SNIA DEVELOPER CONFERENCE

SDC 24

BY Developers FOR Developers

September 16-18, 2024
Santa Clara, CA

Smart Data Accelerator Interface Use Cases Proof Points v1.1 and beyond

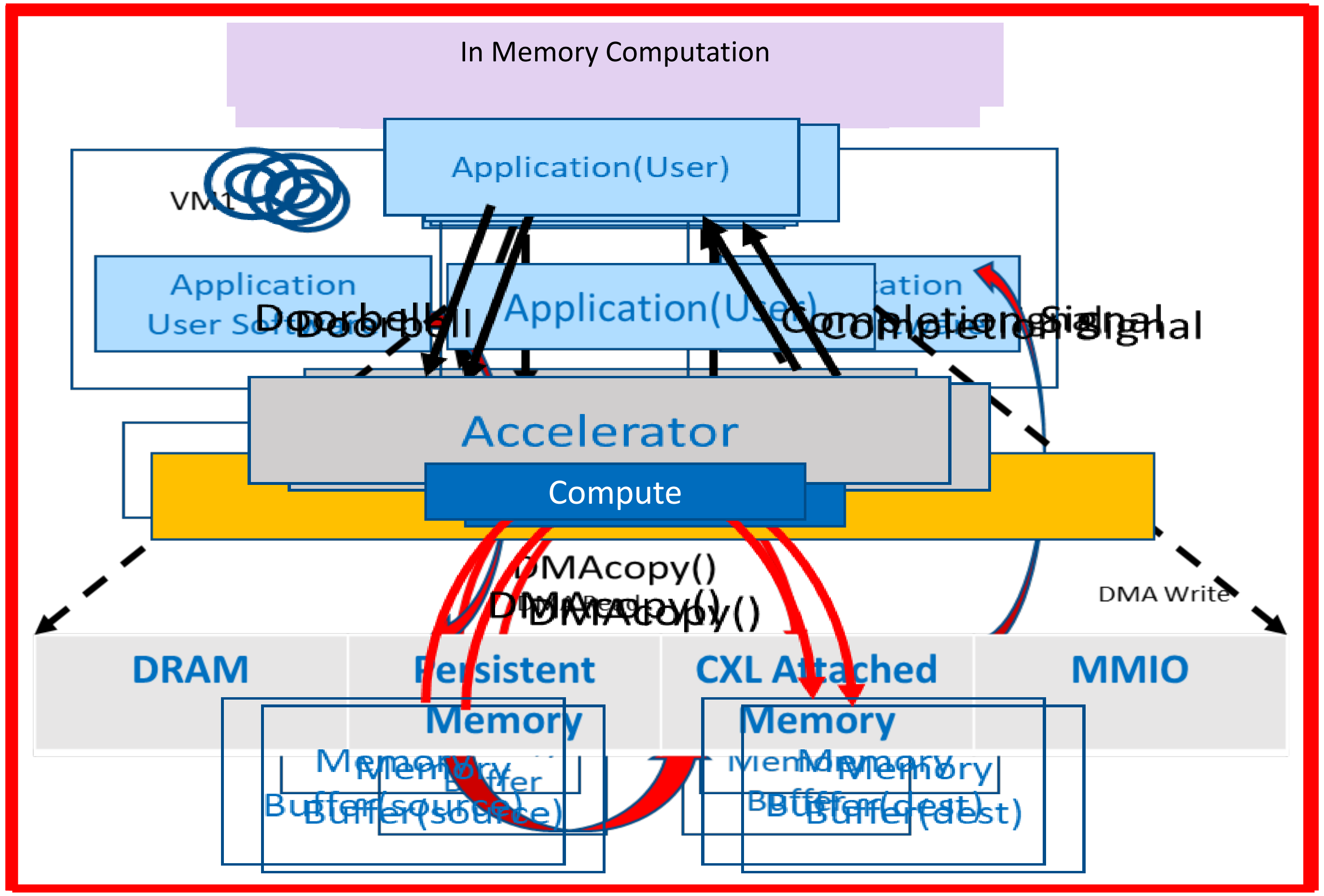
Shyam Iyer
Chair, SNIA SDXI TWG
Member, SNIA Technical Council
Distinguished Engineer, Dell

Agenda

- SDXI Intro and brief overview of v1.0
- SDXI v1.1 preview
- Software Enablement
- Proof points
- Summary

