CXL and NVMe Collaborating for Computation

Jason Molgaard
Principal Storage Solutions Architect
Solidigm

REGIONAL SDC 24
BY Developers FOR Developers
APRIL 24, AUSTIN, TX

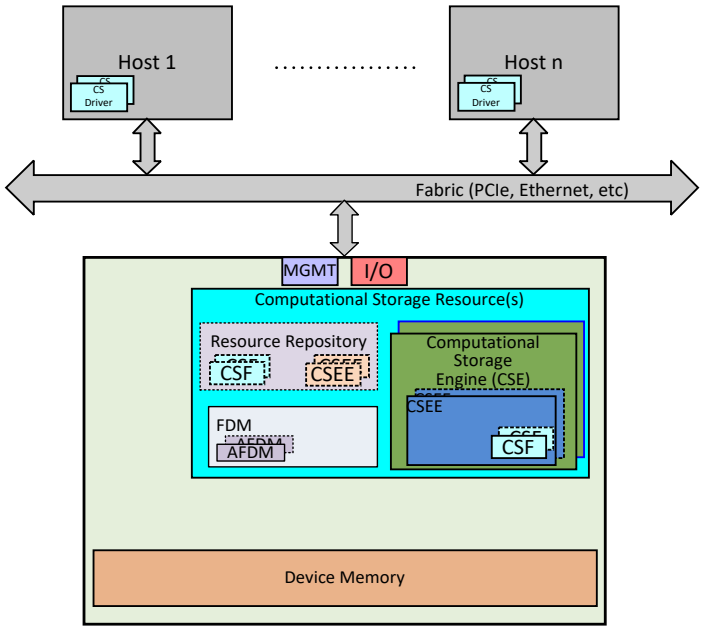
A SNIA Event

Agenda

- Computational Storage Basics
- Combining CXL and NVMe
- Use Cases

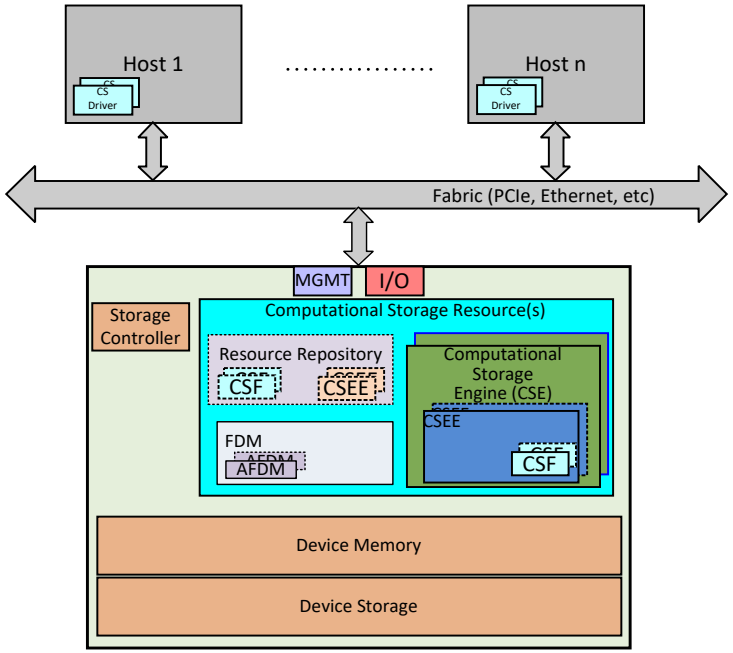
Computational Storage Architecture

Computational Storage Processor



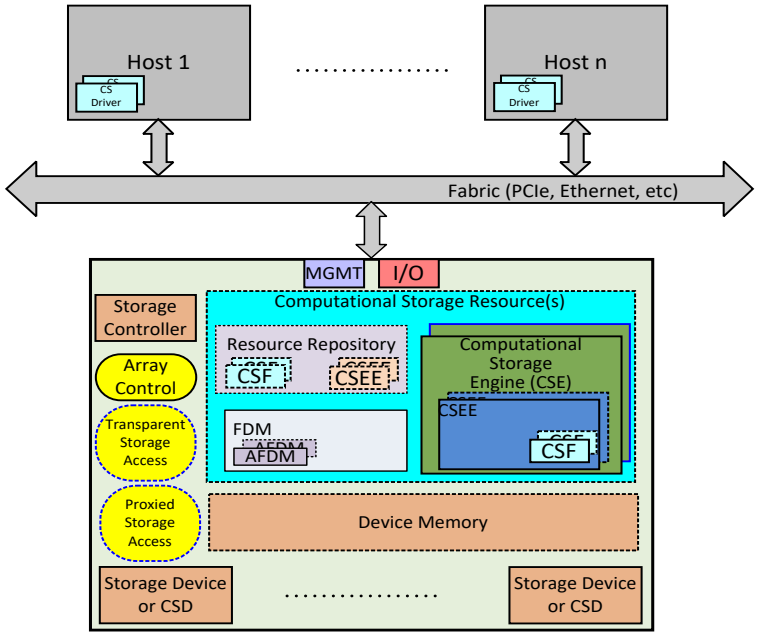
Computational Storage Processor (CSP)

