STORAGE DEVELOPER CONFERENCE

SDC 23

BY Developers FOR Developers

Envisioning a Computational Storage Architecture with an SDXI Data Mover: Early Efforts

Shyam Iyer

Jason Molgaard

Agenda

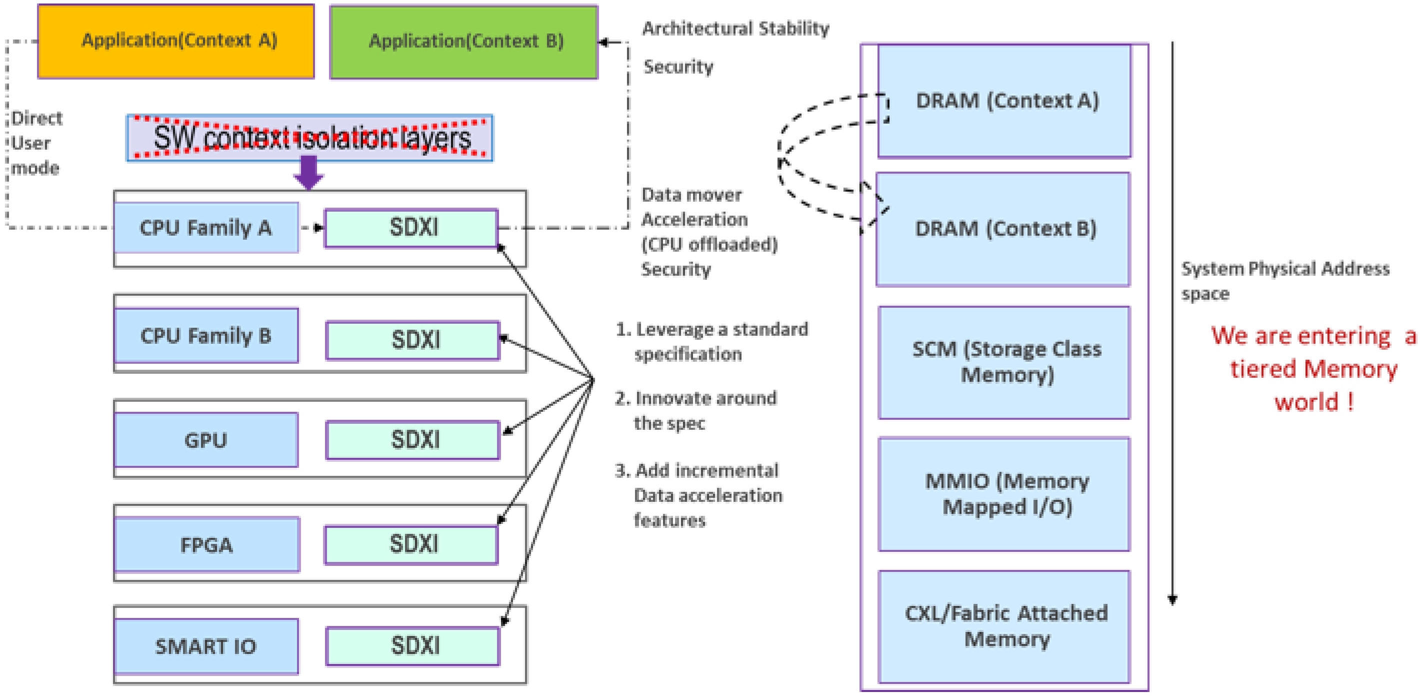
- Overview of SDXI
- Overview of Computational Storage
- SDXI+CS Combination

SDXI Overview

SDXI(Smart Data Accelerator Interface)

- Software `memcpy` is the current data movement standard
  - Stable ISA
  - However,
    - Takes away from application performance
    - Incurs software overhead to provide context isolation.
    - Offload DMA engines and their interfaces are vendor-specific
    - Not standardized for user-level software.
- Smart Data Accelerator Interface (SDXI) is a SNIA standard for a memory to memory data movement and acceleration interface that is -
  - Extensible
  - Forward-compatible
  - Independent of I/O interconnect technology
- SNIA SDXI TWG was formed in June 2020 and tasked to work on this proposed standard
  - 23 member companies, 89 individual members
- **v1.0 released!**
  - https://www.snia.org/sdxi

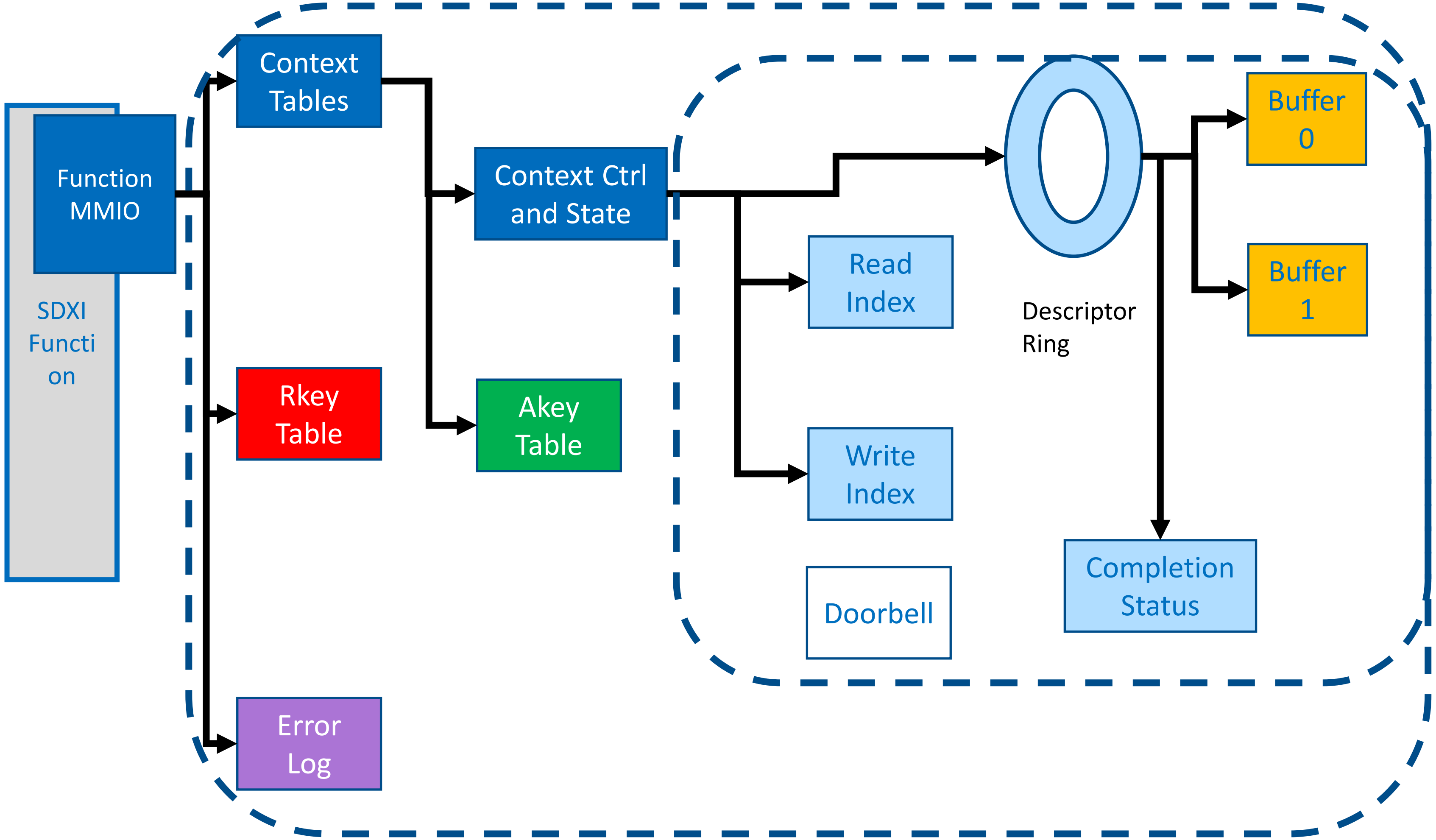
SDXI Memory-to-Memory Data Movement



Application(Context A)
Application(Context B)
Architectural Stability
Security
Direct User mode
SW context isolation layers
CPU Family A
SDXI
CPU Family B
SDXI
GPU
SDXI
FPGA
SDXI
SMART IO
SDXI
Data mover Acceleration (CPU offloaded) Security
1. Leverage a standard specification
2. Innovate around the spec
3. Add incremental Data acceleration features
DRAM (Context A)
DRAM (Context B)
SCM (Storage Class Memory)
MMIO (Memory Mapped I/O)
CXL/Fabric Attached Memory
System Physical Address space
We are entering a tiered Memory world !

