SNIA DEVELOPER CONFERENCE

SDC 24

BY Developers FOR Developers

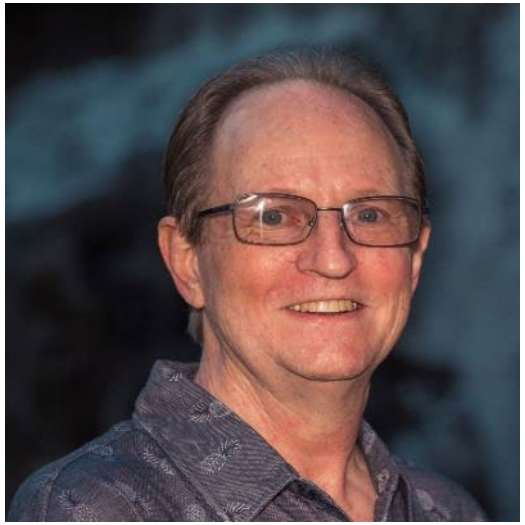
September 16-18, 2024
Santa Clara, CA

Host Addressable SLM

NVMe and CXL Collaborating

Bill Martin and Jason Molgaard

Speakers



Bill Martin

SAMSUNG

Jason Molgaard

SOLIDIGM

SDC 24

Agenda

- NVMe® TP4184 Host Addressable SLM
- Computational Storage: A Use for Host Addressable SLM
- Combining NVMe® and CXL® Technologies
- Use Cases

NVMe® TP4184
Host Addressable SLM

Why Provide Host Addressable SLM?

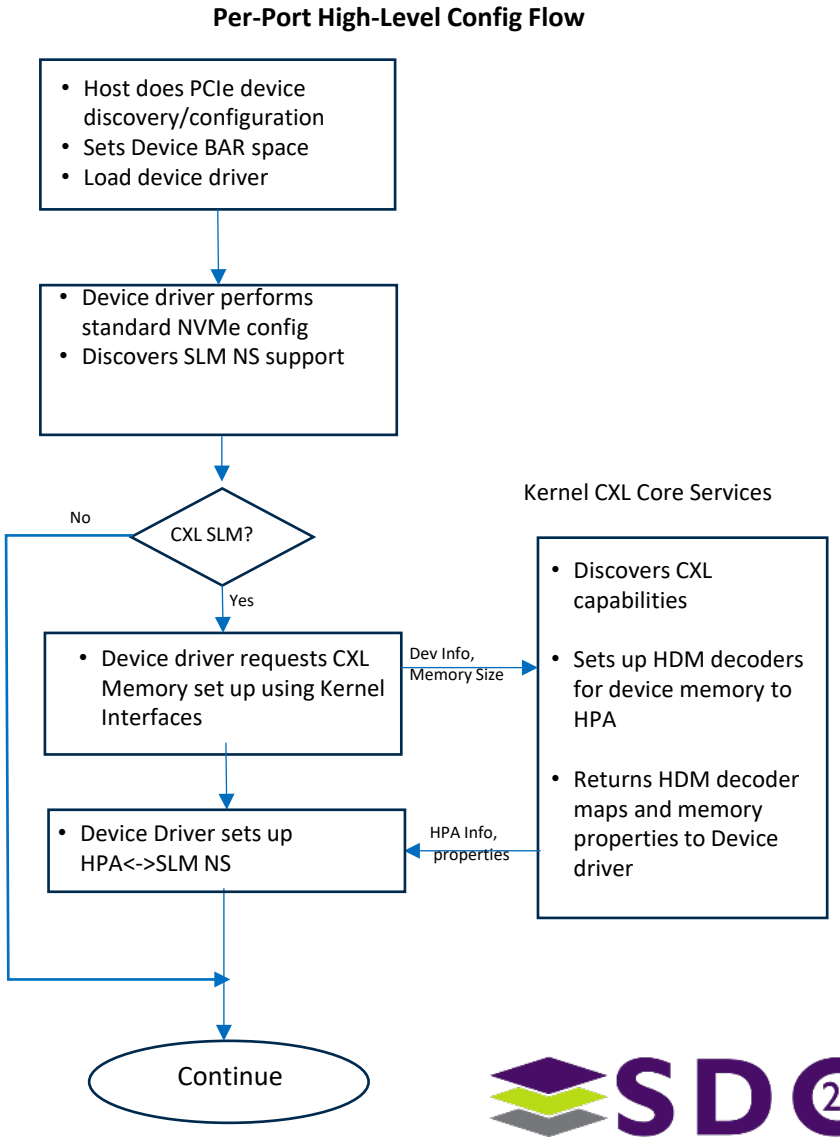
- NVMe devices currently provide host accessible memory in the form of SLM
  - Accessing this memory via a memory protocol will
    - Be more efficient
    - Allow cache coherency of that memory
    - Allow peer-to-peer communication using a memory model
    - Eliminate a context switch
- Computational Storage Use
  - Computational Storage Drives have more host accessible memory than a traditional Storage Device
  - Benefits from peer-to-peer communication (more on this later)
- Why not CMB/PMR?
  - No mechanism to specify a location in CMB/PMR to the device