Computational Storage Drive



Computational Storage Drive (CSD)

Computational Storage Array

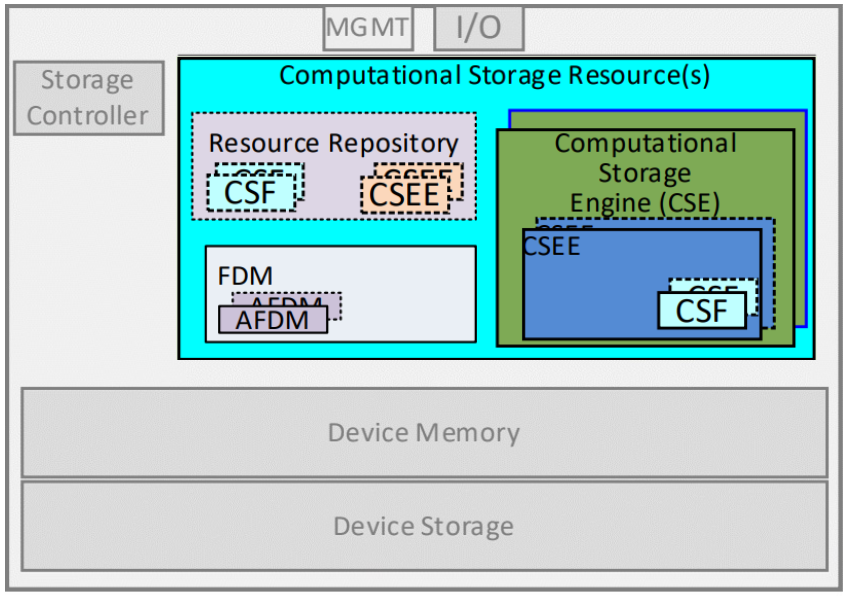


Computational Storage Array (CSA)

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

REGIONAL SDC 24

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.

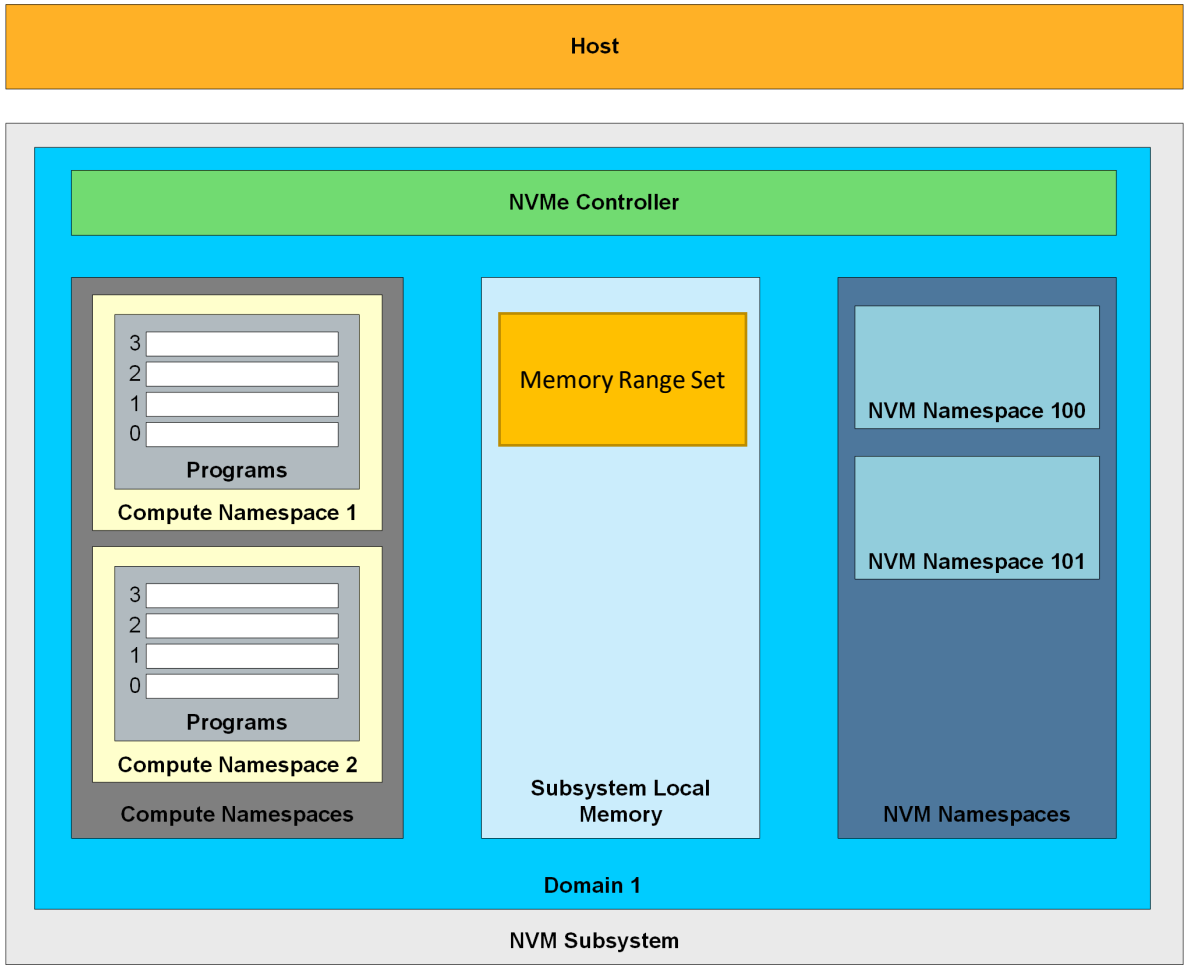
AFDM - Allocated Function Data Memory is a portion of FDM that is allocated for one or more specific instances of a CSF operation.