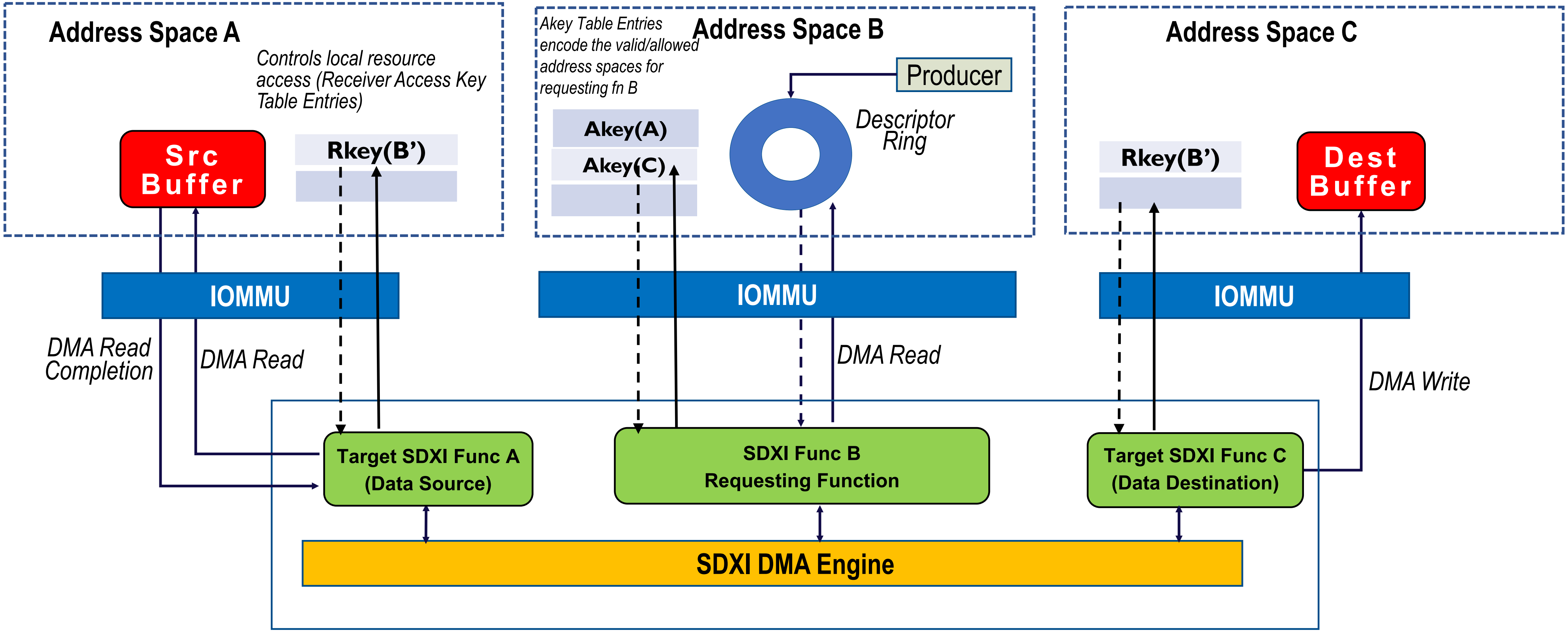
- Data movement between different address spaces.
- Data movement without mediation by privileged software.
- Allows abstraction or virtualization by privileged software.
- Capability to quiesce, suspend, and resume the architectural state of a per-address-space data mover.
- Forward and backward compatibility across future specification revisions.
- Additional offloads leveraging the architectural interface.
- Concurrent DMA model.

Memory Structures(1) – Simplified view

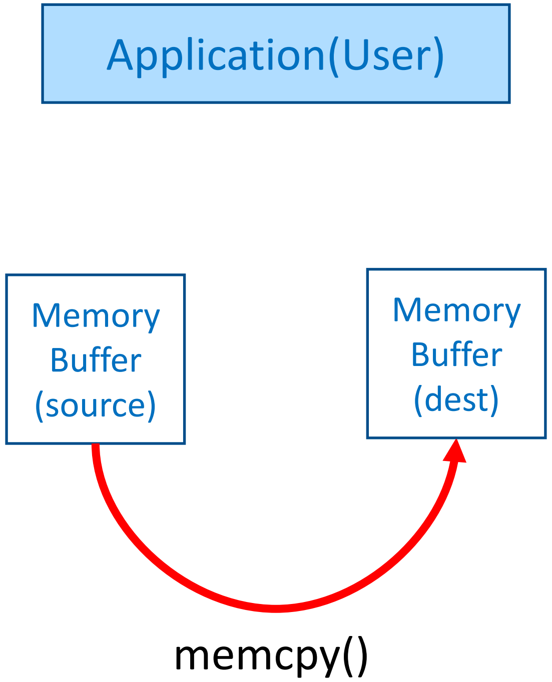


- All states in memory
- One standard descriptor format
  - Scope for future expansion
- Easy to virtualize
- Architected function setup and control
  - *layered model for interconnect specific function management
  - SDXI class code registered for PCIe implementations

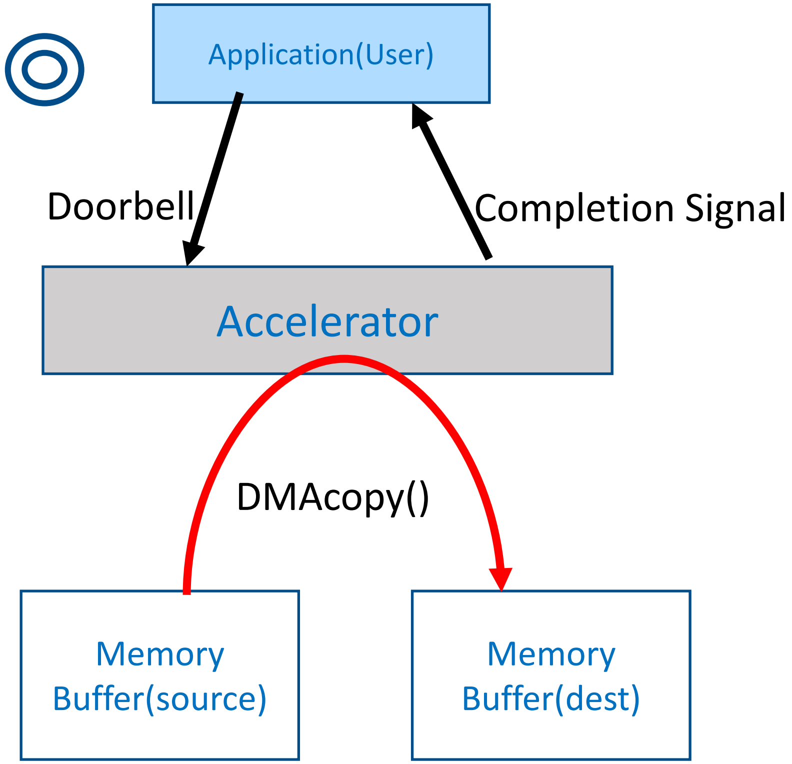
Multi-Address Space Data Movement within an SDXI function group (2)


Address Space A
Controls local resource access (Receiver Access Key Table Entries)
Src Buffer
Rkey(B')
Akey Table Entries encode the valid/allowed address spaces for requesting fn B
Address Space B
Producer
Akey(A)
Akey(C)
Descriptor Ring
Address Space C
Rkey(B')
Dest Buffer
IOMMU
IOMMU
IOMMU
DMA Read Completion
DMA Read
DMA Read
DMA Write
Target SDXI Func A (Data Source)
SDXI Func B Requesting Function
Target SDXI Func C (Data Destination)
SDXI DMA Engine

Application Pattern 1 (Buffer Copies)

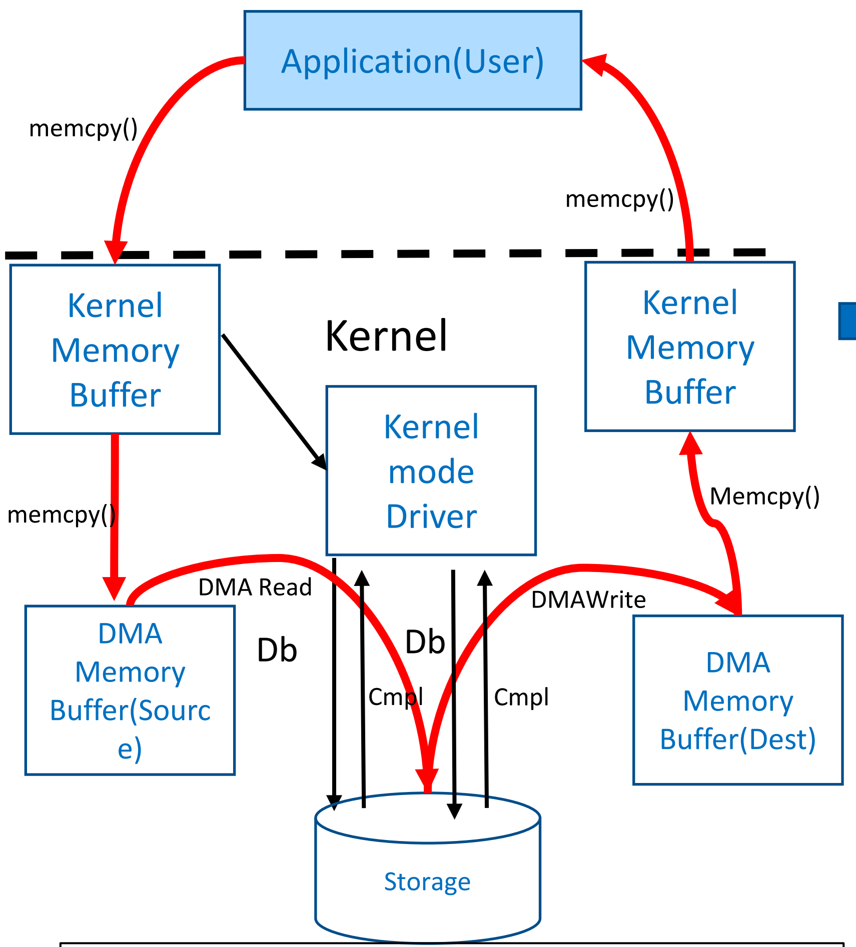


- Takes away from application performance

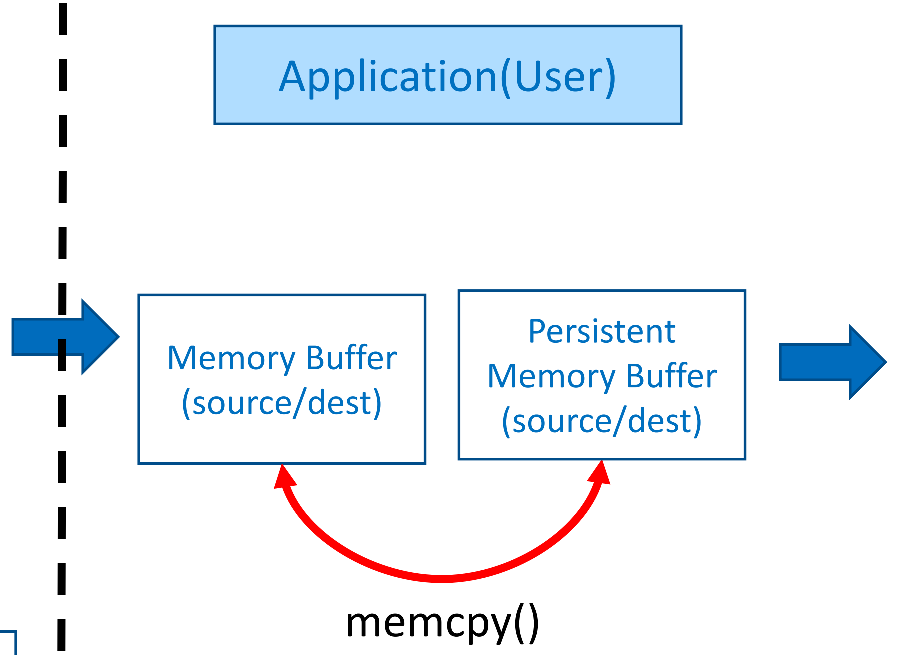


- HW based memory copies can be offloaded without affecting application performance
- Additional computations as the data is copied. For e.g., compression, memory fills, memcmps, etc.