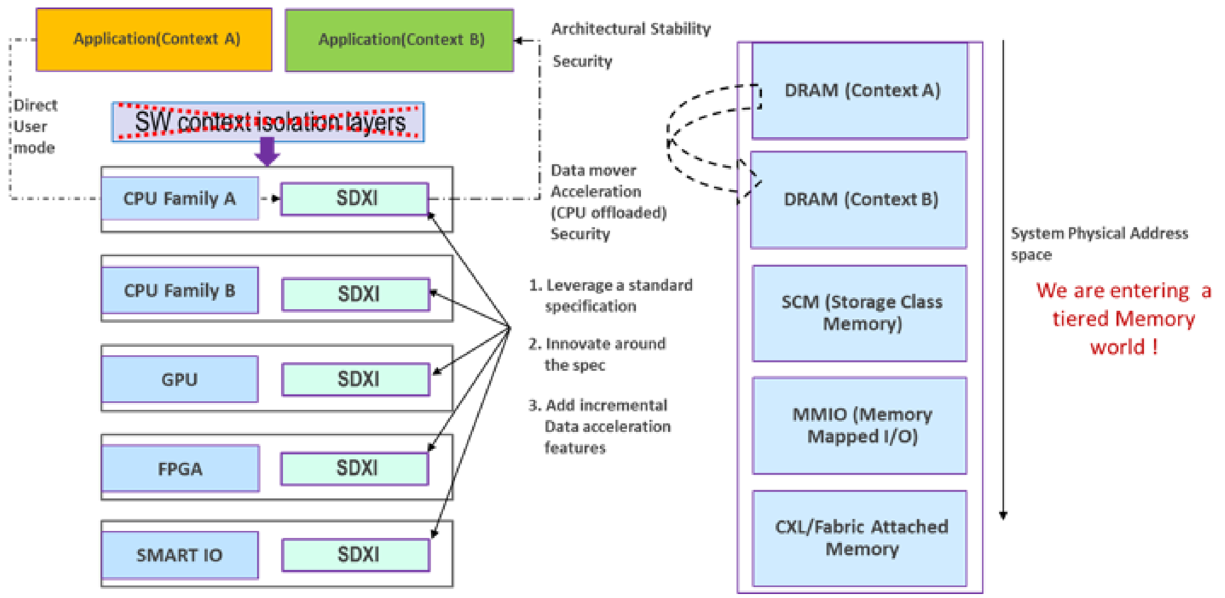
SDXI Intro and brief overview of v1.0

Sample accelerator usage models

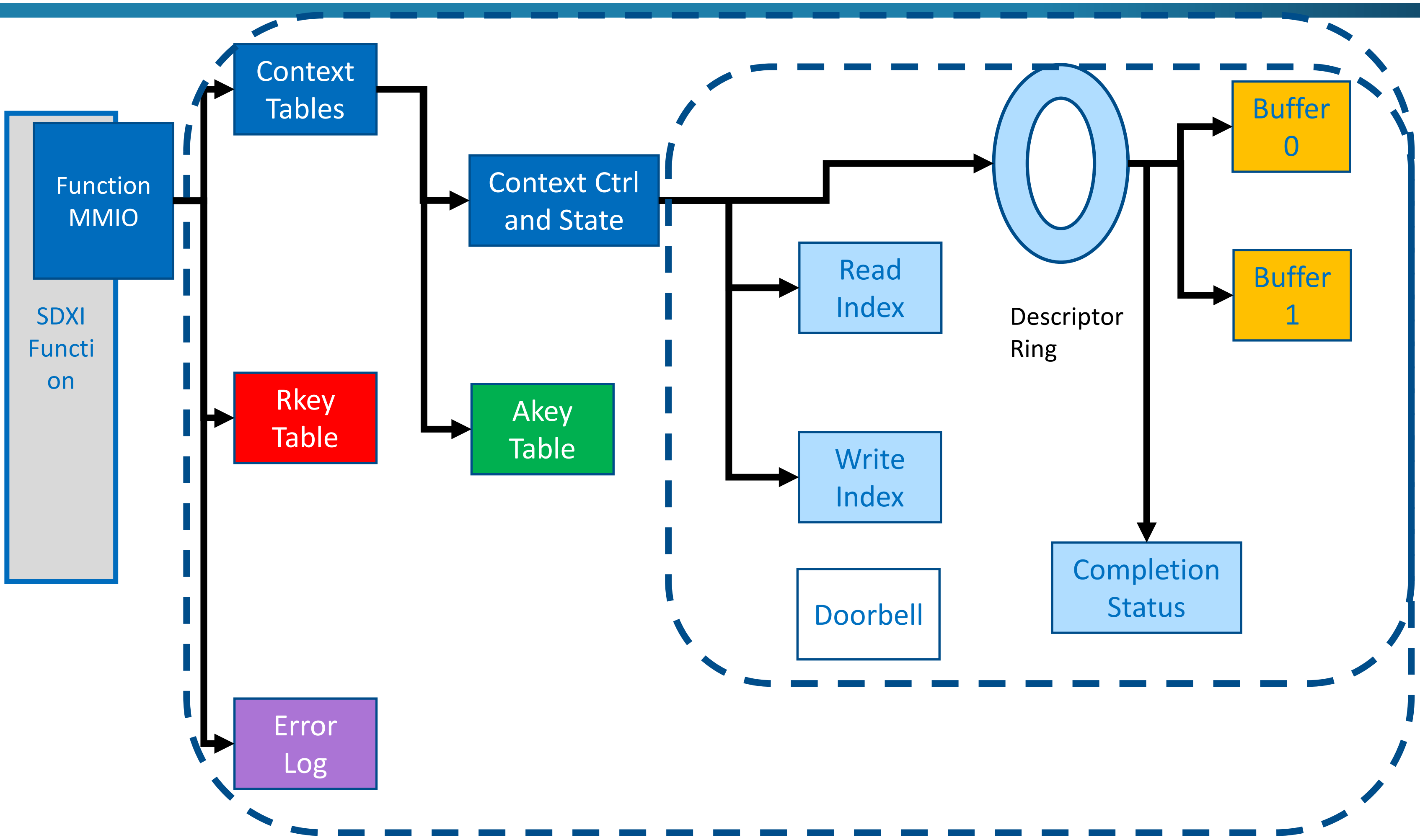


SDXI Intro

- 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
  - Features:
    - Virtualized address space to address space data movement
    - Offloads data movement, common memory operations, and data transformations while moving data
    - Offloads data movement while preserving address space and context isolation.
    - Standardized interfaces and architected states for DMA engine
    - Standardized for user-level software.
- v1.0 released!
  - https://www.snia.org/sdxi
- SNIA's SDXI TWG is now working on v1.1 now