Resource Repository – Resources that are available but not activated

REGIONAL SDC 24

NVMe Computational Storage Basics

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

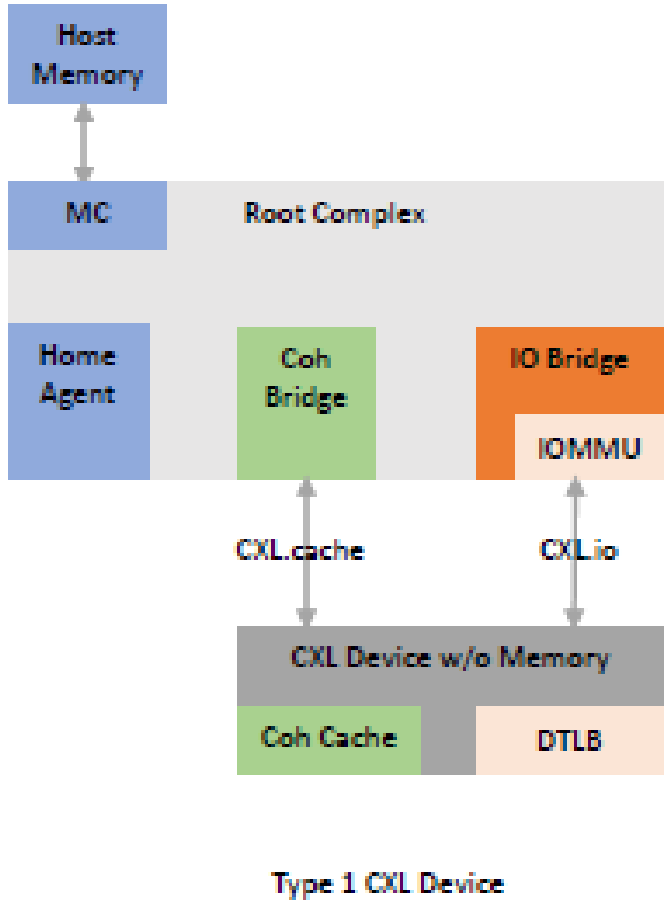


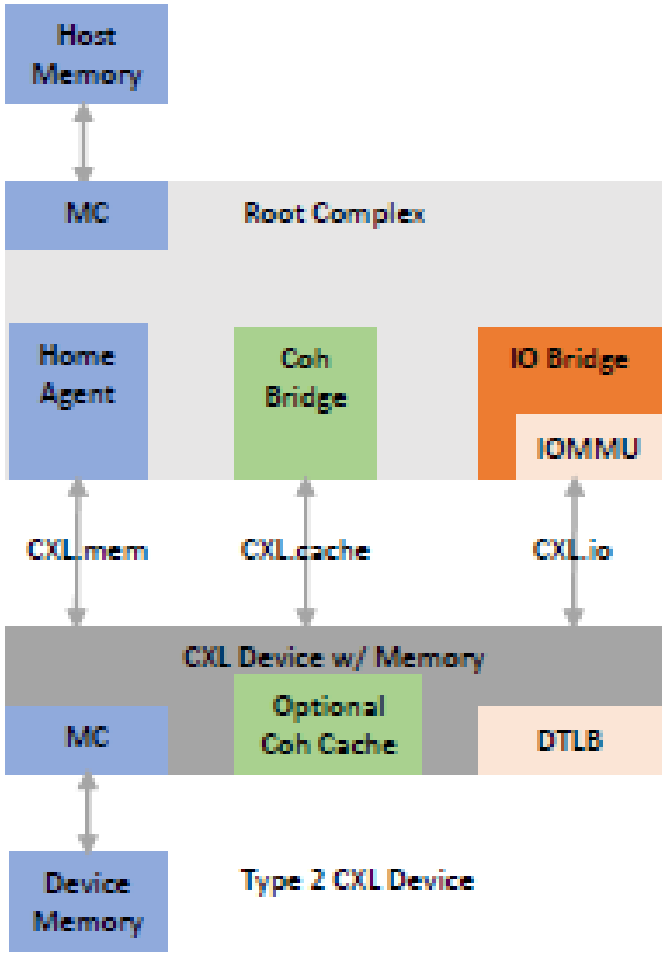
Combining CXL and NVMe

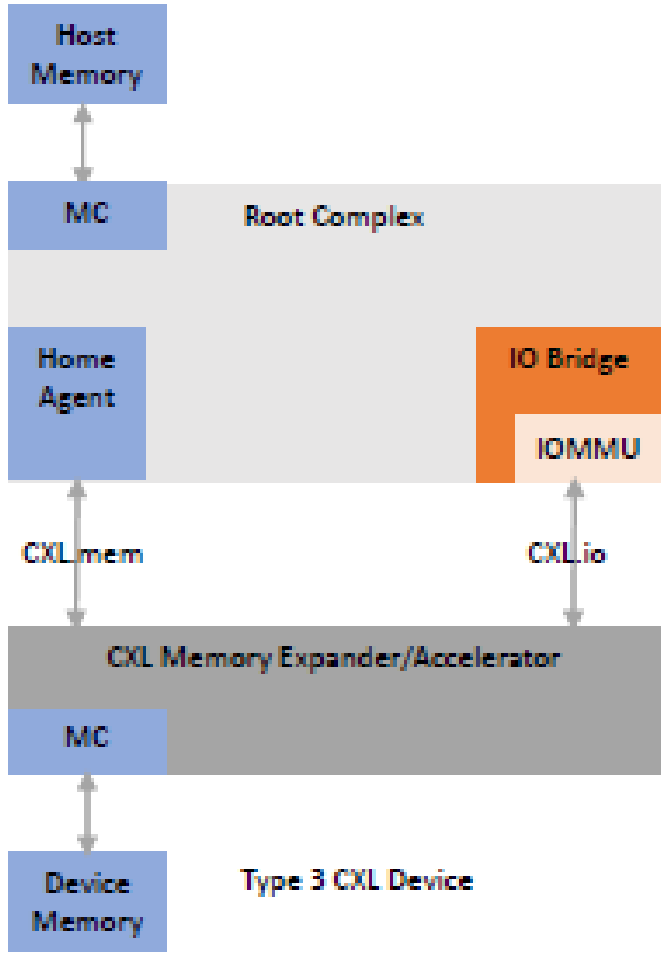
REGIONAL SDC 24

BY Developers FOR Developers

CXL System Architecture







REGIONAL SDC 24

Why Combine CXL and NVMe?

- Memory and Storage are converging
- Computational Storage

What if ?...

- The host could interact with the CSx using load/store semantics?
- The host could be coherent with the CSx memory?
- The FDM/SLM could be an extension of the host memory?