Host Addressable SLM – NVMe® TP4184

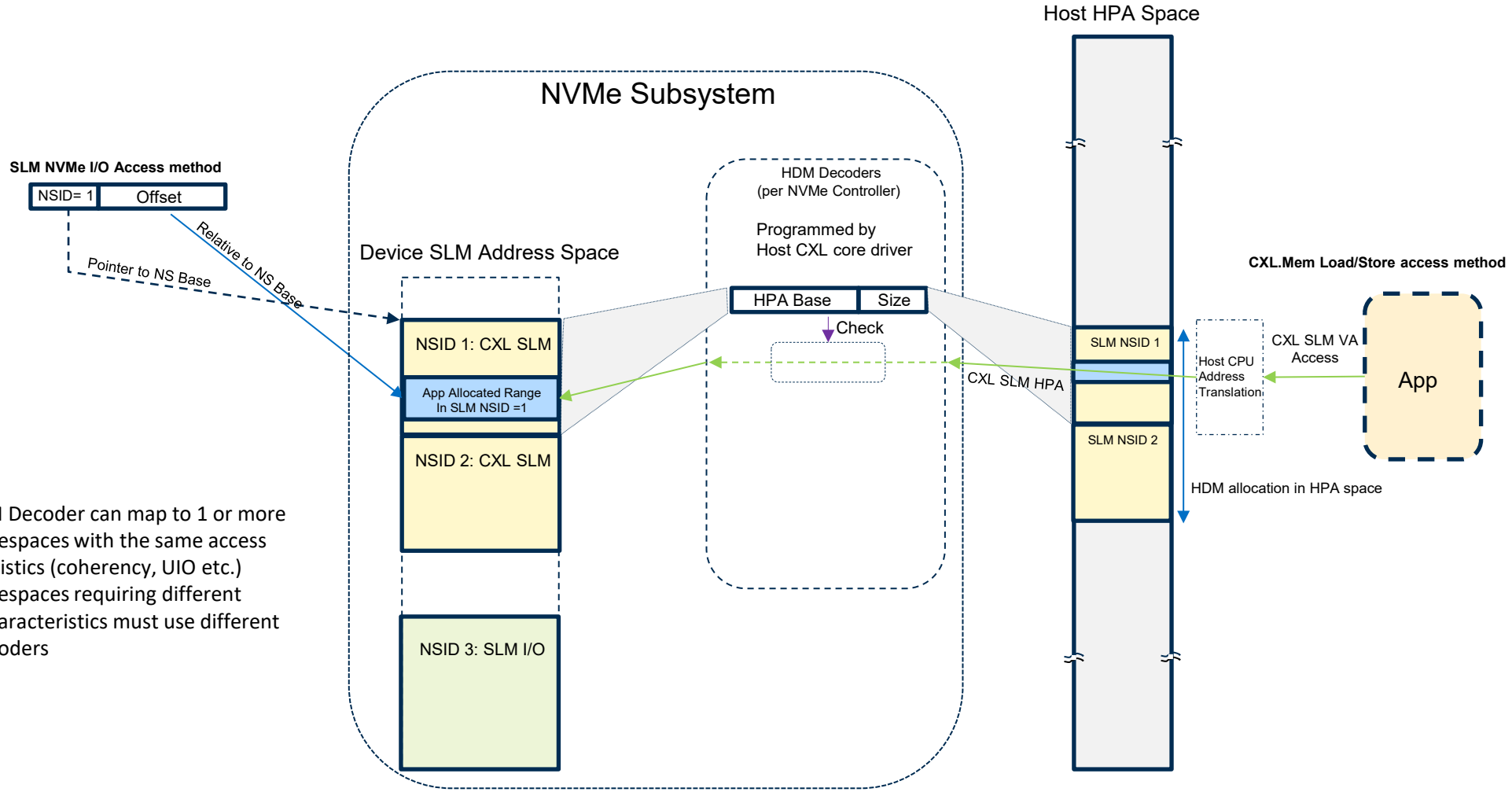
- NVMe TP4184 is in the Architecture Definition phase
- SLM is addressed at a Host Physical Address (HPA)
  - PCIe BAR; or
  - CXL HDM
- SLM memory can have a virtual mapping for host applications
  - Host application does not have to switch contexts for SLM memory access
- SLM memory is accessible by the host and the device
- SLM can be read/written with:
  - Host Load/Store commands
  - CXL.mem commands
- Compute is triggered with Computational Programs commands
  - Utilizes the host addressable SLM
- Allows P2P data movement based upon HPA
- SLM is still accessible using the SLM Command Set Memory Read and Memory Write commands

Potential Config Flow for CXL® SLM

- **NVMe device with CXL SLM is similar to a CXL Type 2 device**
- **Plan for a CXL Type 2 device is for the OS to do standard PCIe configuration (e.g. allocating BAR space) and then load driver CXL configuration using device's PCI ID**
- **NVMe device with CXL SLM will use an enhanced NVMe driver**
  - Enhanced for CXL based SLM configuration
  - Device will use existing NVMe Class Code and may use a new Programming Interface (PI) identifier to expose CXL capabilities
- **Device driver discovers device capabilities and calls OS CXL core services for CXL Memory set up**
  - CXL core services offers kernel interfaces for the driver to set up required CXL capabilities such as HDM decoders and return necessary information (e.g. HPA range)
    - Device CXL memory allocation is controlled by the NVMe driver
    - Linux support for CXL Type 2 devices is not yet available
  - Driver owns runtime management of Device CXL memory



Host – CXL® SLM Address Mapping

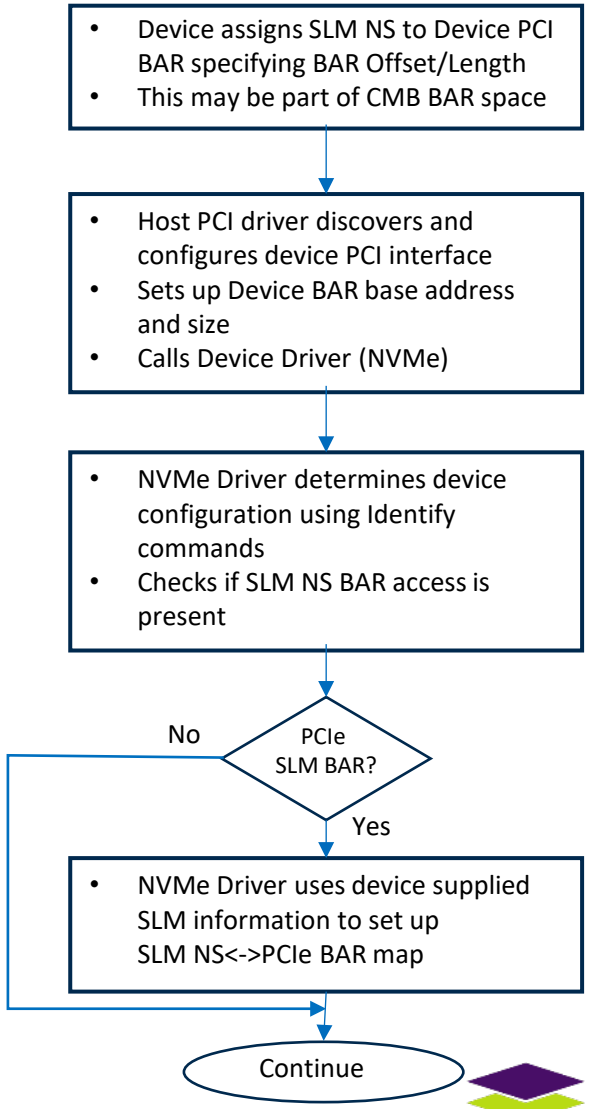


Notes:

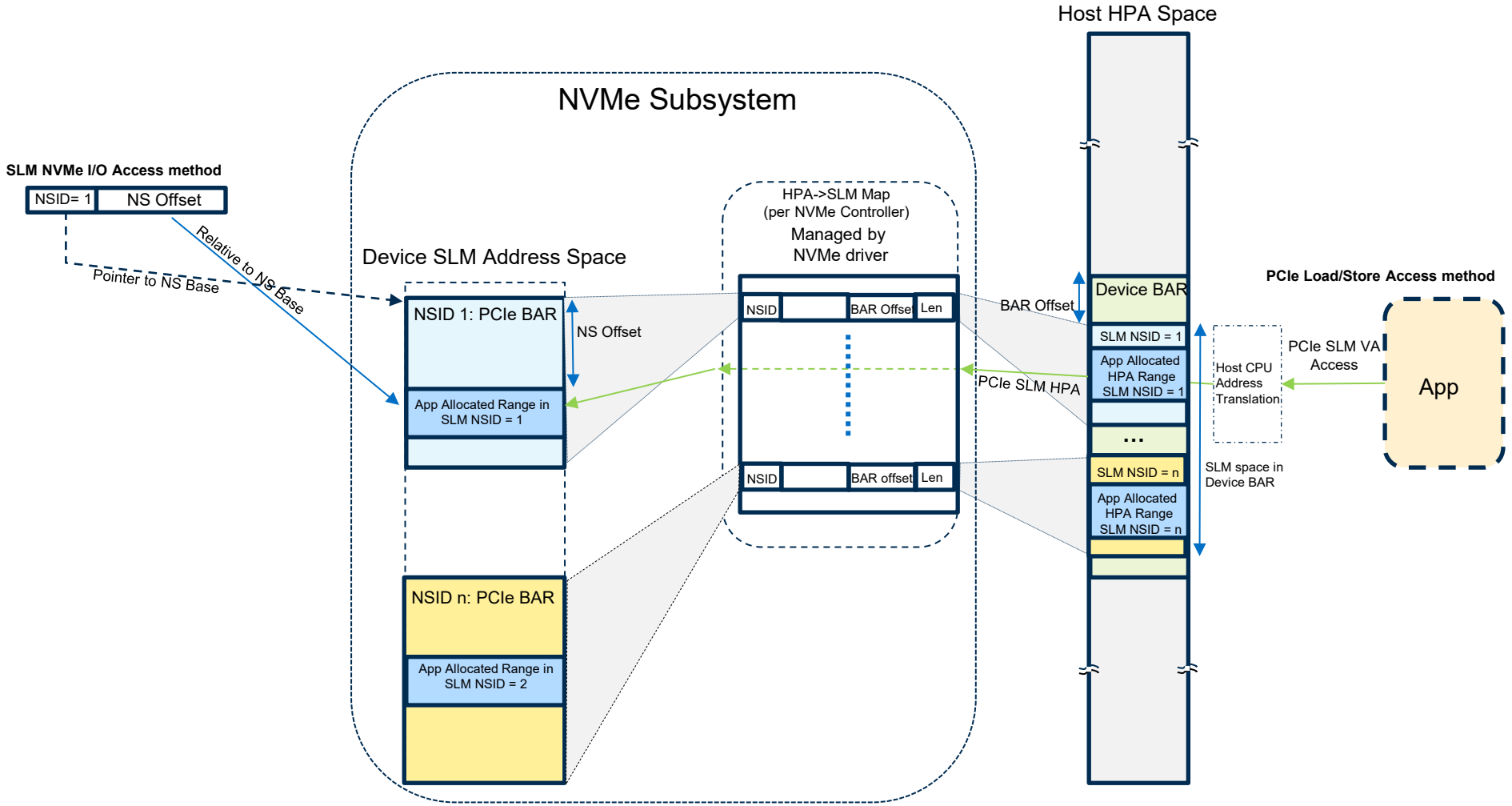
- One HDM Decoder can map to 1 or more SLM namespaces with the same access characteristics (coherency, UIO etc.)
- SLM namespaces requiring different access characteristics must use different HDM decoders

Configuration of PCIe BAR to SLM

- Device statically assigns one or more SLM Namespaces for PCIe BAR access
- Device advertises BAR space needs which includes SLM usage
  - BAR space for SLM NS access may be smaller than SLM NS size
- Host (PCI Driver) allocates Device BAR space as a normal part of device discovery/config
- Host (NVMe Driver) manages available BAR space for SLM access by determining
  - NSID, SLM NS Offset, PCIe BAR offset, Length
  - May be a contiguous range within CMB BAR
- Device maintains a mapping to translate PCIe Address to SLM (example shown on next slide)
- Application uses BAR range provided by driver to access SLM directly using PCIe read/write operations