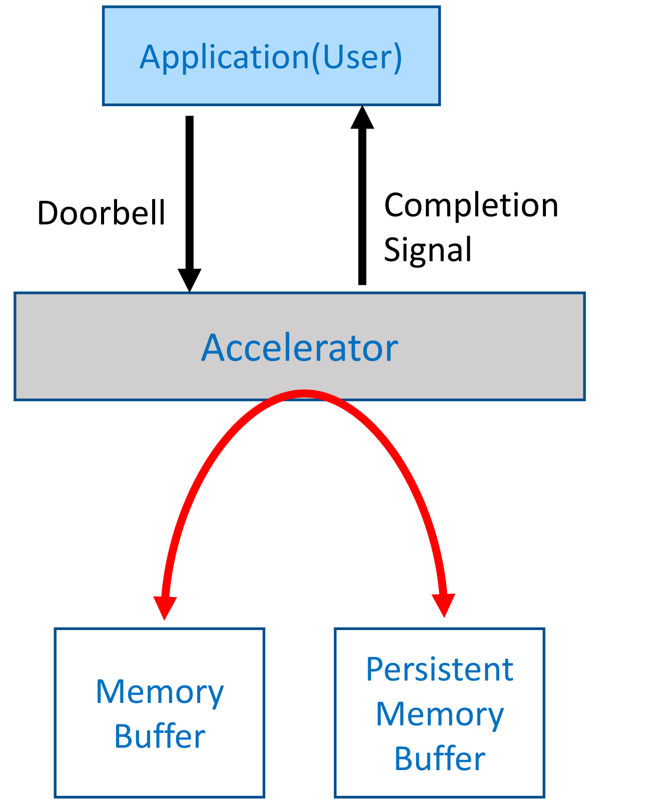
Application Pattern 2 (Storage Data Movement)



- Multiple data buffer copies before hardware-based data movement can occur

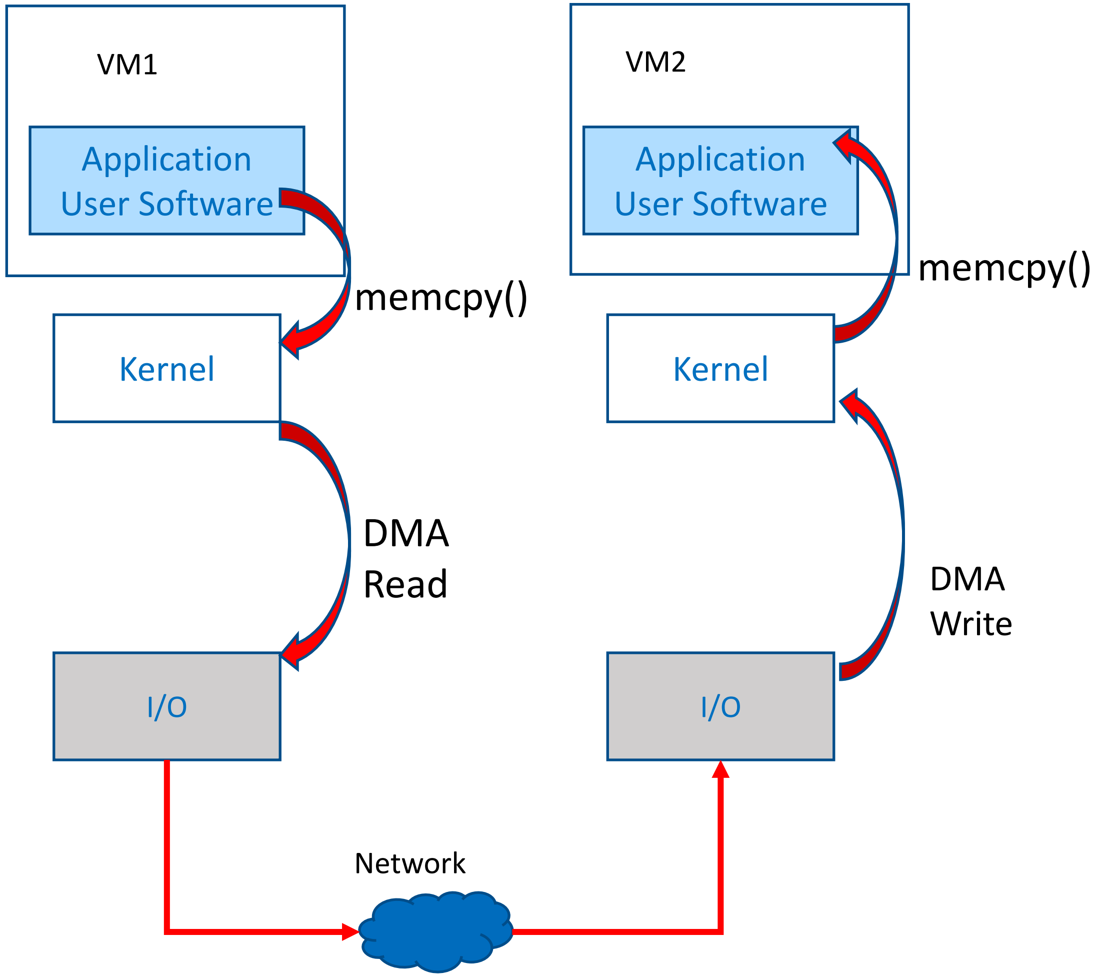

Application(User)
Memory Buffer (source/dest)
Persistent Memory Buffer (source/dest)
memcpy()

- Reduced buffer copies but still takes away from application performance

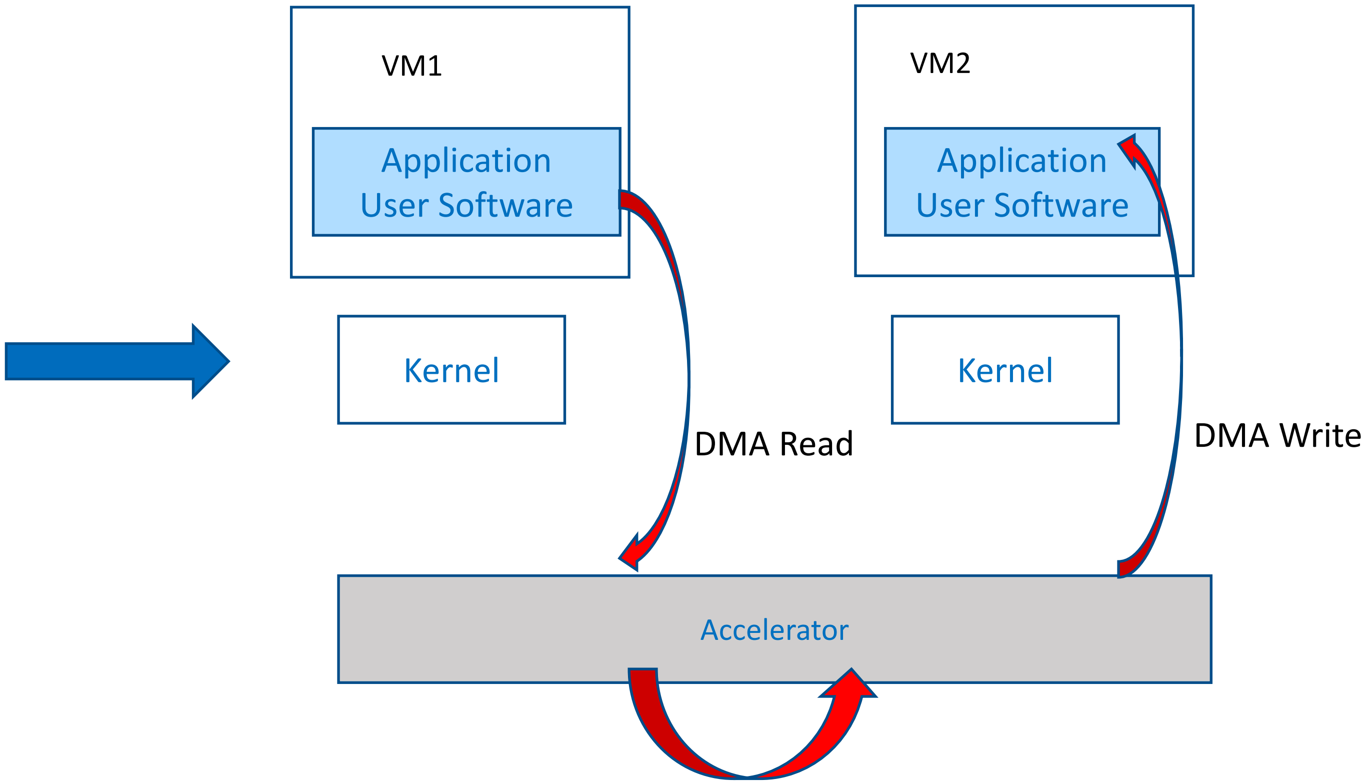


- Reduced buffer copies
- HW based offloaded memory copies

Application Pattern 3 (Virtualized Data Movement)