Benefits of CXL Load/Store Access

- What does CXL bring to the table that we don't have in NVMe?
  - Allows coherent memory between a host and one or more devices with SLM
  - Low latency, fine granularity path to access FDM/SLM
  - CXL.mem allows direct load/store access to FDM/SLM
- How is this different from CMB/PMR?
  - CMB/PMR 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, enabling lower latency and higher throughput
    - PCIe has more strict ordering rules

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
- Avoids or reduces copies that can grow stale

Host Addressable Device Memory

- FDM/SLM memory can be shared with the host
- FDM/SLM can be read/written with CXL.mem commands
- Compute can still be triggered with Computational Programs commands

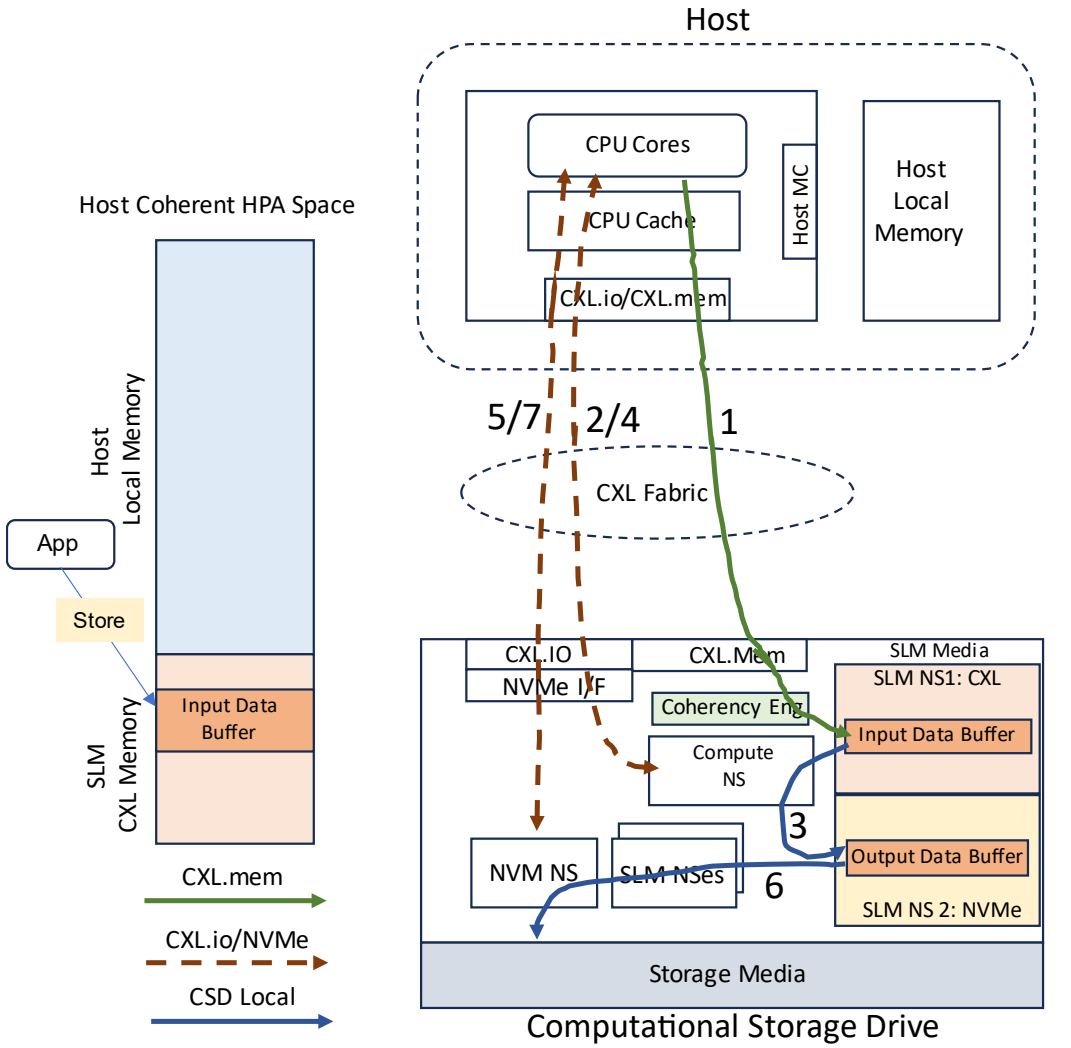
Use Cases

REGIONAL SDC 24

BY Developers FOR Developers

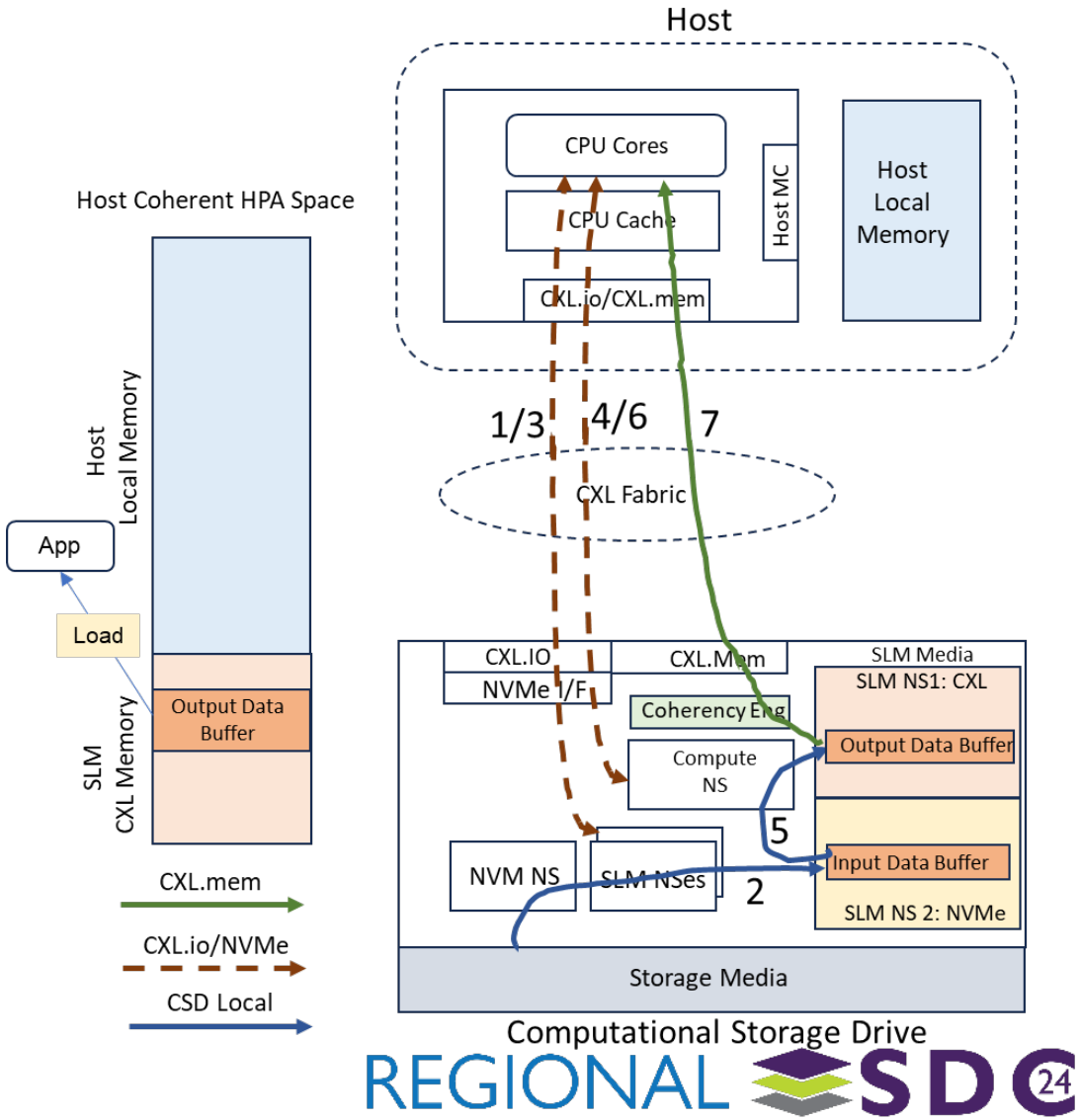
Use Case 1: Data Post-Processing (before writing to storage)

- Value Proposition
  - Avoid copying data using DMA from/to Host Memory
  - Lower latency CXL based direct ld/st access, especially for small input data
- Configuration
  - Input Data Buffer is in SLM CXL memory address space
  - Output Data Buffer is in SLM
- Example Use Case
  1. Application writes (ld/st) 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 NS
  3. Compute NS 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 NS
  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 NS



