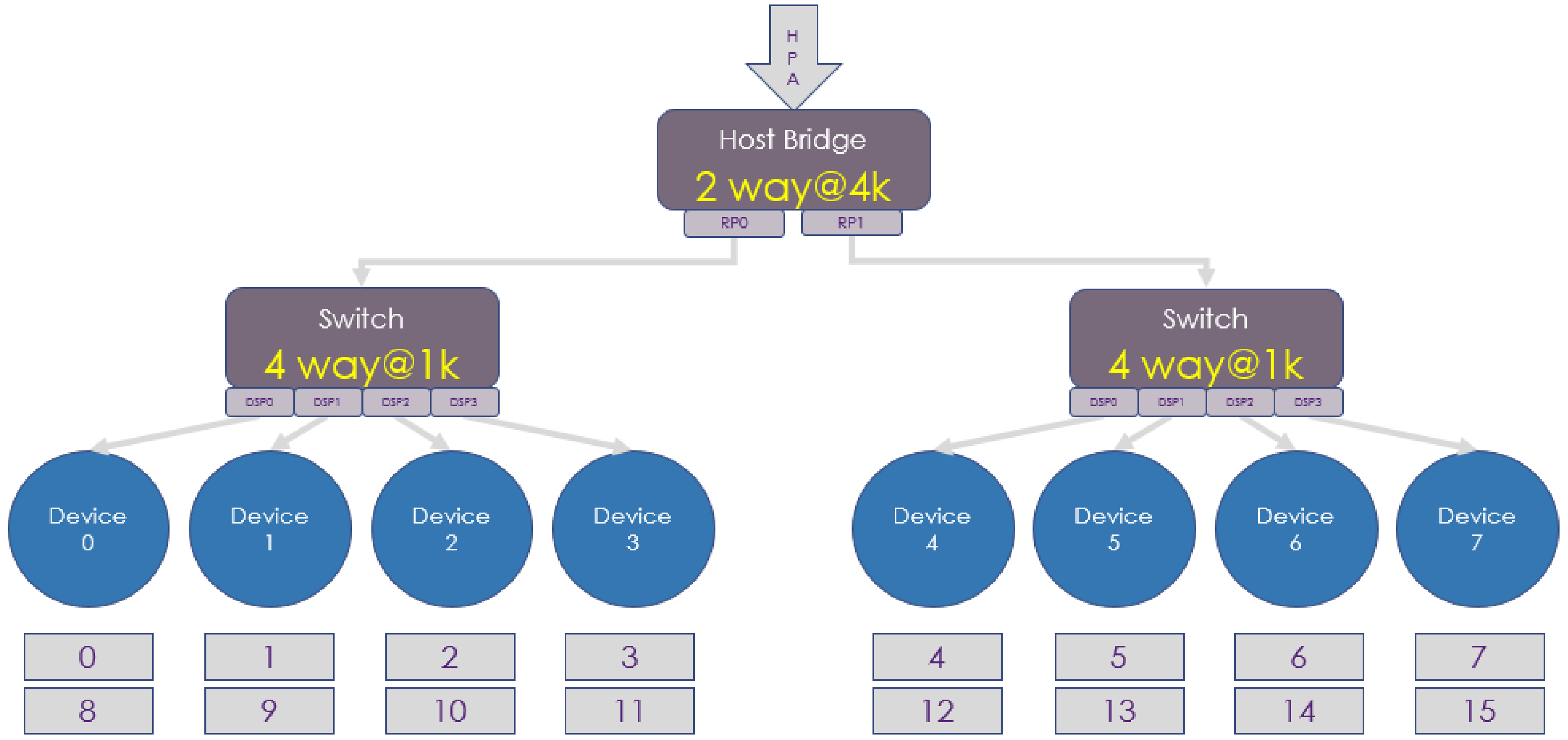


Platform BIOS: N/A
UEFI Driver: N/A
OS Driver: Required
OS Boot
Enumerate ACPI0016 CXL Host Bridges, retrieve _OSC
ACPI0016
Verify ACPI CEDT & CFMWS Sequence
CEDT
CHBS
CFMWS
Enumerate CXL Host Bridges
Determine if NPEM support was enabled by BIOS for this host bridge
HB[].T2MAX = Read upstream host bridge GPF Phase 2 T2MAX value
Enumerate CXL Switches
Determine if NPEM support was enabled by BIOS for this switch
Enumerate CXL Memory Devices
PCIe MMCFG cycles to program device & BAR memory
Device Discovery Sequence
OS First: Program Event Log alert MSI/MSIx
_OSC
Save programmed HPA ranges from HDMs
Get CDAT memory range DSMAS(s) via device CDAT DOE
Identify the NPEM enabled host bridge or switch directly connected to the memory device
Calculate Device GPF Phase 2 Time (DeviceT2) required for the device = GPFPhase2Duration.DeviceGPFPhase2TimeBase * GPFPhase2Duration.DeviceGPFPhase2TimeScale
HB[].T2MAX = MAX[HB[].T2MAX, DeviceT2]
Get #HDMs supported
Get Partition Info
Get LSA - Retrieve Region labels
Internal table of assigned HB, Switch & Device HDM settings
Verify Region Configuration & Assign HPA Ranges Sequence
Enumerate CXL Host Bridges
Enumerate CXL Switches
Enumerate CXL Memory Devices
Determine persistence domain
FADT
Device Discovery Sequence
Device Initialization Sequence
Program HDM for assigned persistent HPA range