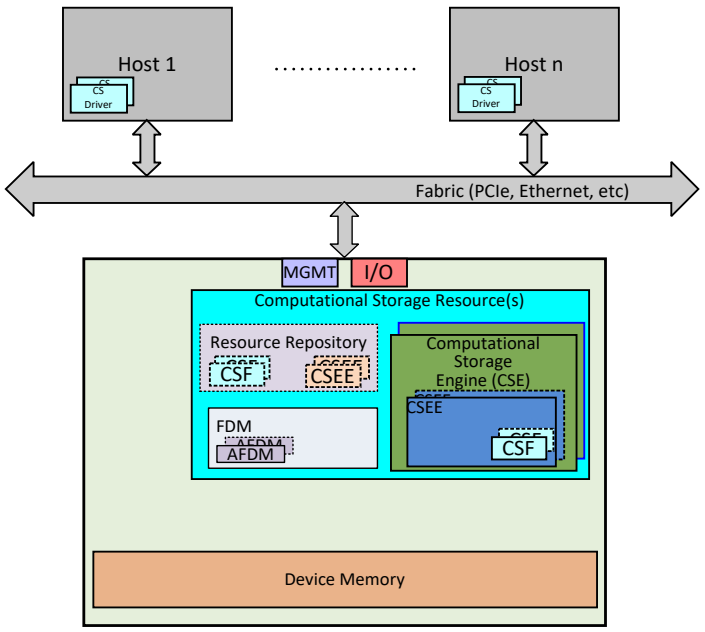
Host – PCI BAR SLM Address Mapping



Computational Storage:
A Use for Host Addressable SLM

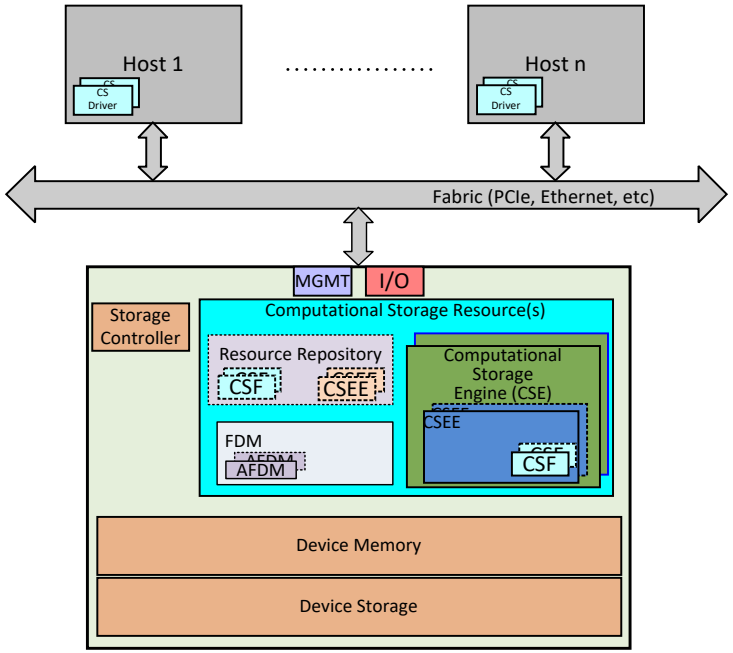
Computational Storage Architecture

Computational Storage Processor



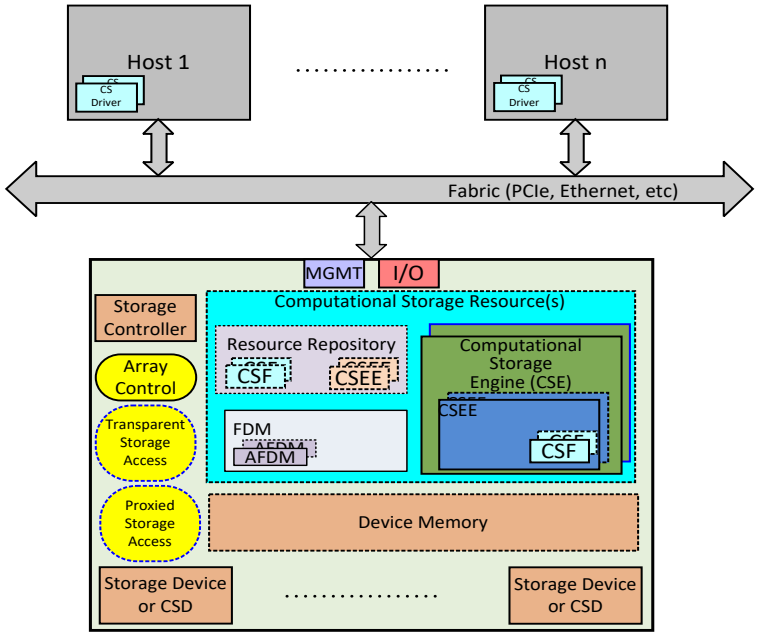
Computational Storage Processor (CSP)

Computational Storage Drive



Computational Storage Drive (CSD)

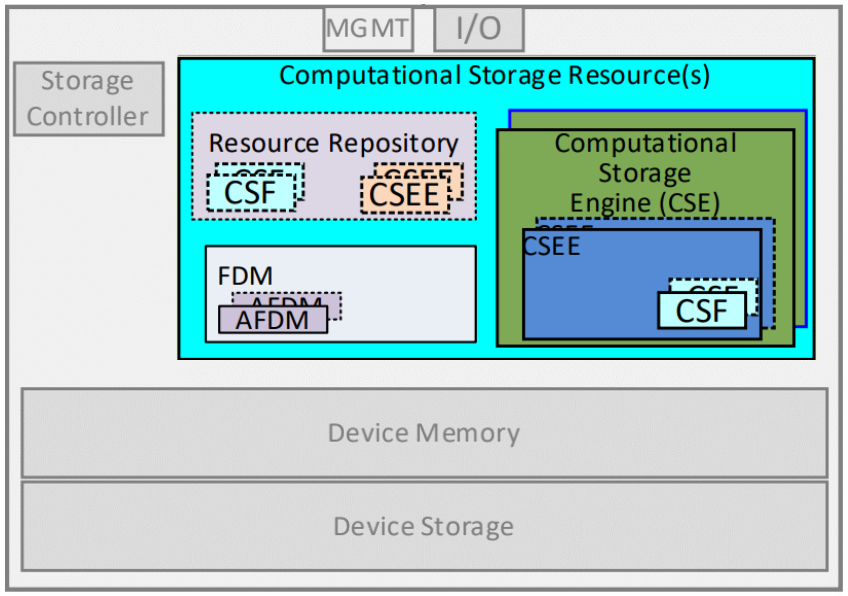
Computational Storage Array



Computational Storage Array (CSA)