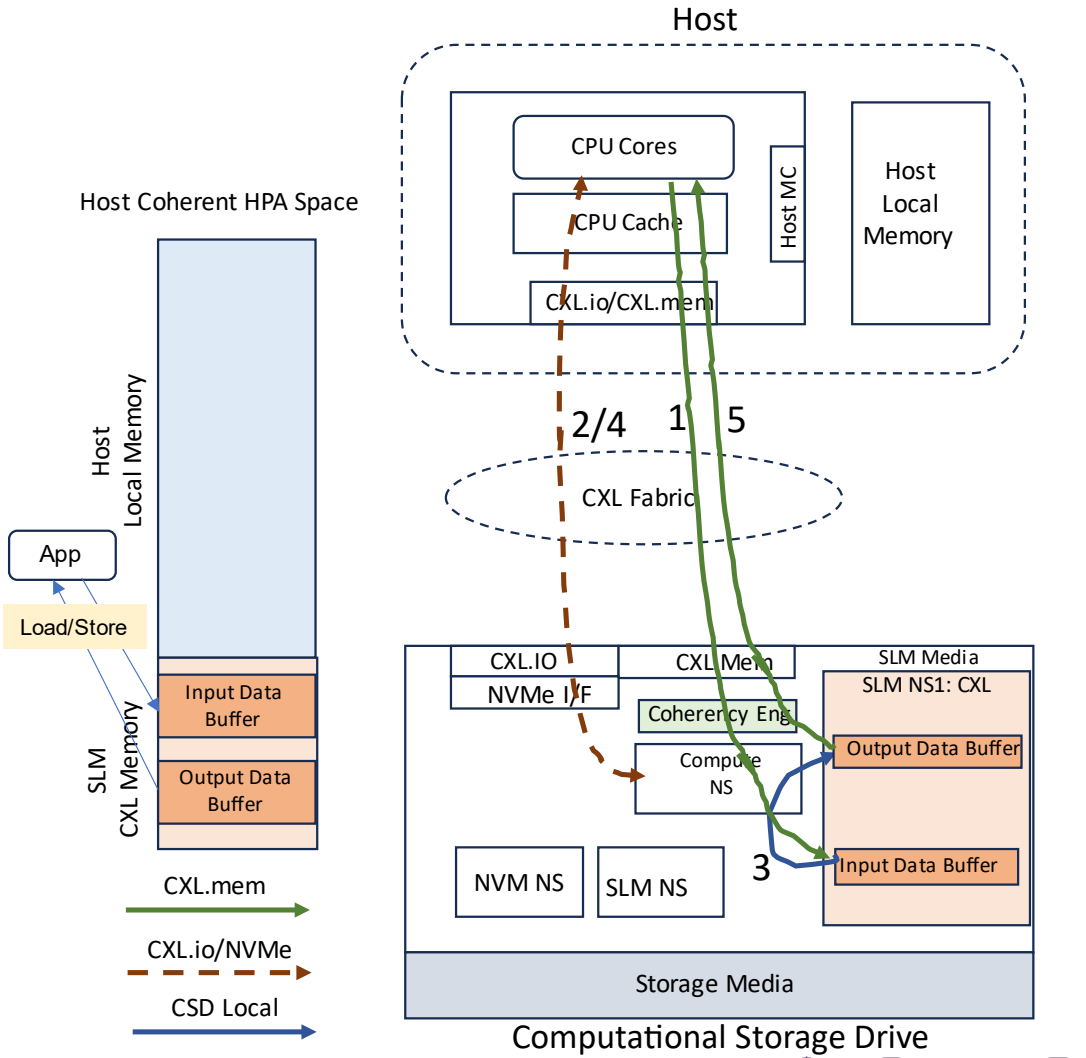
Use Case 2: Data Pre-Processing (before sending to host)

- Value Proposition
  - Avoid copying data using DMA from/to Host Memory
  - Lower latency CXL based direct ld/st access, especially for small output data
- Configuration
  - Input Data Buffer is in SLM
  - Output Data Buffer is in SLM CXL memory address space
- 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