  - SDXI TWG also has a software focused group that is working on a reference libsdxi implementation

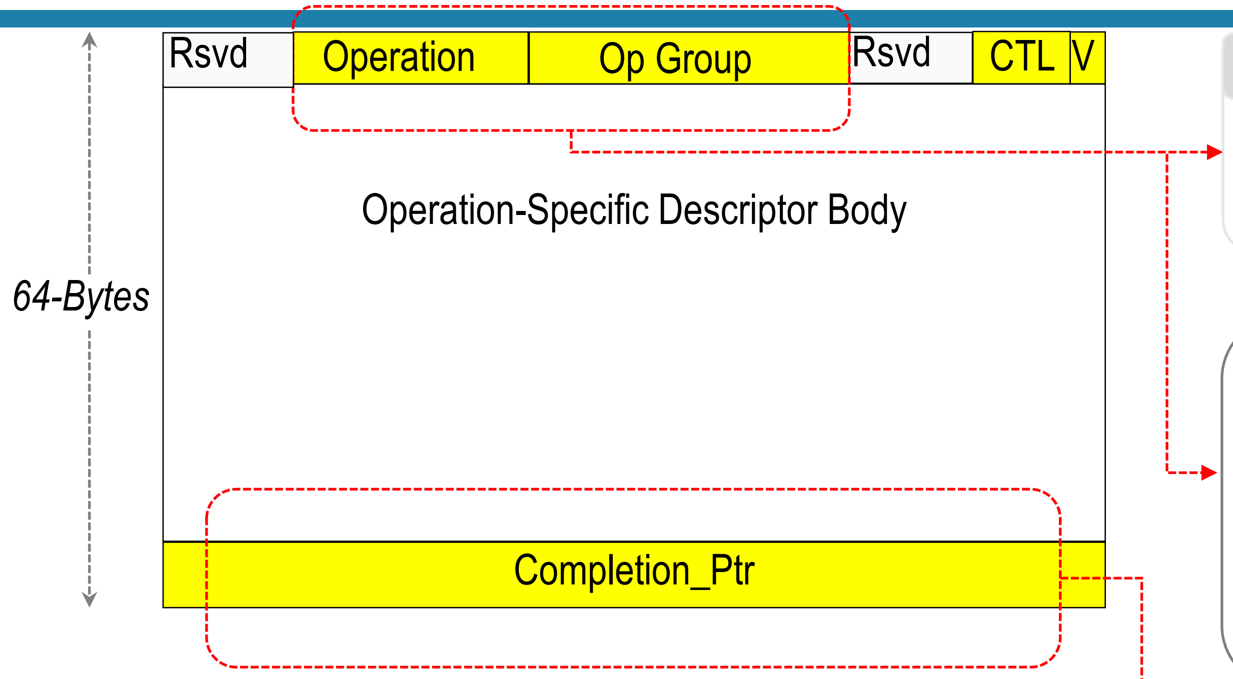

Application(Context A)
Application(Context B)
Architectural Stability
Security
Direct User mode
SW context isolation layers
CPU Family A
SDXI
CPU Family B
SDXI
GPU
SDXI
FPGA
SDXI
SMART IO
SDXI
Data mover Acceleration (CPU offloaded) Security
1. Leverage a standard specification
2. Innovate around the 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 !

Memory Structures(1) – Simplified view



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

A Standard Descriptor Format (1)



Architecturally Registered Operation Groups:

DMA Base	Administrative	Vendor-Defined
Full Atomic	Minimal Atomic	Others …

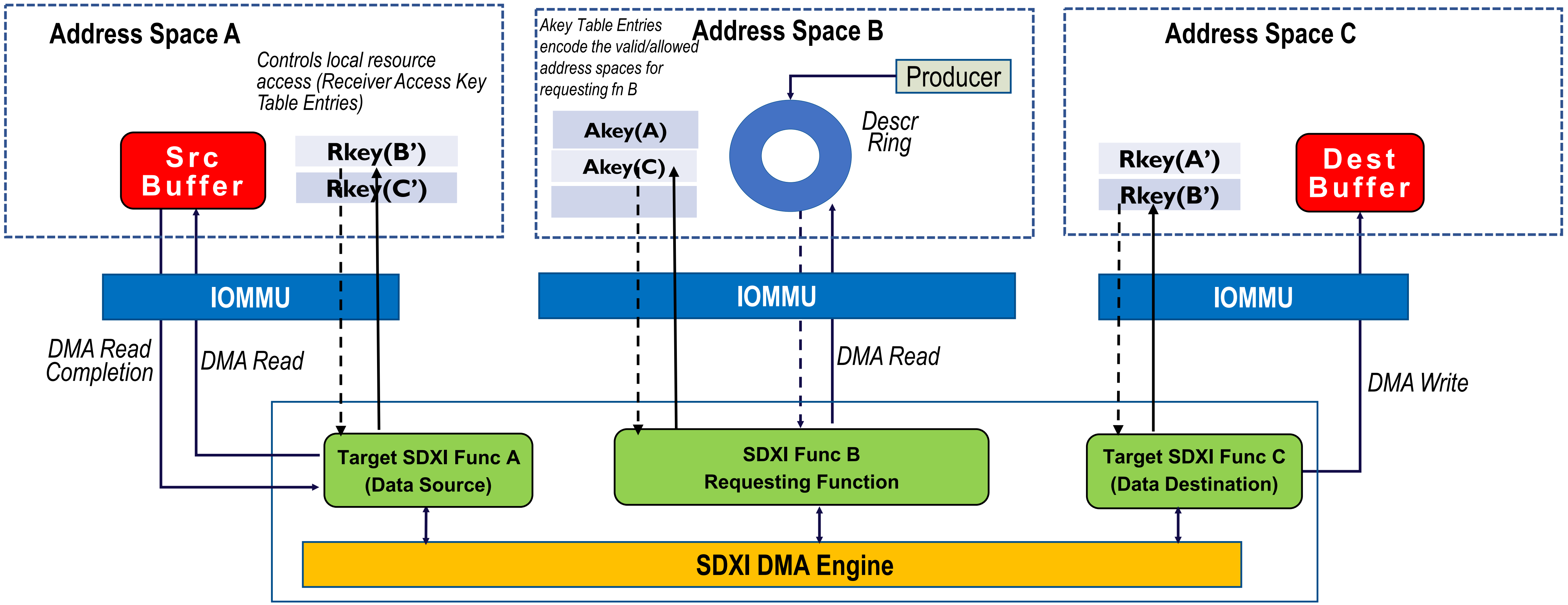
DMA:	Nop, Copy, RepCopy, WriteImm
Atomic:	Bitwise Ops, Add(minimal), Sub, Swap(minimal), Min, Max, CmpSwap(minimal), etc
Admin:	Start/Stop/Update/Sync, Interrupt Function & Contexts (easily virtualizable)

A pointer to a 32-byte aligned region of memory containing the Completion Status Block that contains

- Completion Signal
  - Initialized by SW, Decremented by Function on Success
- Error Bit(ER) to indicate the operation encountered an error
- Other bits in the 32-byte field are reserved to support future expansion of error codes

*Room for lots of future operations

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')
Rkey(C')
Akey Table Entries encode the valid/allowed address spaces for requesting fn B
Address Space B
Producer
Akey(A)
Akey(C)
Descr Ring
Address Space C
Rkey(A')
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

Need more on SDXI Internals

- SNIA SDXI Specification v1.0 Internals
  - https://www.youtube.com/watch?v=wjc4ZnCQibw&pp=ygUNc2RjIDIwMjMgc2R4aQ%3D%3D

SDXI v1.1 Preview

SDXI v1.1 investigations

- Connection manager
- New data mover operations for smart acceleration
- SDXI Host to Host investigations
- Scalability & Latency improv
- Cache coherency mod ta movers
- Security Featur g data movers
- Data mov ons involving persistent memory targets
- QoS
- ated use cases
- eterogenous environments

SDXI v1.1 Update

Draft: Subject to Change

SDXI v1.1, v1.2, and v2.0

- While investigating features for v1.1 SDXI TWG developed a framework for features:
  - v1.1
    - Mostly errata fixes from v1.0,
    - Additional use cases prioritized by member participation
    - Retains compatibility with v1.0
  - v1.2
    - Overflow from v1.2
    - Retains compatibility with v1.0, and v1.1
  - v2.0
    - More intrusive features

v1.1 Sneak Peak

- SDXI v1.1 Practical Considerations
  - Definable Operations to enable innovation
  - Define new data mover operations to enable critical member use cases
  - Improvements around memory ordering
  - Improved point of view for
    - Connection Manager
    - Use cases involving memory fabrics,
    - Host to Host use cases
    - QoS use case
    - Storage Use Cases involving NVMe, and Computational Storage
    - Security considerations
    - AI Use cases

v1.1 Candidate: Definable Operations Group

- v1.0 Vendor-defined operations group definition was rigid
- Required vendors to register a vendor opcode to avoid collisions
  - Slows innovation
- Innovators want flexibility in defining new operations
  - However, they require leverage with:
    - Software, and APIs without rewriting infrastructure code
- Definable operations group to the rescue!
  - Requires new UUID for definable operations in vendor space
  - Each vendor can support a profile to enable its own set of definable operations
- Are you using the v1.0 vendor-defined encodings?
  - Expect this v1.0 feature to get deprecated with v1.1