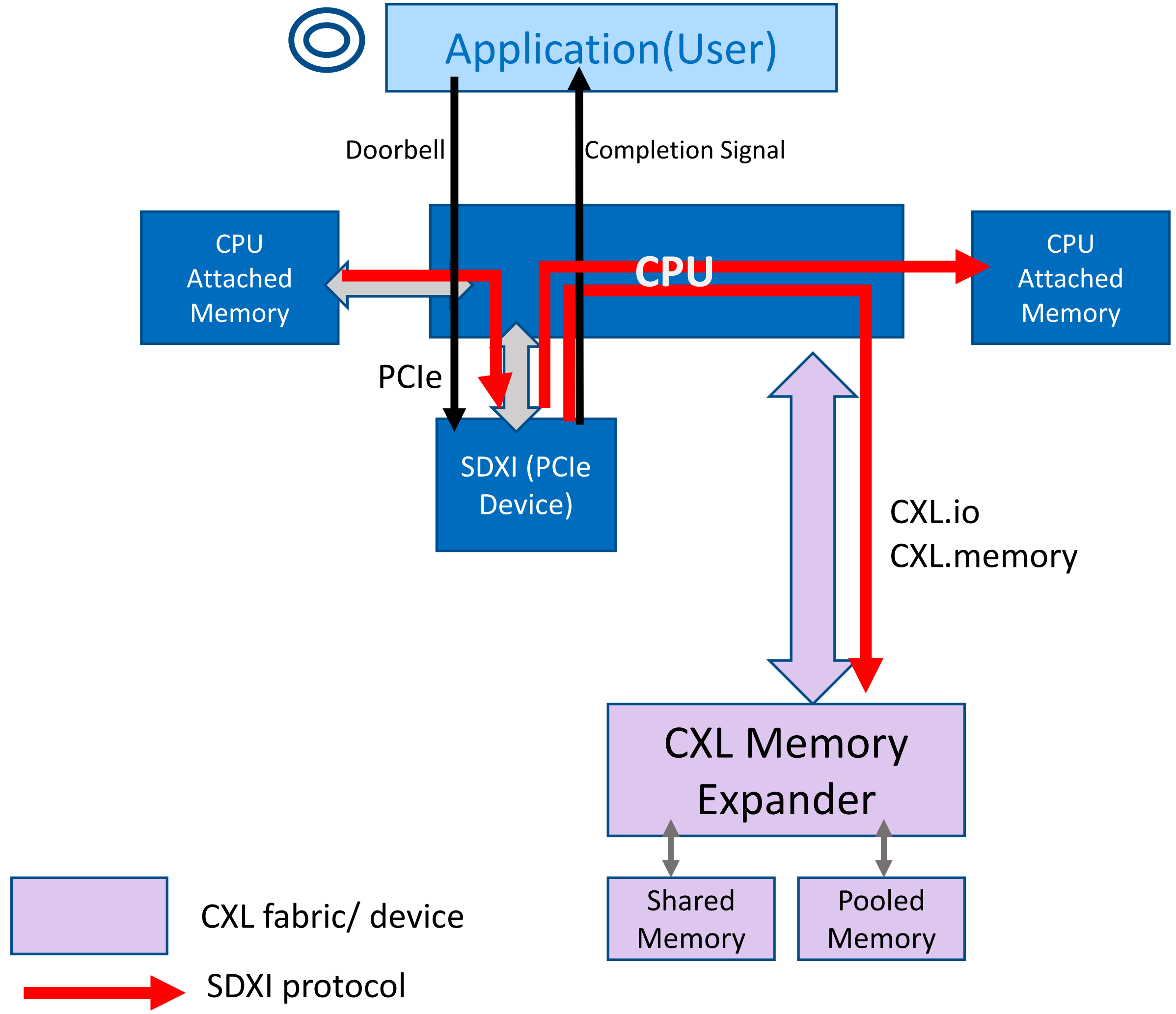


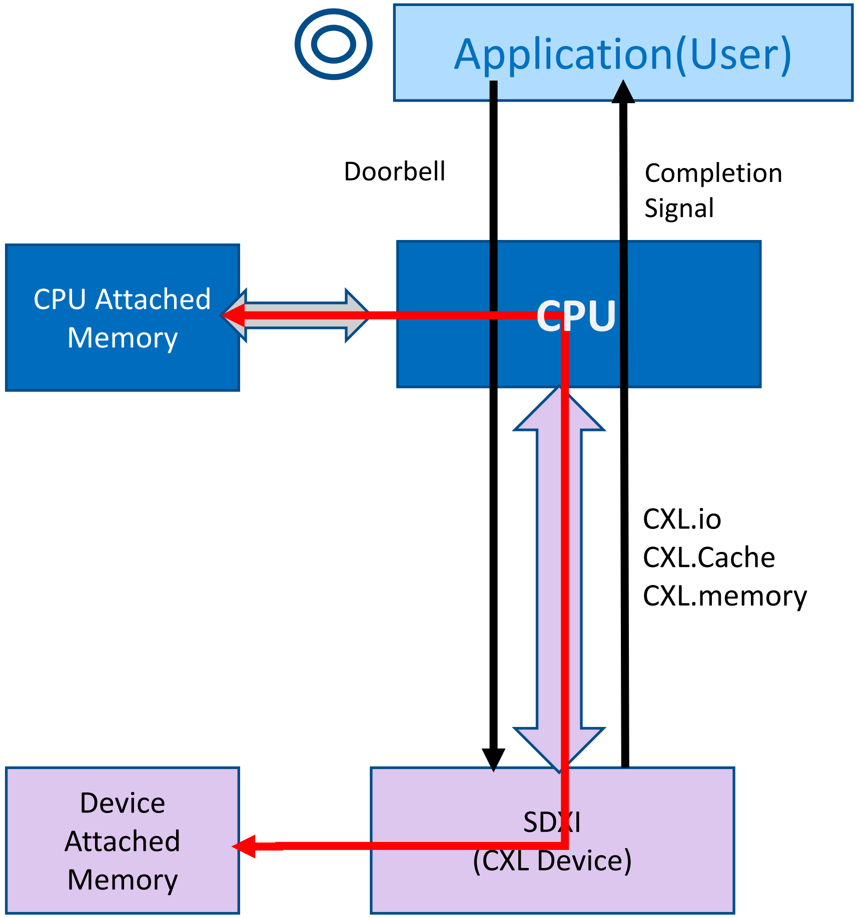
- Context isolation layers introduce multiple buffer copies



- Best of both: Context isolation layers and optimized HW based memory buffer copies

Emerging use cases: SDXI Assisted Data Movement in a CXL Architecture





SDXI v1.1 investigations

- Management architecture for data movers(includes connection manager)
- New data mover operations for smart acceleration
- SDXI Host to Host investigations
- Scalability & Latency improvements
- Cache coherency models for data movers
- Security Features involving data movers
- Data mover operations involving persistent memory targets
- QoS
- CXL-related use cases
- Heterogenous environments

Draft: Subject to Change

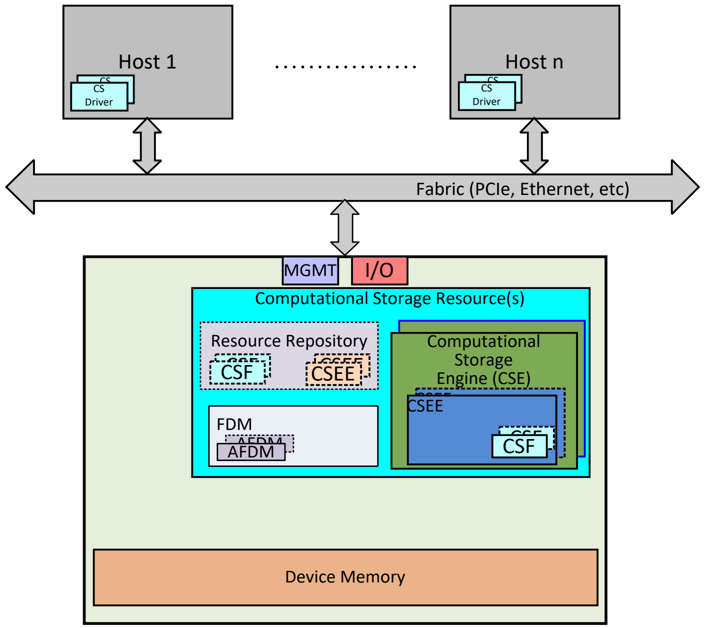
Additional SDXI Ecosystem activities

- SDXI Software group
  - Libsdxi project
    - OS agnostic user space library
  - Linux Upstream driver efforts
    - SDXI TWG members are supporting this effort outside SNIA as a community
  - SDXI emulation project investigation for ecosystem development
  - Investigations to enable SDXI compliance for SW and HW interoperability
- SNIA's CS+SDXI Subgroup

Computational Storage Overview

Computational Storage Architecture

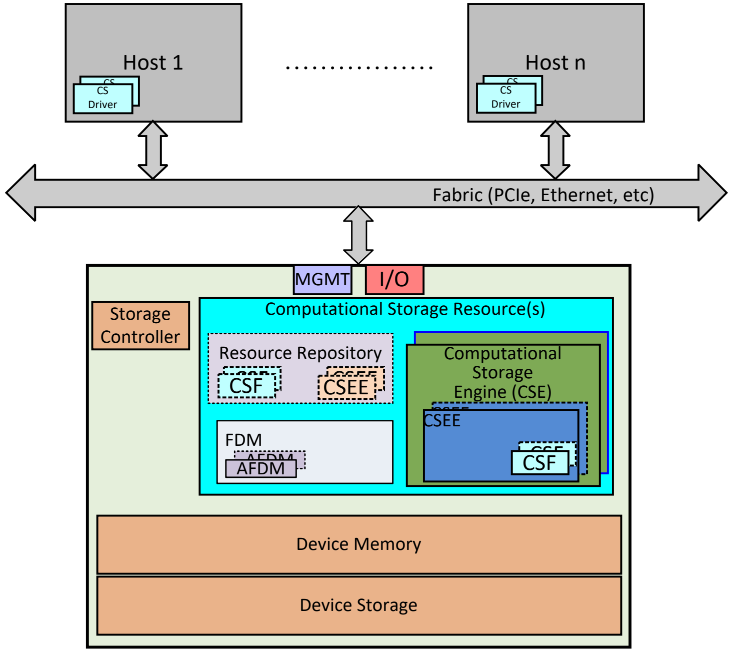
Computational Storage Processor



Host 1
CS Driver
Host n
CS Driver
Fabric (PCIe, Ethernet, etc)
MGMT
I/O
Computational Storage Resource(s)
Resource Repository
CSF
CSEE
Computational Storage Engine (CSE)
CSEE
CSF
FDM
AFDM
Device Memory