CSx = Computational Storage **Device** – CSP or CSD or CSA

A Deeper Dive of the CSx Resources

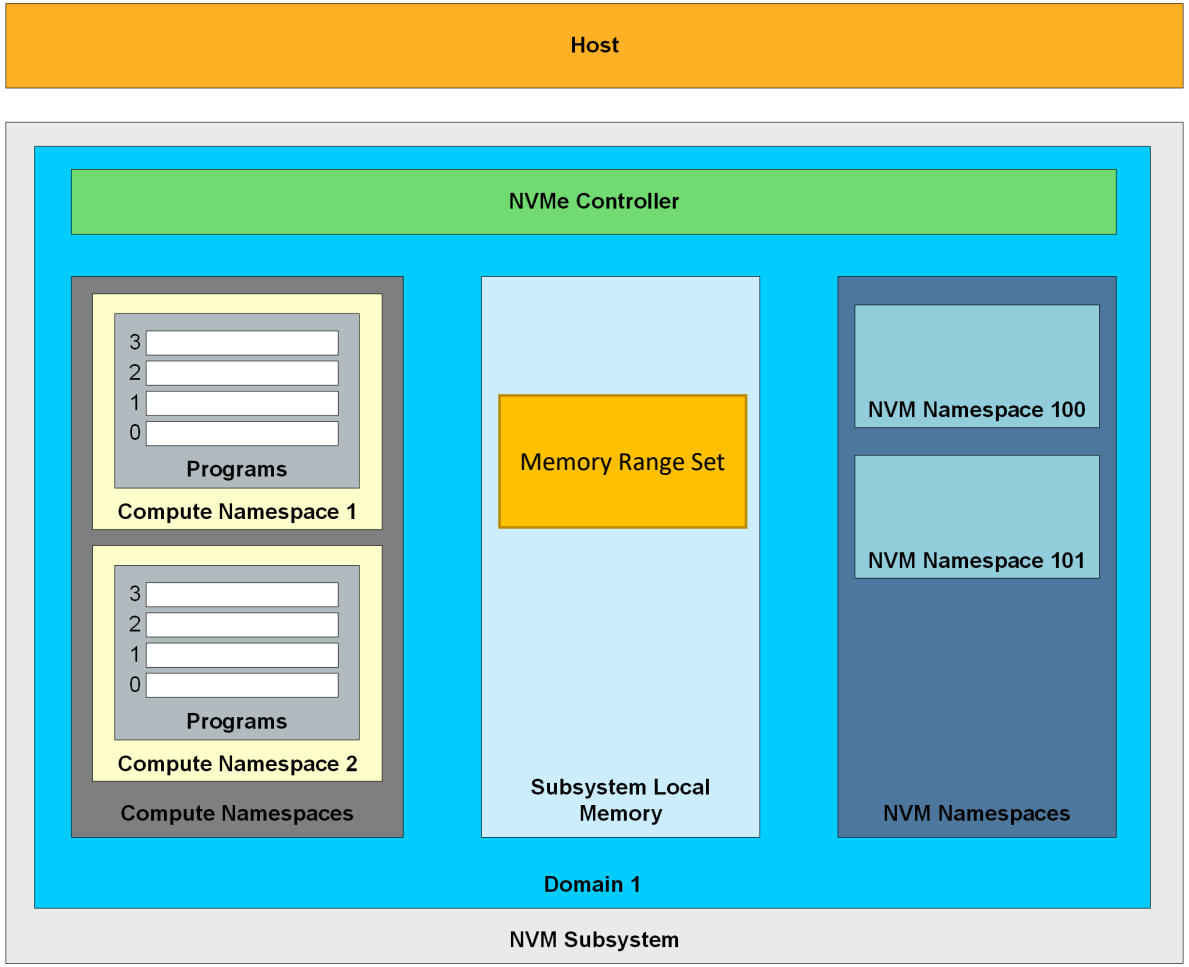


Computational Storage Drive (CSD)

- **CSR -** Computational Storage Resources are the resources available in a CSx necessary for that CSx to store and execute a CSF
- **CSF -** A Computational Storage Function is a set of specific operations that may be configured and executed by a CSE in a CSEE
- **CSE -** Computational Storage Engine is a CSR that is able to be programmed to provide one or more specific operation(s)
- **CSEE -** A Computational Storage Engine Environment is an operating environment space for the CSE
- **FDM -** Function Data Memory is device memory that is available for CSFs to use for data that is used or generated as part of the operation of the CSF
- **AFDM -** Allocated Function Data Memory is a portion of FDM that is allocated for one or more specific instances of a CSF operation
- **Resource Repository –** Resources that are available but not activated

NVMe® Computational Storage Basics

- Computational Programs command set introduced Compute Namespace
- Subsystem Local Memory (SLM) command set introduced Memory Namespace
- Compute Namespace can access SLM® Namespace using a Memory Range Set
- CSE = Compute Engine
- CSF = Program
- Function Data Memory (FDM) = SLM
- Allocated FDM = Memory Range Set
- Device Storage = NVM Namespace



Combining NVMe® and CXL® Technologies

Benefits of CXL® Load/Store Access

- What does CXL bring to the table that benefits NVMe® technology?
  - Provides coherent memory between a host and one or more devices with SLM
  - Provides low latency, fine granularity path to access SLM
  - CXL.mem provides direct Load/Store access to SLM
  - Supports larger memory capacities

- How is this different from PCIe® BAR Access?
  - CXL allows both coherency with host memory and MMIO space - PCIe BAR access only allows host Load/Store access over PCIe using uncached MMIO space
  - CXL provides coherency for device access to host memory
  - CXL protocol is more efficient than PCIe memory access protocol
    - CXL enables lower latency and higher throughput
    - CXL protocol has less strict ordering rules than PCIe memory access protocol
  - CXL allows Peer-to-Peer communication using CXL.mem instead of MMIO access