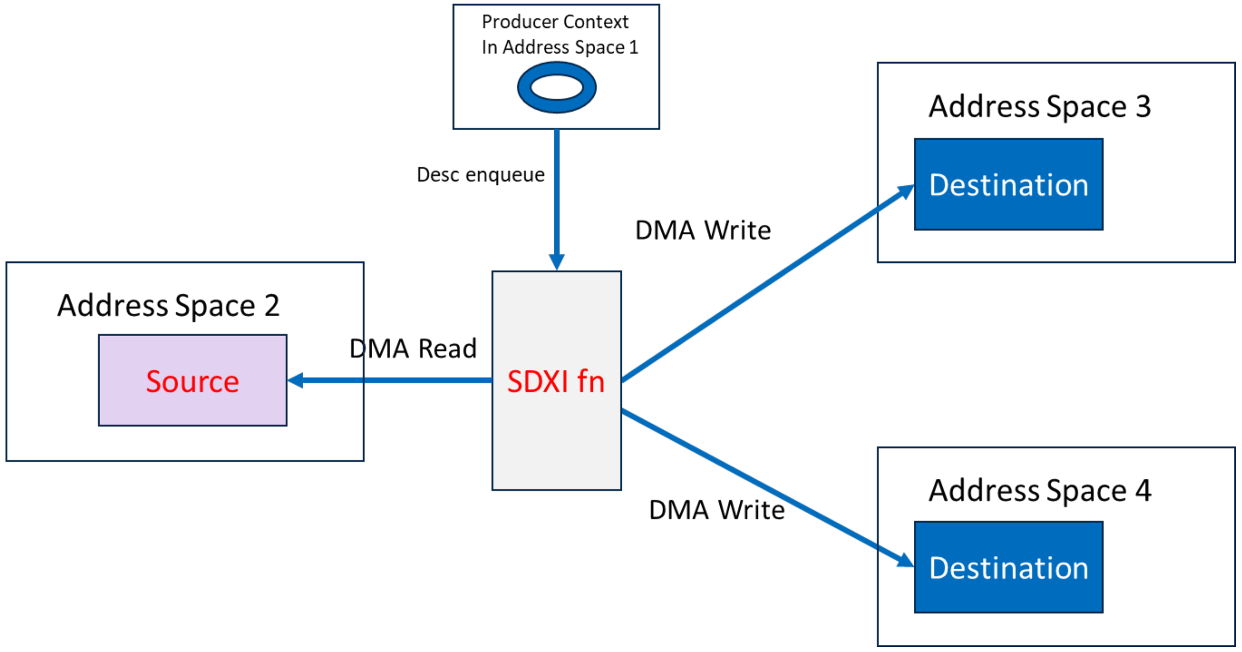
While we are on the topic of deprecation:

- Potential candidates for deprecation from v1.0
  - Mailbox
  - Vendor Defined Operations Group in favor of Definable Operations Group
- Are you affected?
  - Please yell or join the workgroup!

v1.1 Candidate: Make me another copy!

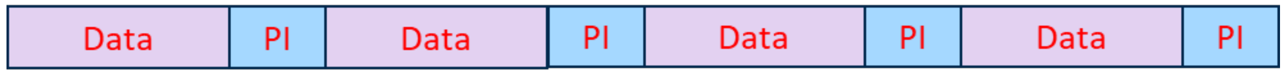
Double Copy

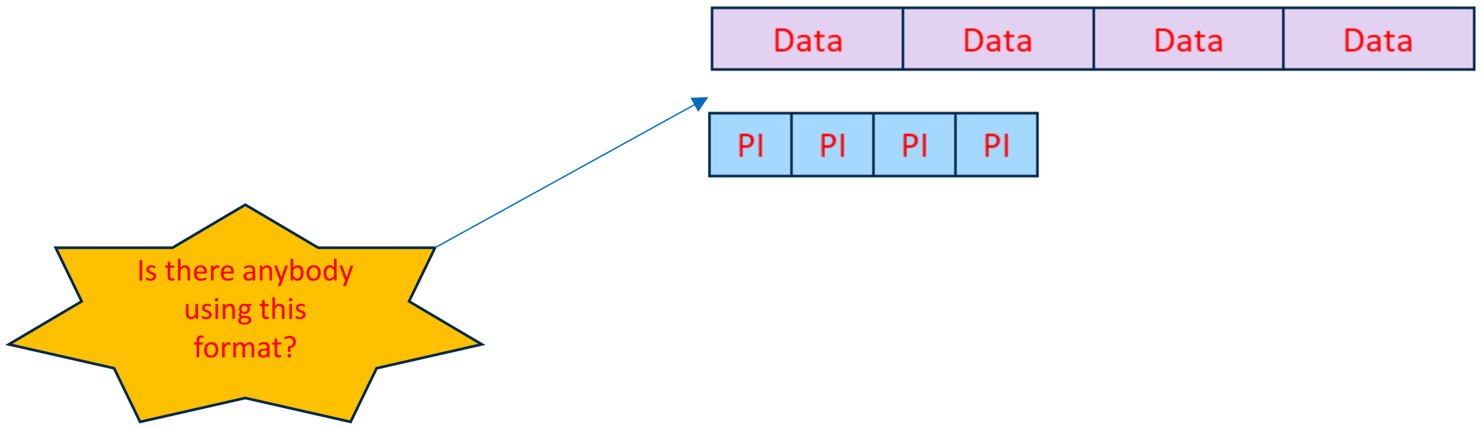
- Single Source buffer two destination buffers
- Single Source buffer, and two destination buffers
    - Each buffer can be in different address spaces.
    - Producer context can also be in an independent address space.