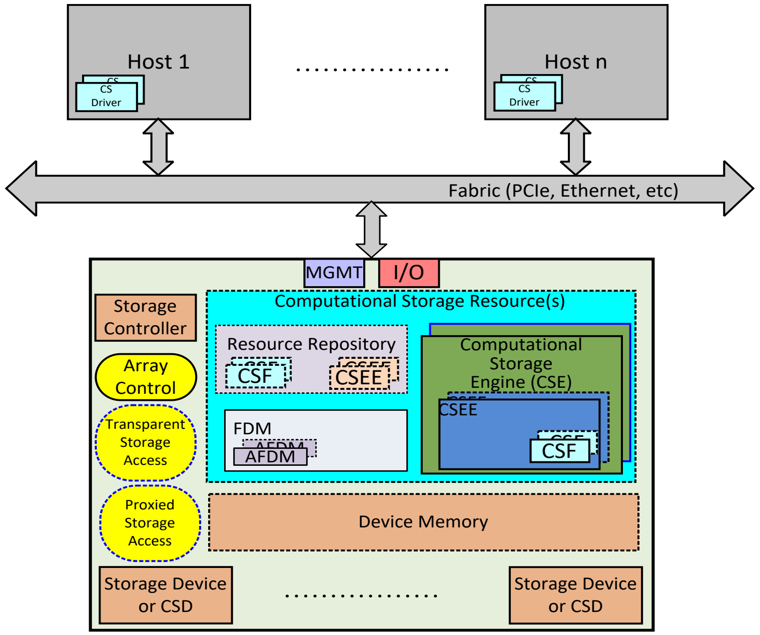
Computational Storage Processor (CSP)

Computational Storage Drive



Computational Storage Drive (CSD)

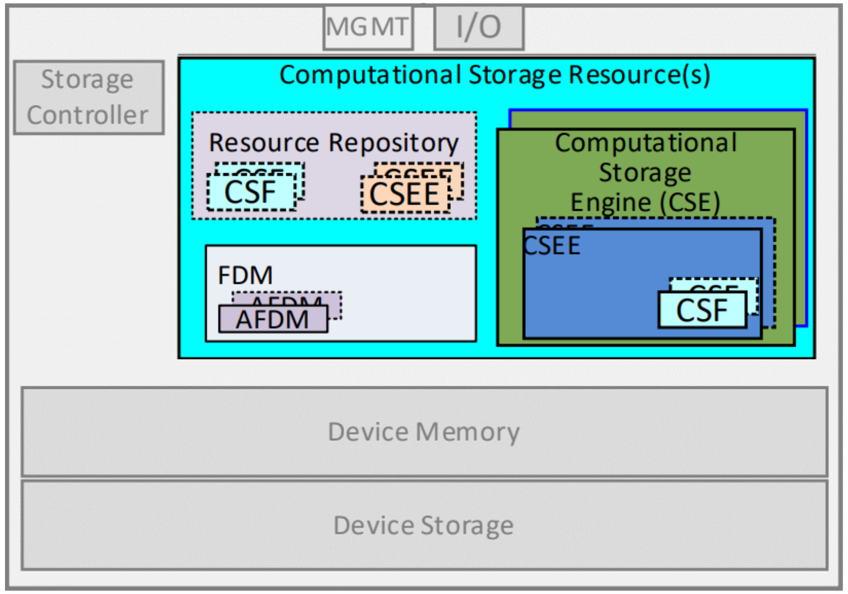
Computational Storage Array



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).

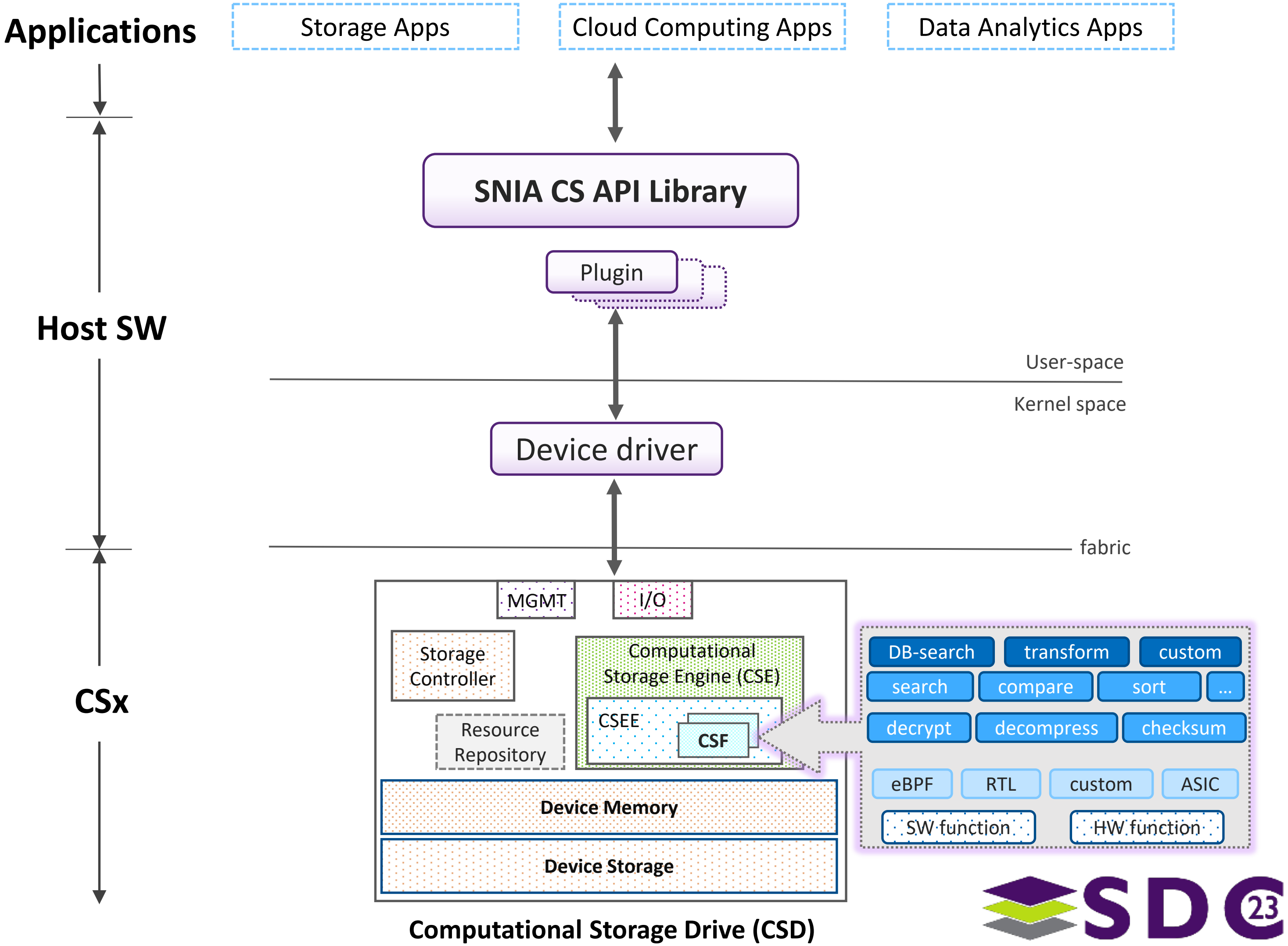
CSEE - A Computational Storage Engine Environment is an operating environment space for the CSE.

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.

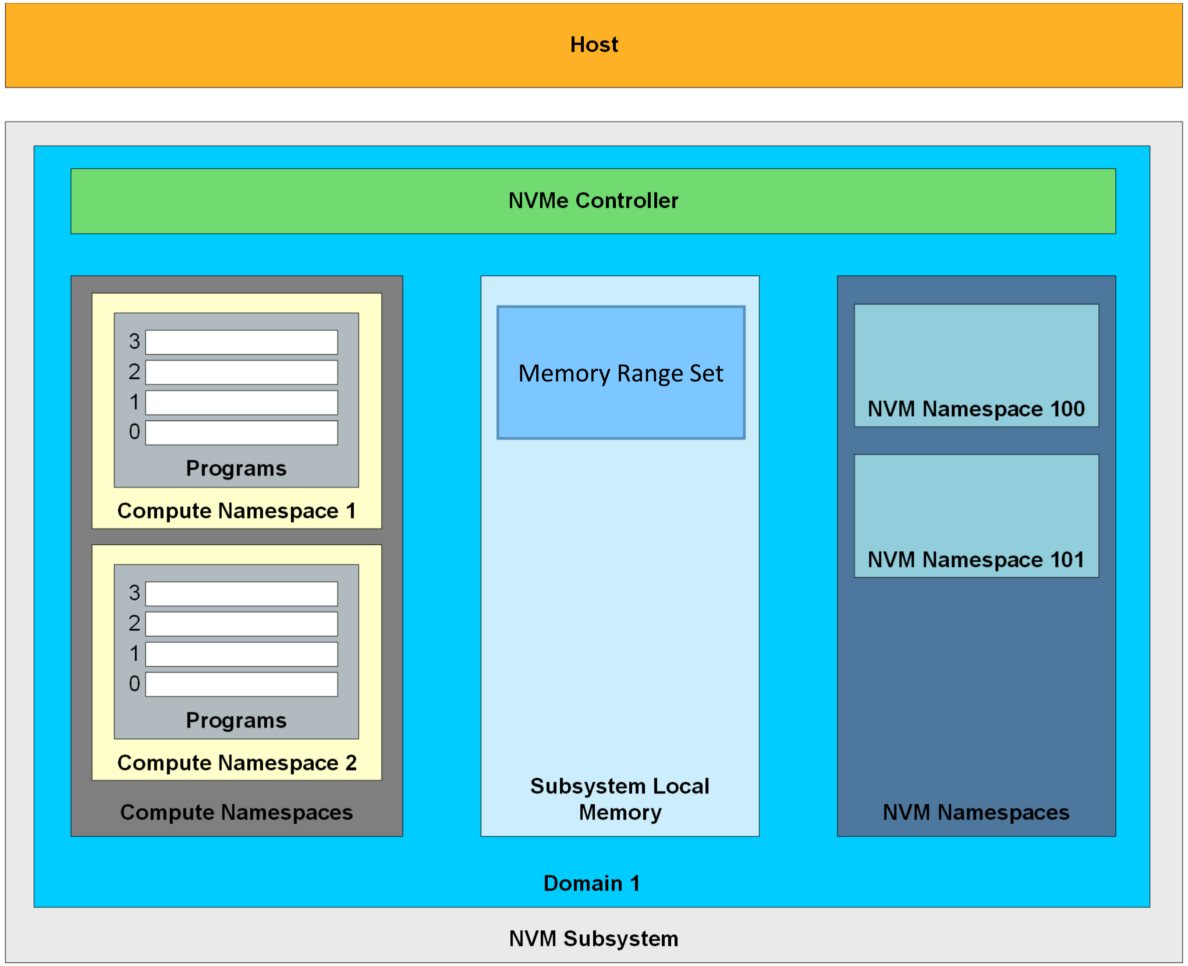
AFDM - Allocated Function Data Memory is a portion of FDM that is allocated for one or more specific instances of a CSF operation.