Internal table of assigned HB, Switch & Device HDM settings
Program upstream switch HDM for assigned persistent HPA range
Program upstream switch GPF Phase 2 timeout HB[].T2MAX
Program upstream host bridge HDM for assigned persistent HPA range
Program upstream host bridge GPF Phase 2 timeout HB[].T2MAX
After unlocking devices: Invalidate/Flush CPU Caches Sequence
Platform HW
Utilize HMAT for CXL Host Bridge, CXL switch CDAT, and added memory device CDAT info to build BW & Latency info for OS memory NUMA allocation
HMAT
Continue PMEM/SCM namespace enumeration, Device DAX, FS DAX, etc
Allow persistent memory access
These calculations are covered in the SRAT & HMAT section of the document
Platform HW Interaction
ACPI Interaction
Volatile memory capacity only step
Persistent memory capacity only step
Persistent & Volatile memory capacity step
Non-CXL specific step
Sequence outlined in another figure in this doc
NPEM specific step
Optional step
GPF Persistent memory capacity step

Interleaving

- HDM Decoders
  - Allow interleaving across devices
  - New to PCIe: interleave sets
- Important concept for PMem
  - For volatile memory, changing the interleave may impact performance
  - For PMem, changing the interleave loses your data
- Label Storage Area
  - Defined in CXL 2.0 spec
  - Provides region (interleave set) and namespace configuration