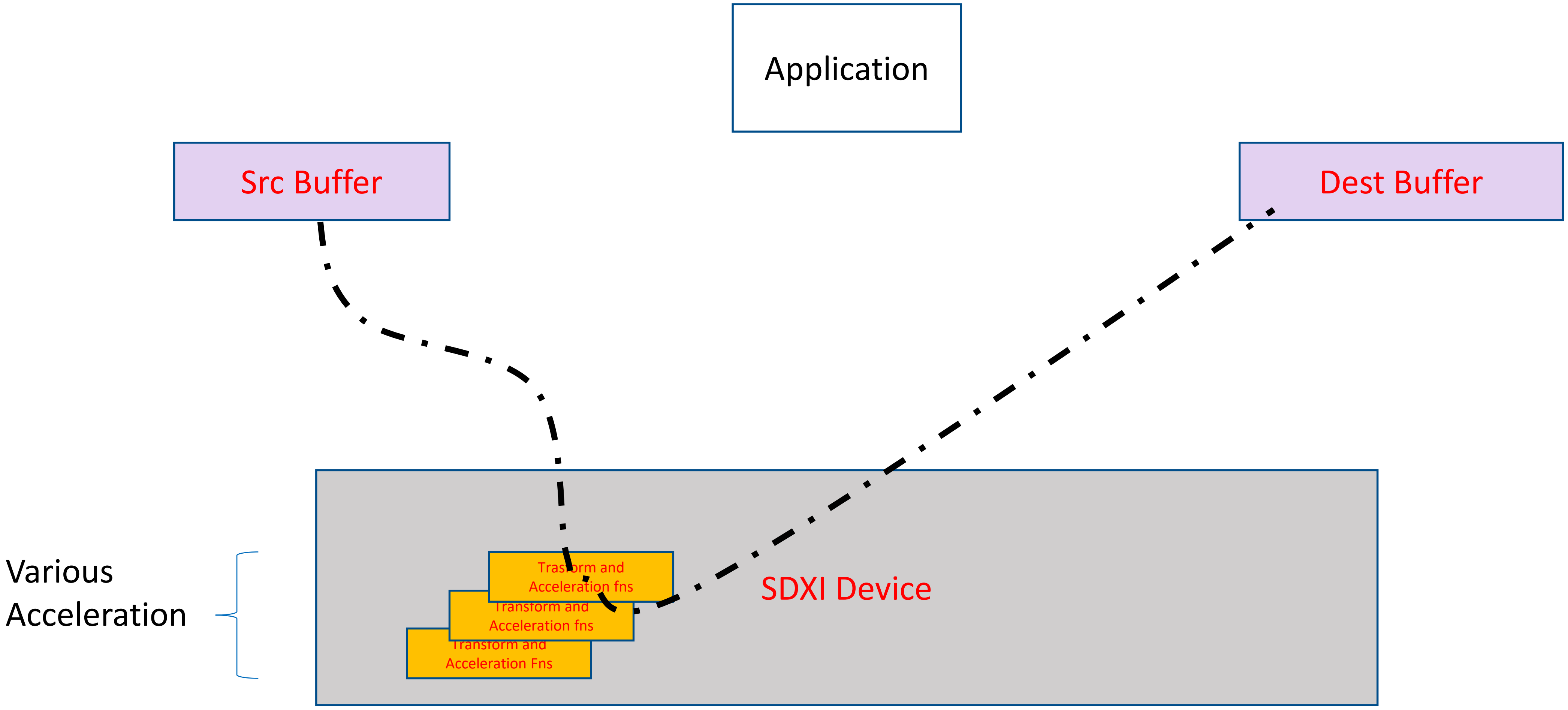
v1.1 Candidate: Data Integrity

- Cyclic Redundancy Checks(CRC)
- Protection Information(PI)
  - Memory to memory with PI Check, Strip, Insert, Update, Compare, etc.