SNIA Computational Storage APIs

- One set of APIs for all CSx types
- APIs hide device details
  - Hardware, Connectivity
- Abstracts device details
  - Discovery
  - Access
  - Device Management
  - Memory Management
    - alloc/free/init
  - Storage/Memory Access
  - Download
  - Execute CSFs
- APIs are OS agnostic



NVMe Computational Storage Architectural Components



- Compute Namespaces
  - Compute Engines
  - Programs
- Programs operate on data in Subsystem Local Memory
  - Allocated as Memory Range Set
  - Includes program input, output
- NVM Namespaces
  - Persistent storage of data
  - NVM
  - ZNS
  - KV
- Data is transferred between NVM Namespaces and SLM using a copy command

This presentation discusses NVMe work in progress, which is subject to change without notice.

Correlation of SNIA/NVMe terms

SNIA Terms	NVMe Terms
Computational Storage Engine	Compute Engine/Compute Namespace
Computational Storage Engine Environment	Virtual (Not currently defined)
Resource Repository	Programs
Downloaded CSF and CSEE	Downloaded programs
Pre-loaded CSF and CSEE	Device-defined programs
Activation	Activation
Function Data Memory (FDM)	Subsystem Local Memory (SLM)
Allocated FDM (AFDM)	Memory Range Set
Device Storage	NVM Namespaces

SDXI+CS Combination

Why Combine SDXI and Computational Storage

- Computational Storage reduces host data movement. Why do we need a data mover?
  - Peer-to-peer data movement offloads the host but requires a data mover
    - Computation happens where the appropriate compute engine resides
  - SDXI reduces the amount of data movement within the host software stack

- SDXI is a memory to memory data mover. Computational Storage computes on a storage device. Why combine these technologies?
  - Memory is everywhere
  - Data in use is where the computation happens
    - Data may be in the host or a Computational Storage Device – SDXI bridges the two worlds
    - SDXI transformations can provide the compute in Computational Storage Device

- Computation may be required as the data is moved
  - SDXI can provide transformations as data is moved in and out of Computational Storage

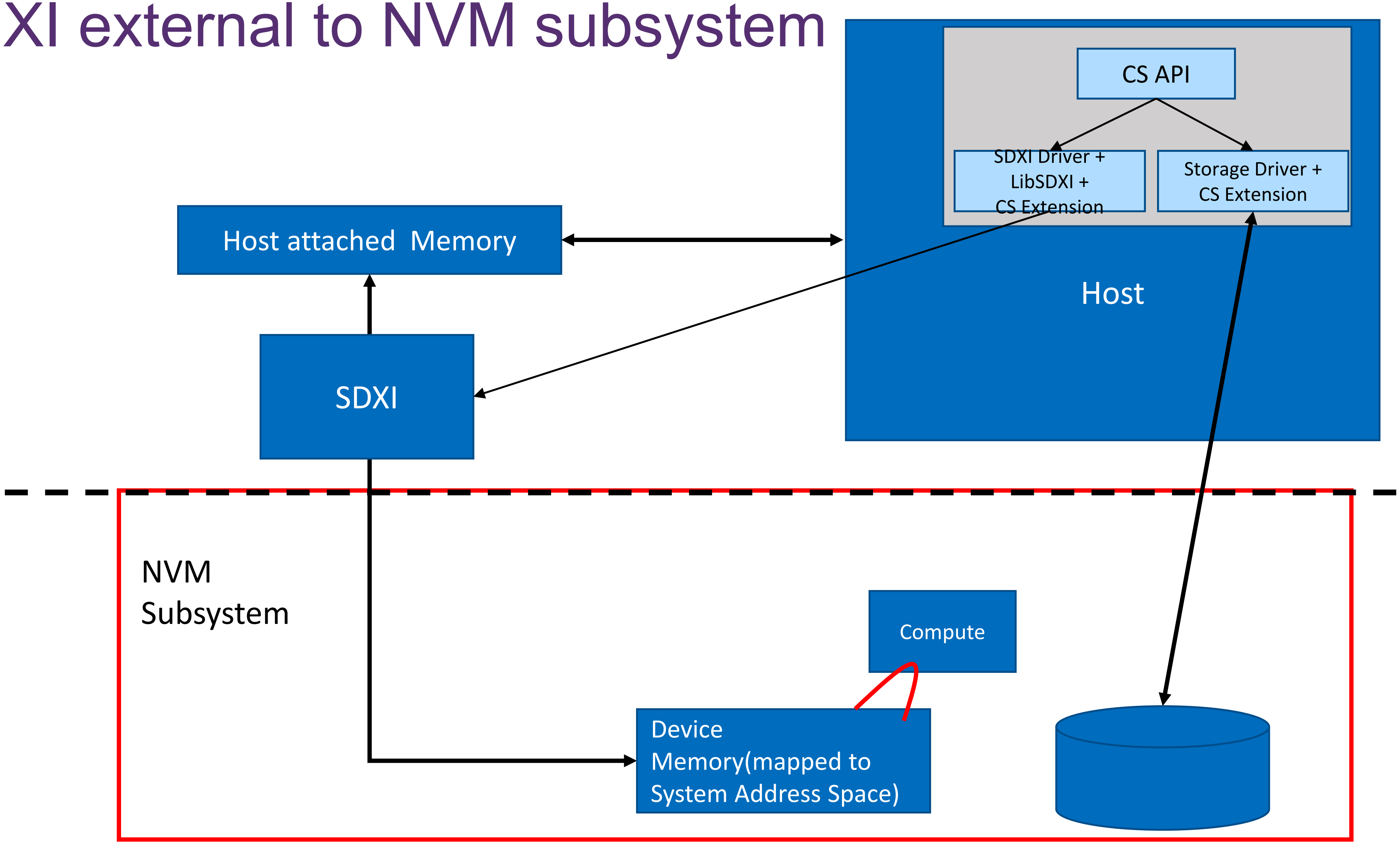
SNIA SDXI+CS Subgroup

- What is the subgroup
  - The CS TWG and SDXI TWG collaboration
  - Develop a unified block diagram that imagines a combined CS and SDXI system and architecture
  - Develops use cases for SDXI-based CS devices
  - Membership to the subgroup requires membership to both the CS TWG and SDXI TWG
- Member companies
  - AMD, Dell, HPE, Marvell, Samsung, Solidigm, Huawei, Micron, Western Digital, ARM, StorageX Technology, Pliops

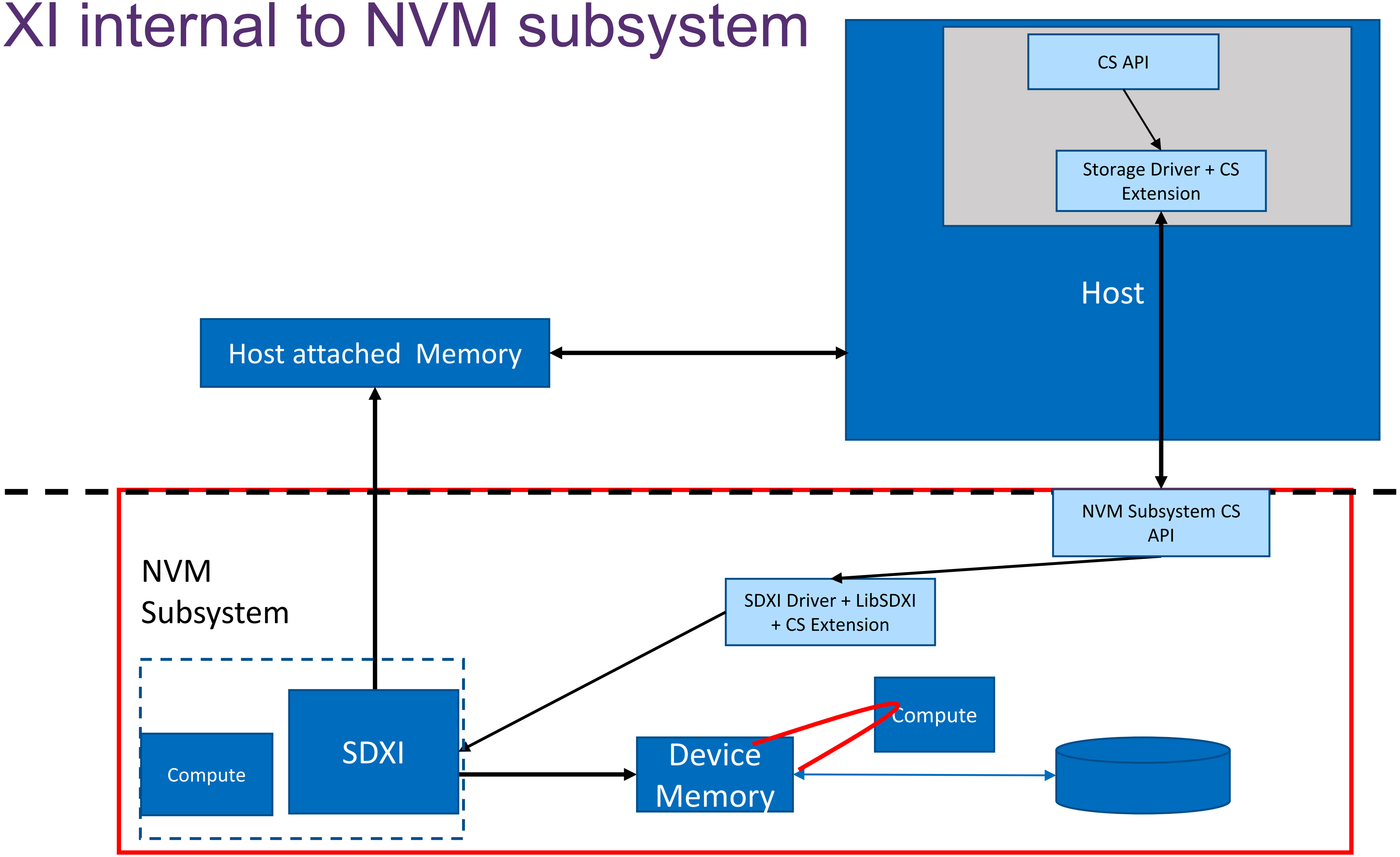
CS + SDXI – Combining with NVMe

- SDXI is a data mover engine (standardizes memory to memory DMA)
  - SDXI is fabric agnostic – could be PCIe, CXL, or something else
  - Spec includes an example usage with PCIe
- Data movement through DMA is an integral part of NVMe
- SDXI interacts with host addressable memory
  - NVMe device private memory is not mapped to system address space
    - Private device memory is not accessible by SDXI instance external to the device
    - SDXI instance within an NVM Subsystem could enable data movement between private device memory and host addressable memory or between different private device memory address ranges
  - CMB/PMR regions mapped to the system address space can be accessed by SDXI instances external to the NVM Subsystem