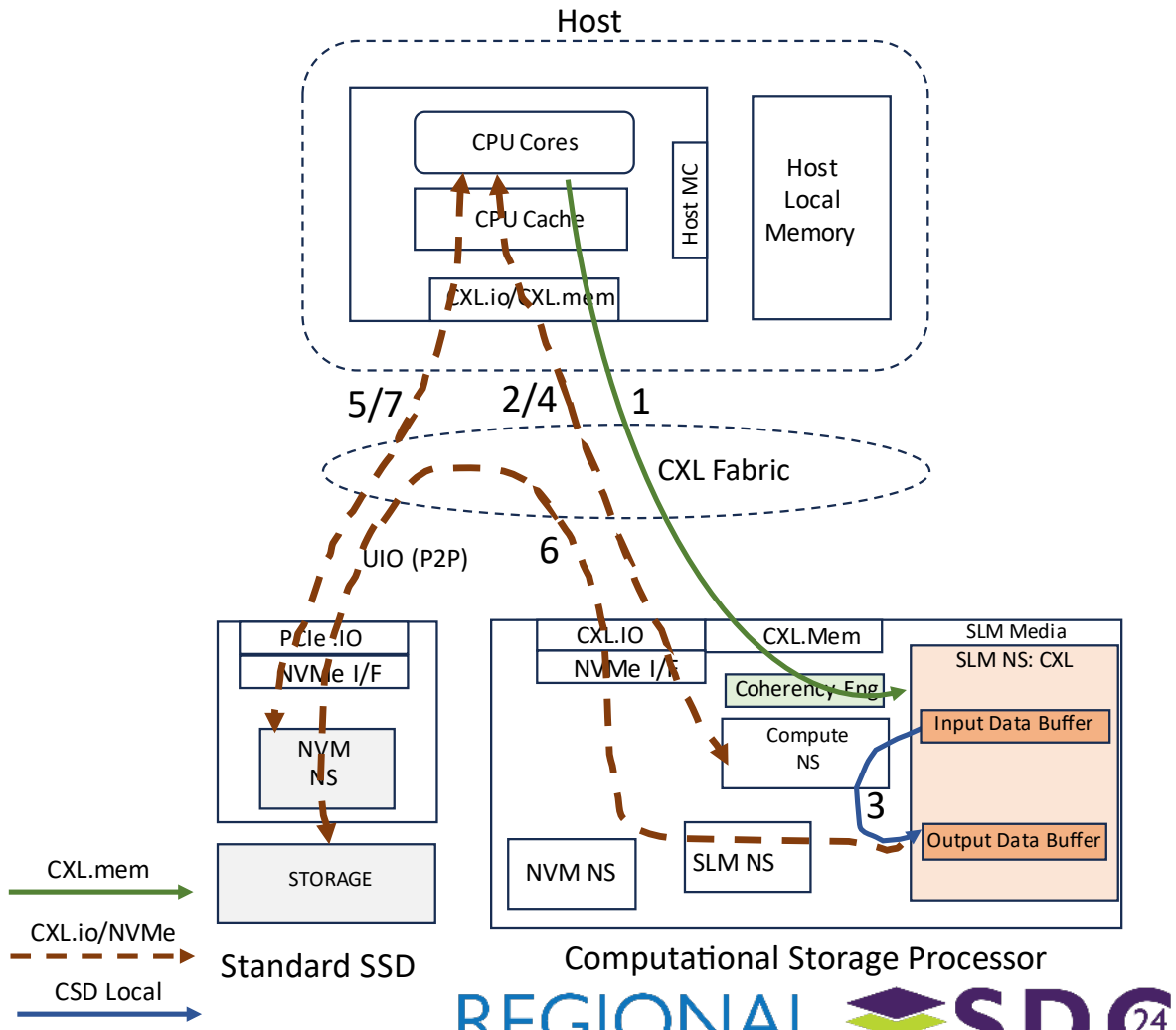
Use Case 3: Compute Offload

- Value Proposition
  - Avoid copying data using DMA from/to Host Memory
  - Lower latency CXL based direct ld/st access, especially for small input/output data
  - Enables general purpose compute offload
- Configuration
  - Input Data Buffer is in SLM CXL memory address space
  - Output Data Buffer is in SLM CXL memory address space
- Example Use Case
  1. Application writes (ld/st) 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 NS
  3. Compute NS 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 NS
  5. Application reads (ld/st) Output Data Buffer using CXL.mem



Use Case 4: Data Post-Processing with a Standard SSD

- Value Proposition
  - Bypass data movement through Host memory
- Configuration
  - Input Data Buffer is in SLM CXL memory address space
  - Output Data Buffer is in SLM CXL memory address space
- Example Use Case
  1. Application writes (ld/st) 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 NS
  3. Compute NS 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 NS
  5. Host generates IO Write to SSD NVM NS
     - 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 NS



Summary and Next Steps

- CXL and NVMe can be used simultaneously
  - CXL brings Load/store access to SLM
  - CXL enables Host and CSx sharing data coherently
  - While still supporting existing command sets, especially Computational Storage
- Benefits
  - Coherency between the CSx and host
  - Lower latency for small data transfers
  - Avoid copying data
  - Bypassing the host for data movement between devices
- Looking Ahead
  - CXL and Computational Storage with NVMe are on trajectories that will intersect
  - Enhancing NVMe SLM to support CXL is a step to enable convergence/collaboration

THANK YOU

Please take a moment to rate this session.

REGIONAL SDC 24

BY Developers FOR Developers