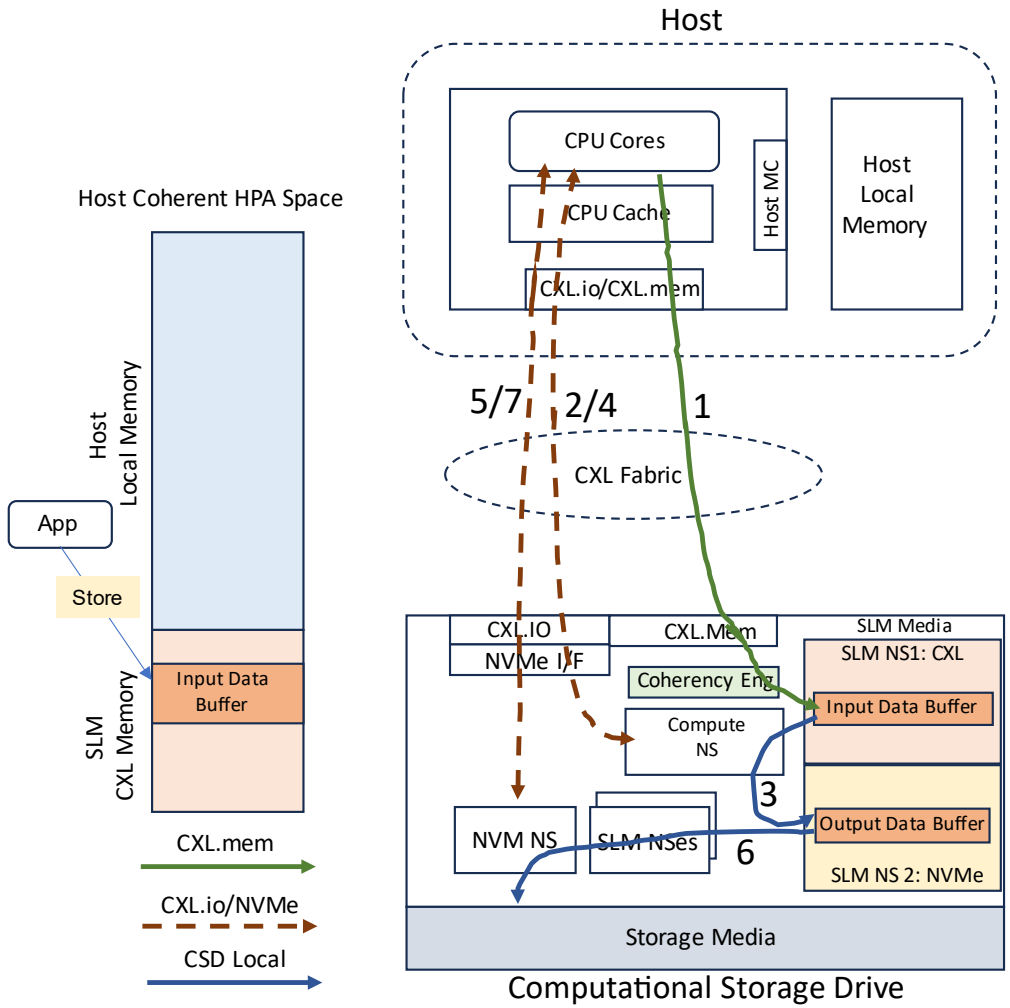
Benefits of Coherency

- All devices perceive the same view of memory
  - Memory viewed between devices is consistent
- All devices perceive the same view of shared data
  - Data is up-to-date
- Devices and hosts can push data to each other or pull data from each other
  - This includes device-to-device communication
- Avoids or reduces copies that can grow stale

Use Cases

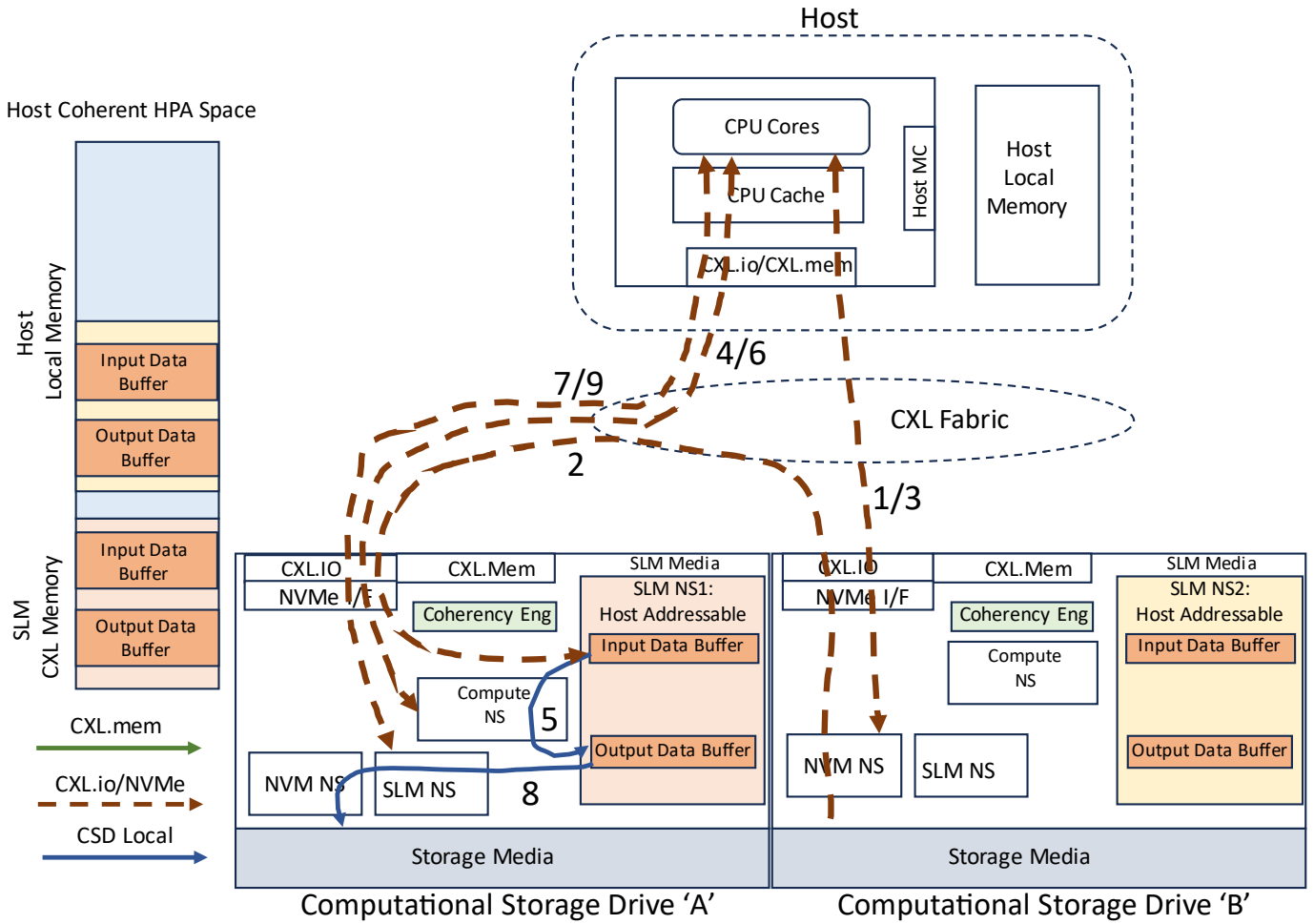
Use Case 1: Data Post-Processing (Before writing to storage)