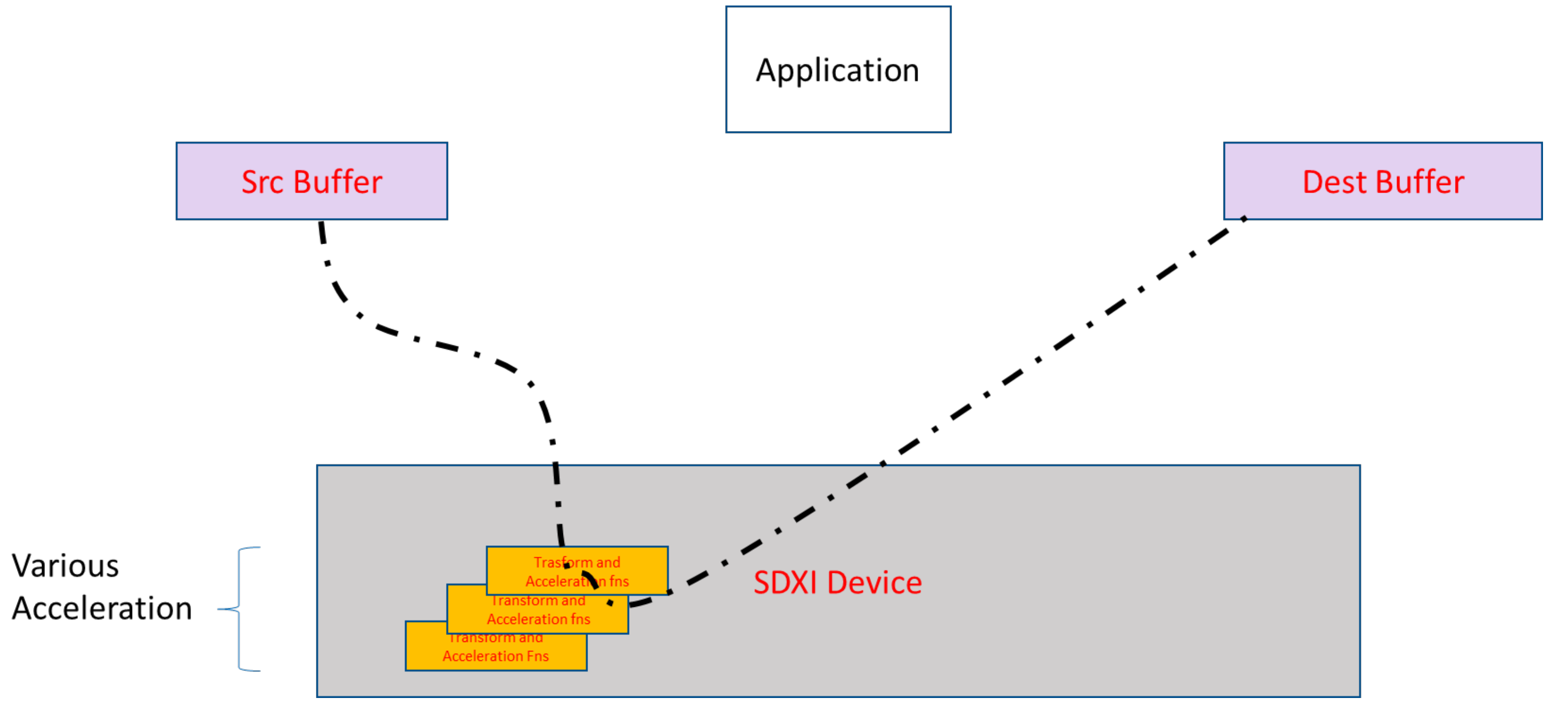
v1.1: New Data Mover Operations


Application
Src Buffer
Dest Buffer
Various Acceleration
Trasform and Acceleration fns
Transform and Acceleration fns
Transform and Acceleration Fns
SDXI Device

v1.1 Memory Operations and Data Transformations

Operations

- POSIX memory ops
- Compression
- Bring Your Own Operation(BYOO)