SDXI external to NVM subsystem

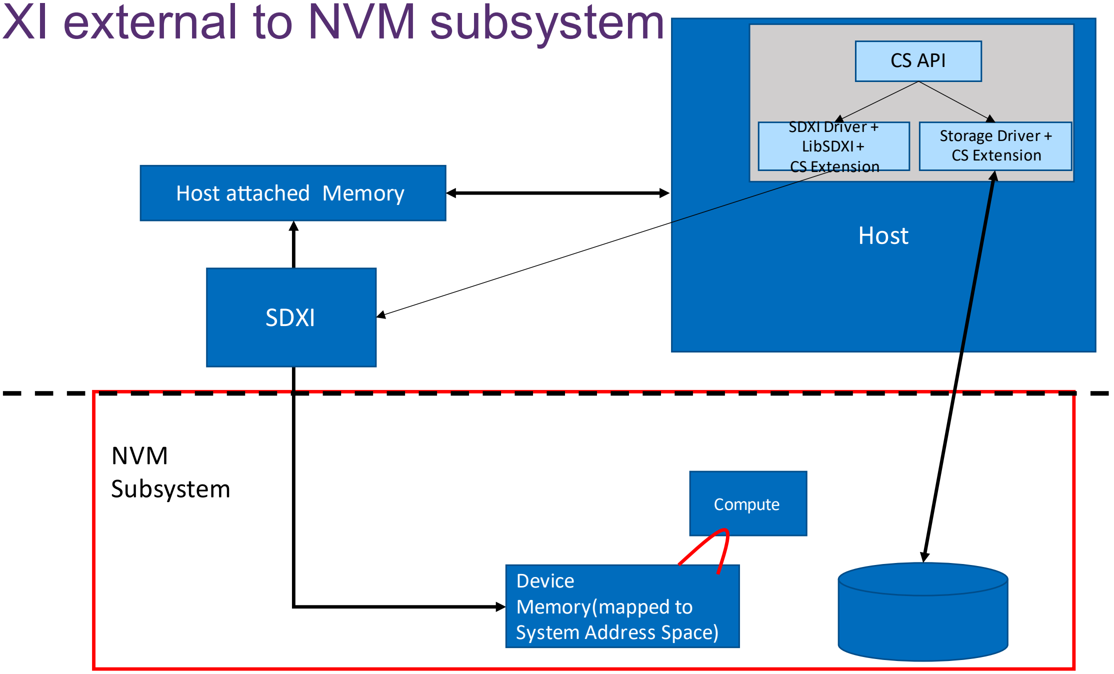

CS API
SDXI Driver + LibSDXI + CS Extension
Storage Driver + CS Extension
Host
Host attached Memory
SDXI
NVM Subsystem
Compute
Device Memory(mapped to System Address Space)

SDXI internal to NVM subsystem



SDXI and NVM Subsystem

SDXI external to NVM subsystem



CS API

SDXI Driver + LibSDXI + CS Extension

Storage Driver + CS Extension

Host

Host attached Memory

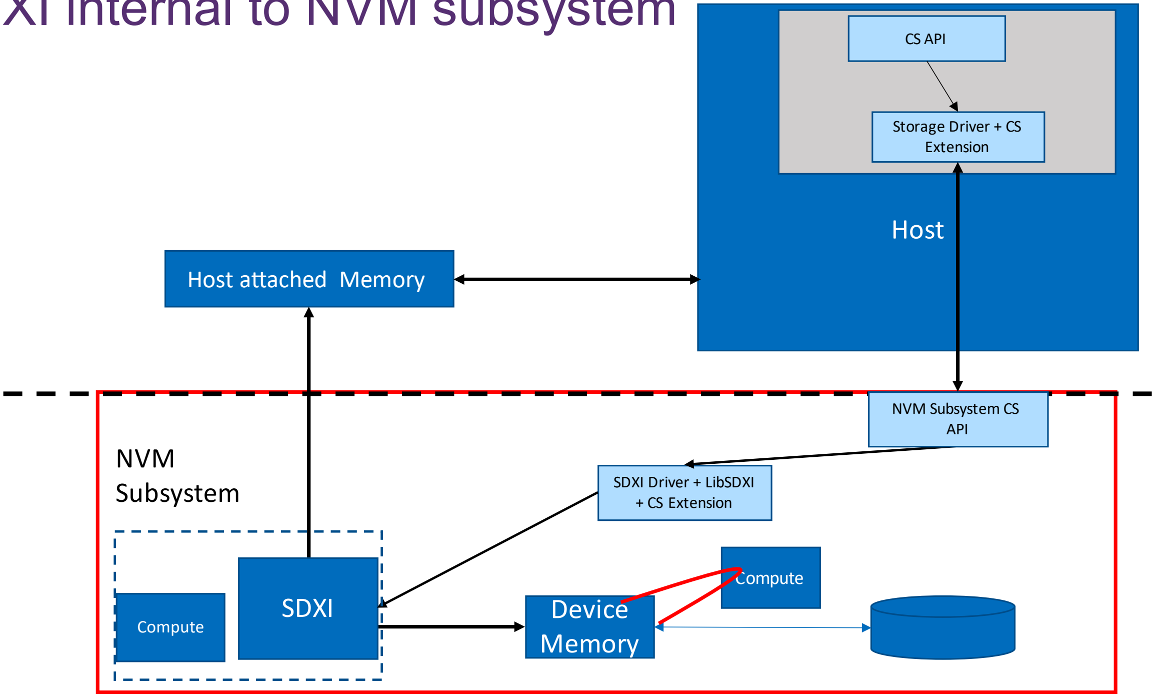
SDXI

NVM Subsystem

Compute

Device Memory(mapped to System Address Space)

SDXI internal to NVM subsystem



CS API

Storage Driver + CS Extension

Host

Host attached Memory

NVM Subsystem CS API

SDXI Driver + LibSDXI + CS Extension

NVM Subsystem

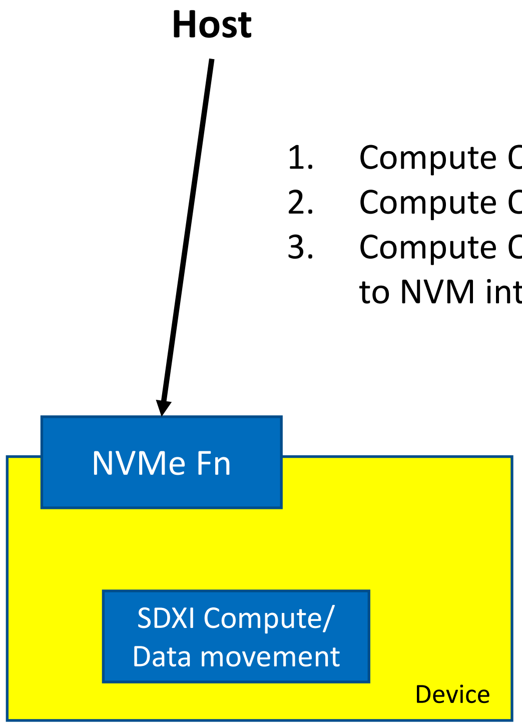
Compute

SDXI

Device Memory

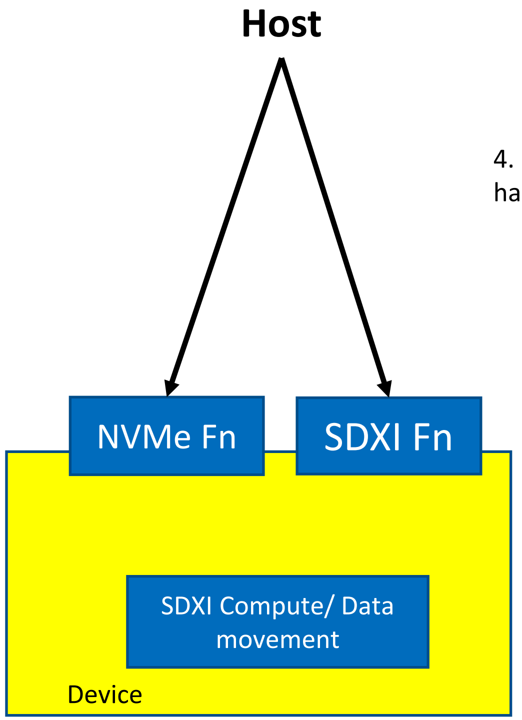
Compute

Device Types



1. Compute Option 1: NVMe + SDXI command
2. Compute Option 2: NVMe computation command
3. Compute Option 3: SDXI operation is transparent to NVM interface