- Value Proposition
  - Avoid copying data using DMA from/to Host Memory
  - Lower latency CXL® based direct Load/Store access, especially for small input data
- Configuration
  - Input Data Buffer is in host addressable SLM memory address space
  - Output Data Buffer is in SLM
- Example Use Case
  1. Application writes (Store) Input Data Buffer using CXL.mem
     - Some or all data may reside in Host Cache on completion
  2. Host issues NVMe® Execute Program command to Compute Namespace
  3. Compute Namespace Operates on data in Input Data Buffer and stores results in Output Data Buffer
     - Uses CXL BI Snoop protocol to keep Host Cache coherent with Input Data Buffer
  4. CQE is posted for the Compute Namespace
  5. Host issues NVMe Copy command to copy data from Output Data Buffer to Storage Media
  6. Data is copied to Storage Media from Output Data Buffer
  7. CQE is posted for the NVM Namespace



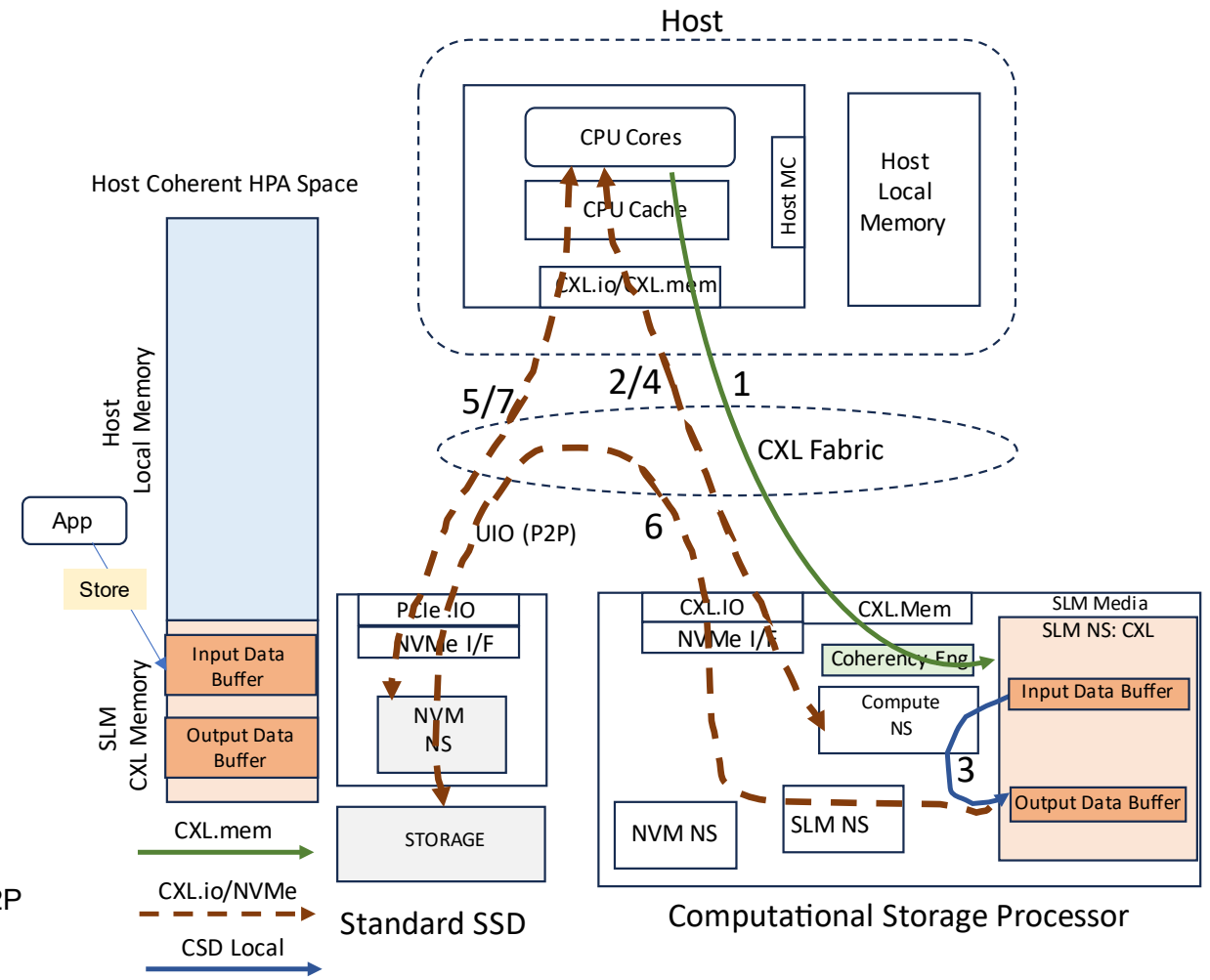
Use Case 2: Peer Data Processing

- Value Proposition
  - Bypass data movement through host memory
- Configuration
  - Data resides in peer device and is moved to Input Data Buffer
  - Output Data Buffer is in host addressable SLM
- Example Use Case
  1. Host issues NVMe® Read command to Drive B with the destination is Input Data Buffer on Drive A
  2. Data is transferred P2P from Drive B to Drive A
  3. CQE is posted for READ command
  4. Host issues NVMe Execute Program command to Compute Namespace on Drive A
  5. Compute Namespace on Drive A operates on data in Input Data Buffer and stores results in Output Data Buffer
  6. CQE is posted for the Compute Namespace
  7. Host issues NVMe Copy command to copy data from Output Data Buffer to Storage Media
  8. Data is copied to Storage Media from Output Data Buffer
  9. CQE is posted for the NVM Namespace