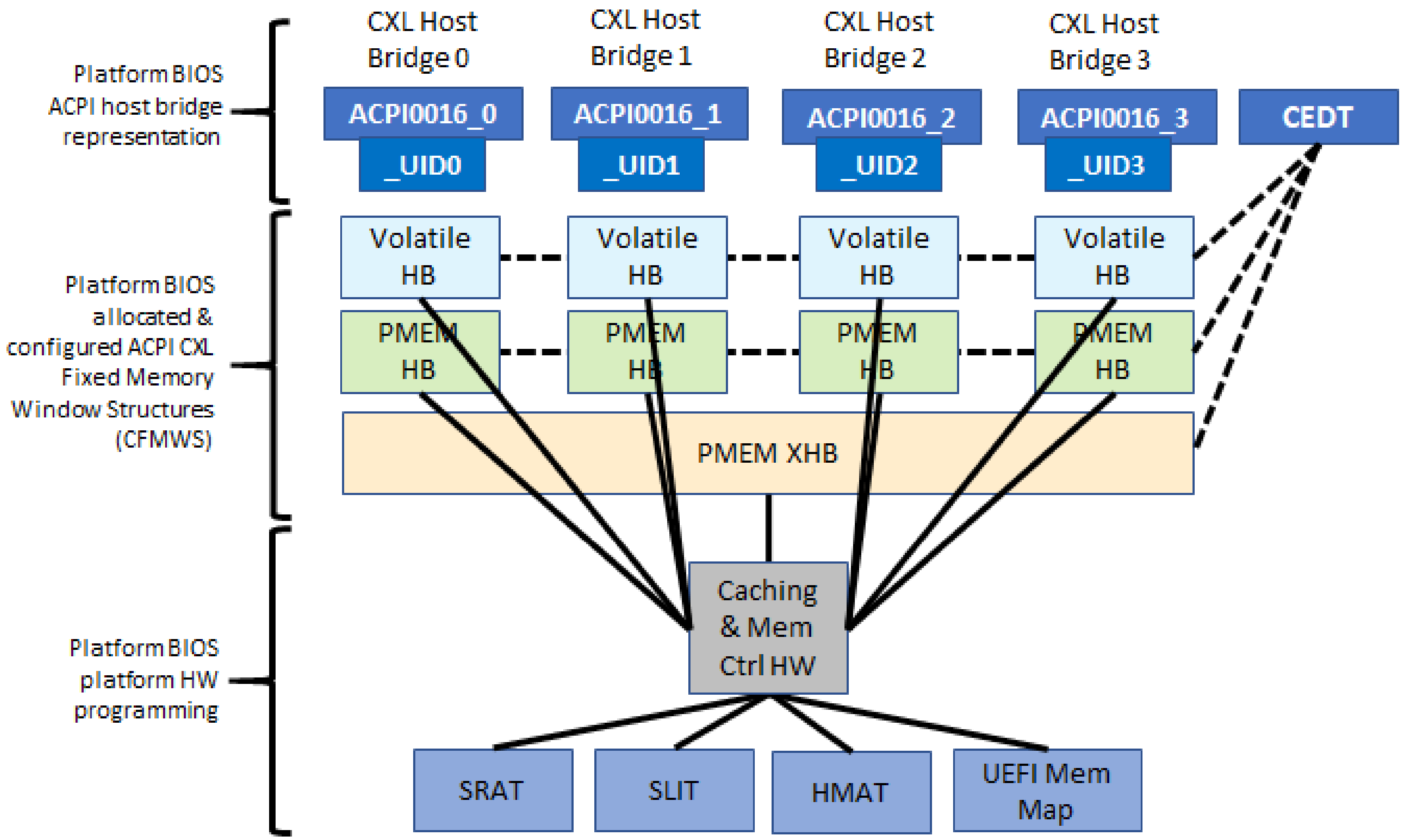
Label Example



Label		Label		Label		Label		Label		Label		Label		Label	
N	8	N	8	N	8	N	8	N	8	N	8	N	8	N	8
Pos	0	Pos	1	Pos	2	Pos	3	Pos	4	Pos	5	Pos	6	Pos	7
IG	1k	IG	1k	IG	1k	IG	1k	IG	1k	IG	1k	IG	1k	IG	1k