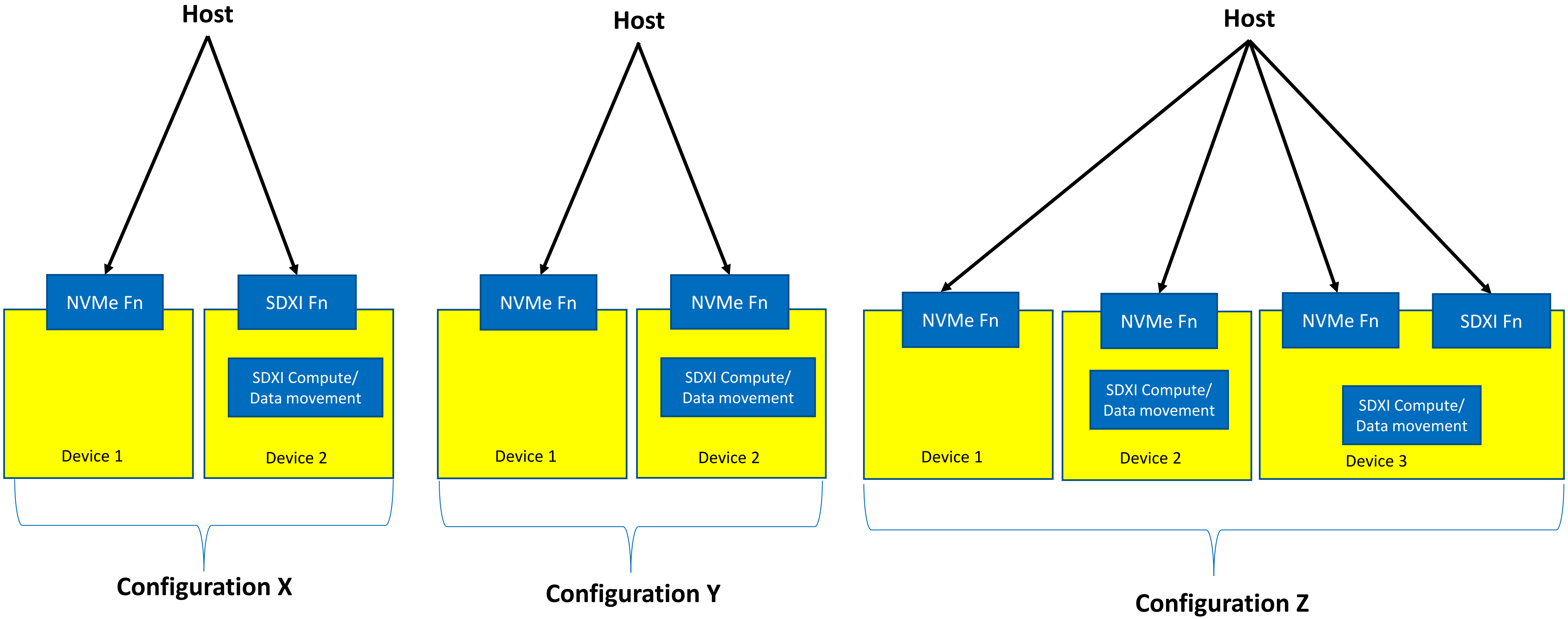
Type A Device



4. Computation Option 4: Device has NVMe fn and SDXI fn interface

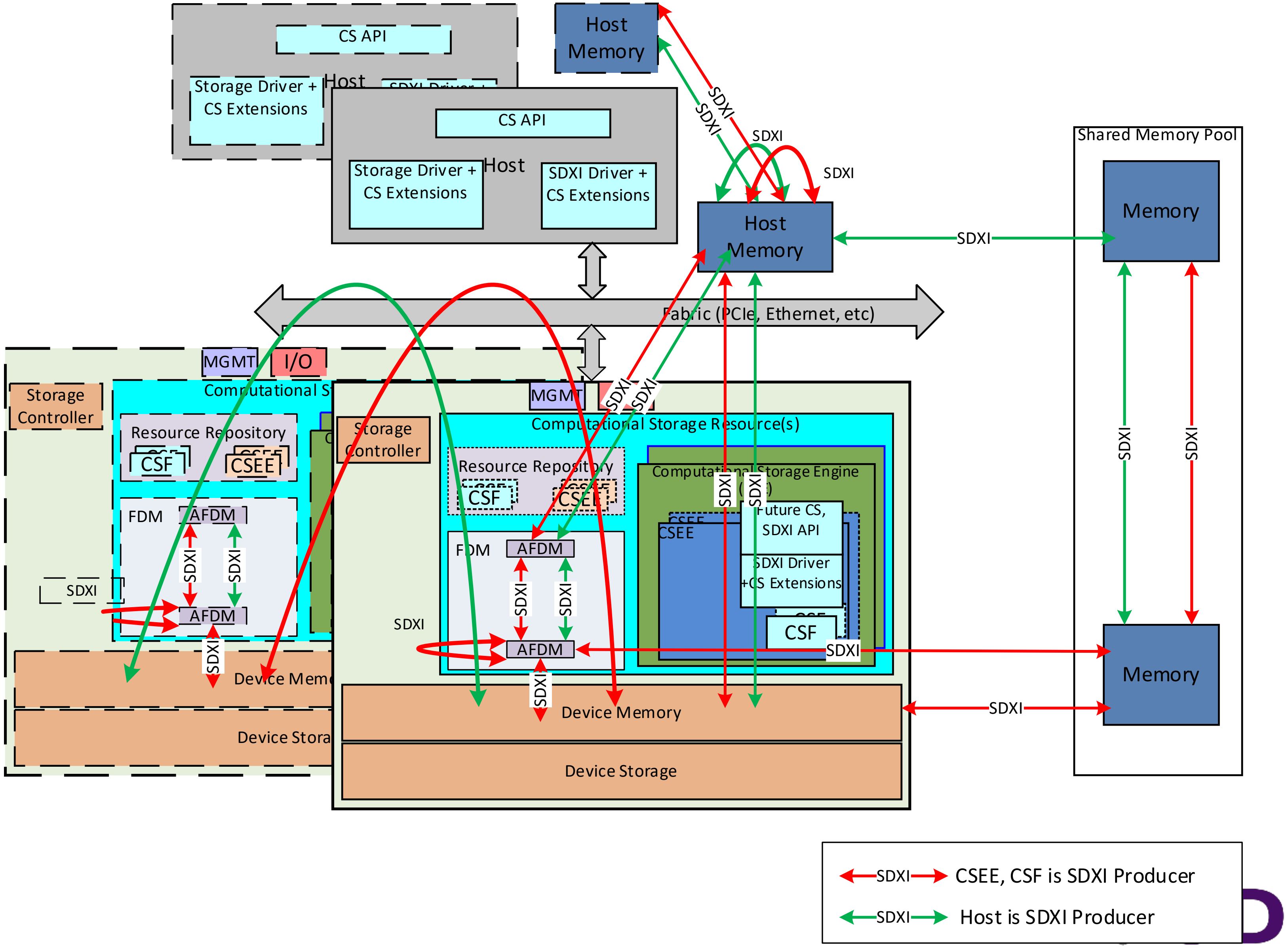
Type B Device

Configurations



Combined SDXI+CS Architecture

- Multiple SDXI producers in a CS Architecture
- SDXI enables data movement across multiple AFDM regions
- LibSDXI/SDXI Driver efforts compliment CS API for memory to memory data movement

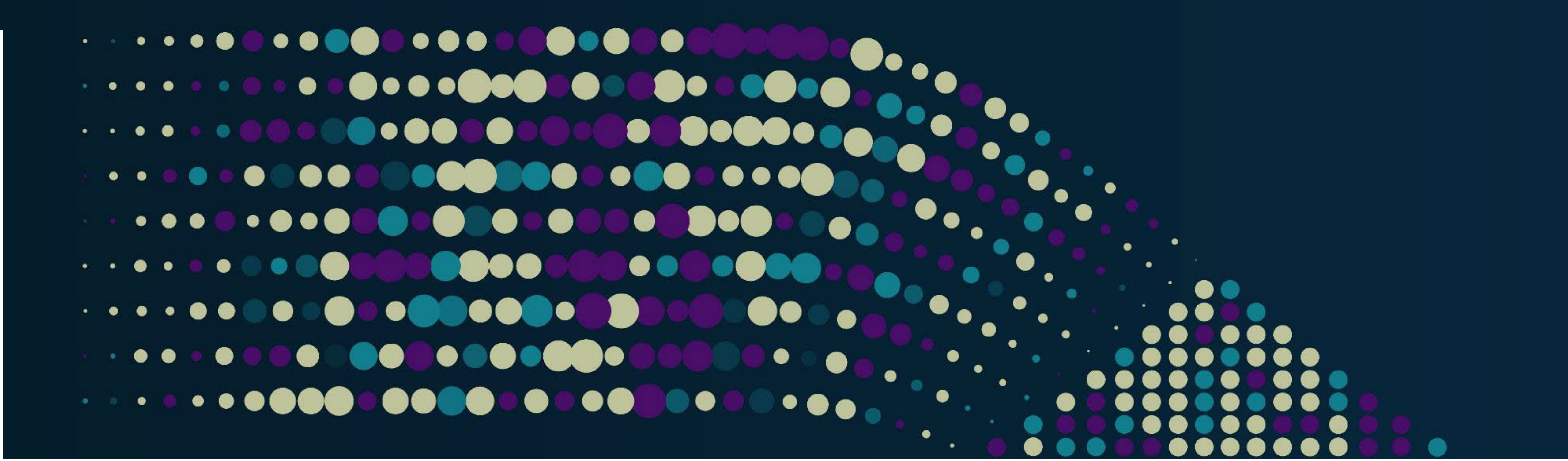


CS API
Host
Storage Driver + CS Extensions
SDXI Driver + CS Extensions
Host Memory
SDXI
Shared Memory Pool
Memory
Fabric (PCIe, Ethernet, etc)
MGMT
I/O
Storage Controller
Computational Storage Resource(s)
Resource Repository
CSF
CSEE
FDM
AFDM
Computational Storage Engine (CSE)
Future CS, SDXI API
SDXI Driver +CS Extensions
Device Memory
Device Storage
SDXI CSEE, CSF is SDXI Producer
SDXI Host is SDXI Producer

Summary and Call to Action

- SNIA is developing SDXI and Computational Storage standards
  - This presentation describes the collaboration between these two work groups
- Computational Storage benefits from SDXI enabled architectures
  - SDXI provides a standard method of moving data to the compute
  - SDXI transformations assist with compute associated with Computational Storage data flows
- Call to Action:
  - Join SDXI, Computational Storage, and the collaboration efforts in SNIA
  - Learn More:
    - https://www.snia.org/sdxi
    - https://www.snia.org/computational

Q&A



Please take a moment to rate this session.

Your feedback is important to us.