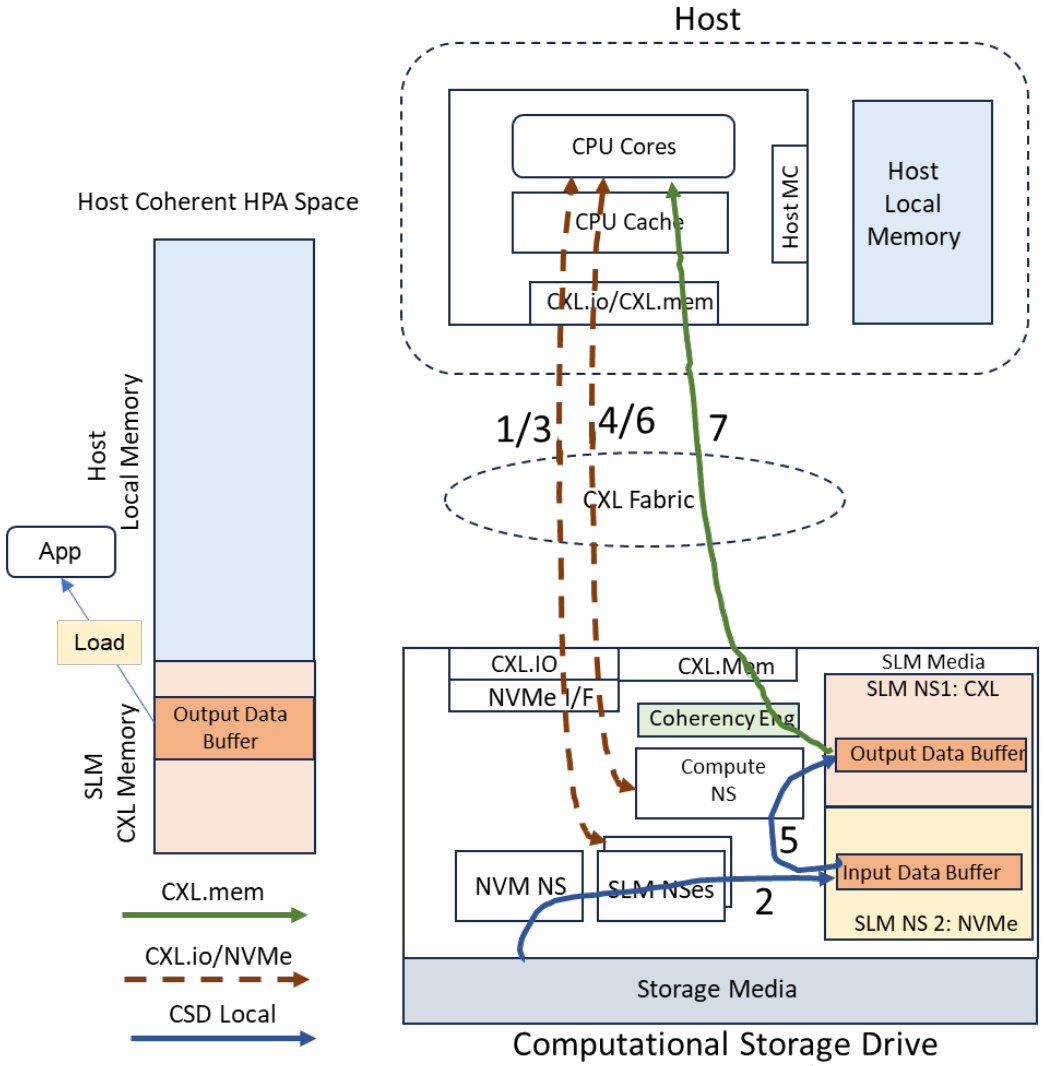
Use Case 3: Data Post-Processing with a Standard SSD

- Value Proposition
  - Bypass data movement through Host memory
- Configuration
  - Input Data Buffer is in host addressable SLM memory address space
  - Output Data Buffer is in host addressable SLM memory address space
- Example Use Case
  1. Application writes (Store) Input Data Buffer using CXL.mem
     - Some or all data may reside in Host Cache on completion
  2. Host issues NVMe® Execute Program command to Compute Namespace
  3. Compute Namespace operates on data in Input Data Buffer and stores results in Output Data Buffer
     - Uses CXL BI Snoop protocol to keep Host Cache coherent with Input Data Buffer and Output Data Buffer
  4. CQE is posted for Compute Namespace
  5. Host generates IO Write to SSD NVM Namespace
     - Data Pointer points to Output Buffer in SLM (HDM)
  6. SSD uses PCIe® UIO for direct P2P from HDM space and writes to storage media
     - Since output buffer is in CXL HDM space, UIO can't use BAR space for P2P
  7. CQE is posted for NVM Namespace



Use Case 4: Data Pre-Processing (before sending to host)

- Value Proposition
  - Avoid copying data using DMA from/to Host Memory
  - Lower latency CXL based direct Load/Store access, especially for small output data
- Configuration
  - Input Data Buffer is in SLM
  - Output Data Buffer is in host addressable SLM
- Example Use Case
  1. Host issues NVMe Memory Copy command to SLM NS
  2. Data copied from NVM NS to Input Data Buffer
  3. CQE is posted for SLM NS
  4. Host issues NVMe Execute Program command to Compute NS
  5. Compute NS operates on data in Input Data Buffer and stores results in Output Data Buffer
     - Uses CXL BI Snoop protocol to keep Host caches coherent with Output Data Buffer
  6. CQE is posted for Compute NS
  7. Application reads (ld/st) Output Data Buffer using CXL.mem



Summary and Next Steps

- CXL® and NVMe® technologies can be used simultaneously
  - Coherent memory between a host and one or more devices with SLM
  - CXL.mem provides direct Load/Store access to SLM
- TP4184
  - Enables Host Addressable SLM
    - Both CXL and PCIe® BAR access methods
  - Currently in the architecture phase
- Looking Ahead
  - CXL and NVMe Computational Storage are on trajectories that will intersect
  - Enhancing NVMe SLM to support CXL is a step to enable convergence/collaboration

What do you think Computational Storage is

- Please complete a survey on your view of Computational Storage
  1) What is Computational Storage? (Multiple Choice)
  2) How would you use computational storage? (Fill in the blank)
  3) What is the future and evolution of computational storage? (Multiple Choice)
  4) Any other thoughts/ideas on computational storage?

Please take a moment to rate this session.

Your feedback is important to us.