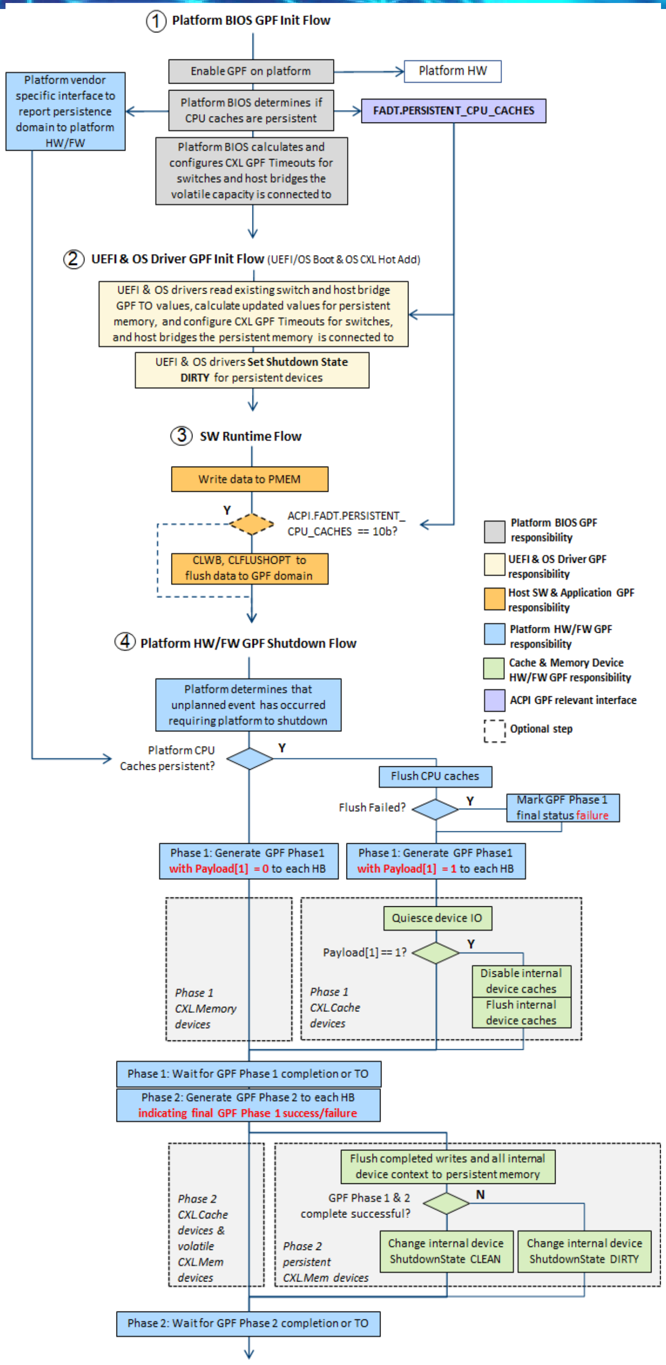
Hot Plug

- Handled by OS
  - No BIOS in flow
- Uses “windows”
  - BIOS provided
- Flow used for PMem
  - Except boot dev



Flush-on-fail To Persistence

- Global Persistent Flush (GPF)
  - Analogous to ADR or eADR with NVDIMMs
- Simple when it works
  - Application just relies on it
- More complex when it fails
  - Dirty Shutdown Count
  - Already part of the programming model
  - Code written for NVDIMM still works correctly
    - Provided the model was followed correctly



1 Platform BIOS GPF Init Flow
Enable GPF on platform
Platform HW
Platform vendor specific interface to report persistence domain to platform HW/FW
Platform BIOS determines if CPU caches are persistent
FADT.PERSISTENT_CPU_CACHES
Platform BIOS calculates and configures CXL GPF Timeouts for switches and host bridges the volatile capacity is connected to
2 UEFI & OS Driver GPF Init Flow (UEFI/OS Boot & OS CXL Hot Add)
UEFI & OS drivers read existing switch and host bridge GPF TO values, calculate updated values for persistent memory, and configure CXL GPF Timeouts for switches, and host bridges the persistent memory is connected to
UEFI & OS drivers Set Shutdown State DIRTY for persistent devices
3 SW Runtime Flow
Write data to PMEM
Y
ACPI.FADT.PERSISTENT_CPU_CACHES == 10b?
CLWB, CLFLUSHOPT to flush data to GPF domain
4 Platform HW/FW GPF Shutdown Flow
Platform determines that unplanned event has occurred requiring platform to shutdown
Platform CPU Caches persistent?
Y
Flush CPU caches
Flush Failed?
Y
Mark GPF Phase 1 final status failure
Phase 1: Generate GPF Phase1 with Payload[1] = 0 to each HB
Phase 1: Generate GPF Phase1 with Payload[1] = 1 to each HB
Quiesce device IO
Payload[1] == 1?
Y
Disable internal device caches
Flush internal device caches
Phase 1 CXL Memory devices
Phase 1 CXL Cache devices
Phase 1: Wait for GPF Phase 1 completion or TO
Phase 2: Generate GPF Phase 2 to each HB indicating final GPF Phase 1 success/failure
Flush completed writes and all internal device context to persistent memory
GPF Phase 1 & 2 complete successful?
N
Change internal device ShutdownState CLEAN
Change internal device ShutdownState DIRTY
Phase 2 CXL Cache devices & volatile CXL Mem devices
Phase 2 persistent CXL Mem devices
Phase 2: Wait for GPF Phase 2 completion or TO
Platform BIOS GPF responsibility
UEFI & OS Driver GPF responsibility
Host SW & Application GPF responsibility
Platform HW/FW GPF responsibility
Cache & Memory Device HW/FW GPF responsibility
ACPI GPF relevant interface
Optional step

STORAGE DEVELOPER CONFERENCE SDC 21

SW Enabling

- Preliminary generic Type 3 CXL Driver already upstream in Linux
  - Orchestrated by NVDIMM framework maintainer Dan Williams
- QEMU patches emulating CXL Type 3 devices posted
  - Written by Ben Widawsky, provided a development platform for the driver
- Cross-company, cross committee collaboration on specifications
  - Prevents messy collisions from different implementation decisions

Summary

- The programming model remains the same
  - Applications written to the SNIA programming model continue to work
- CXL offers:
  - Moving PMem off the memory bus
  - Scalability (all types of memory)
  - Flexibility
- PMem on CXL specified as of CXL 2.0, published last November
  - OS enabling is emerging

Please take a moment to rate this session.

Your feedback is important to us.

STORAGE DEVELOPER CONFERENCE SDC 21