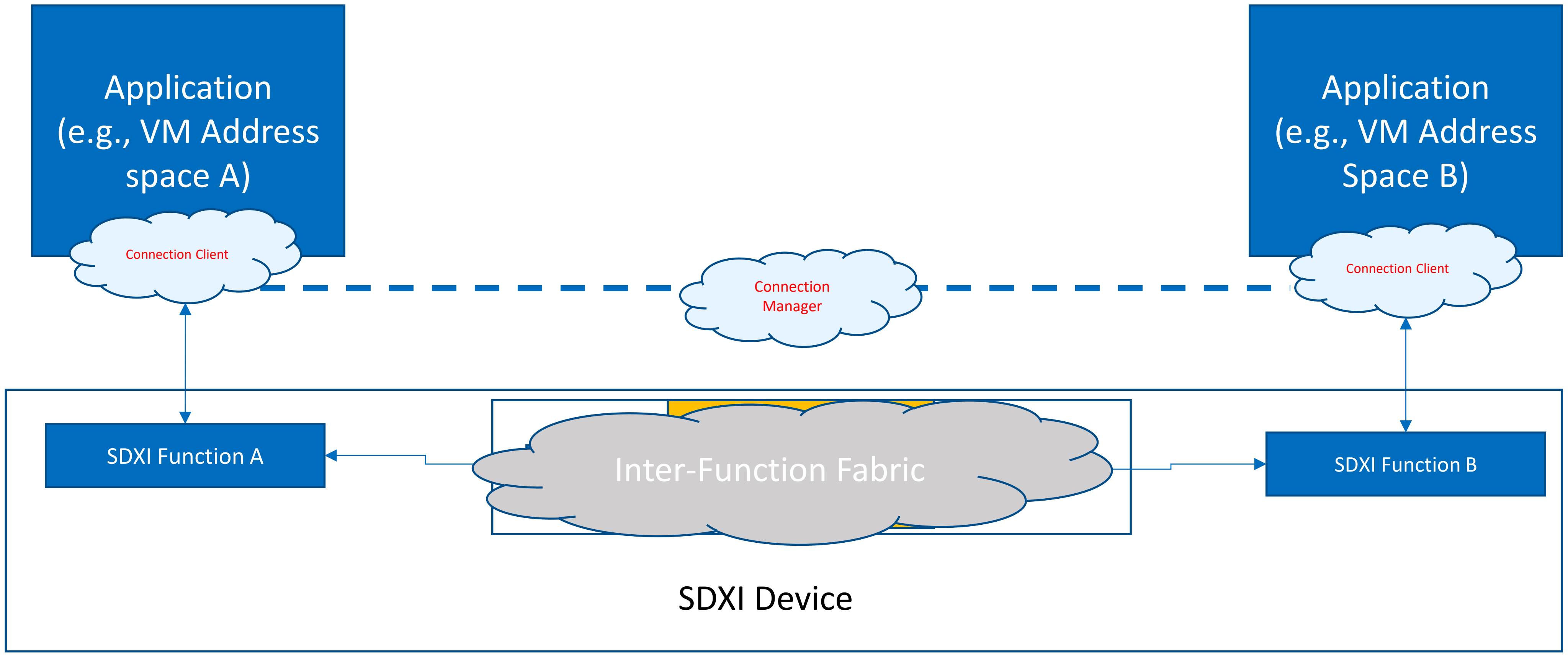


Application
Src Buffer
Dest Buffer
Various Acceleration
Trasform and Acceleration fns
Transform and Acceleration fns
Transform and Acceleration Fns
SDXI Device

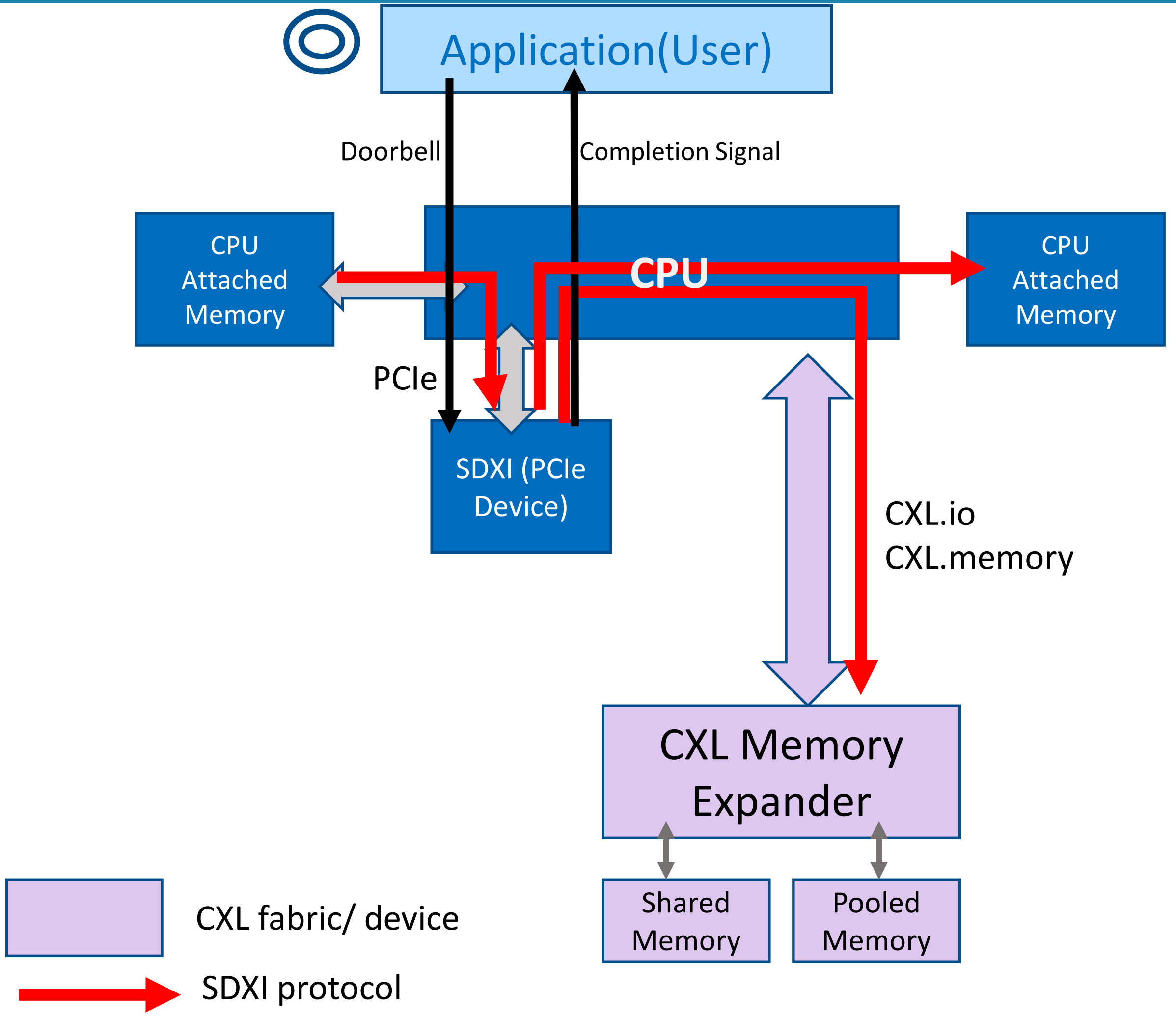
v1.1: Memory Ordering improvements

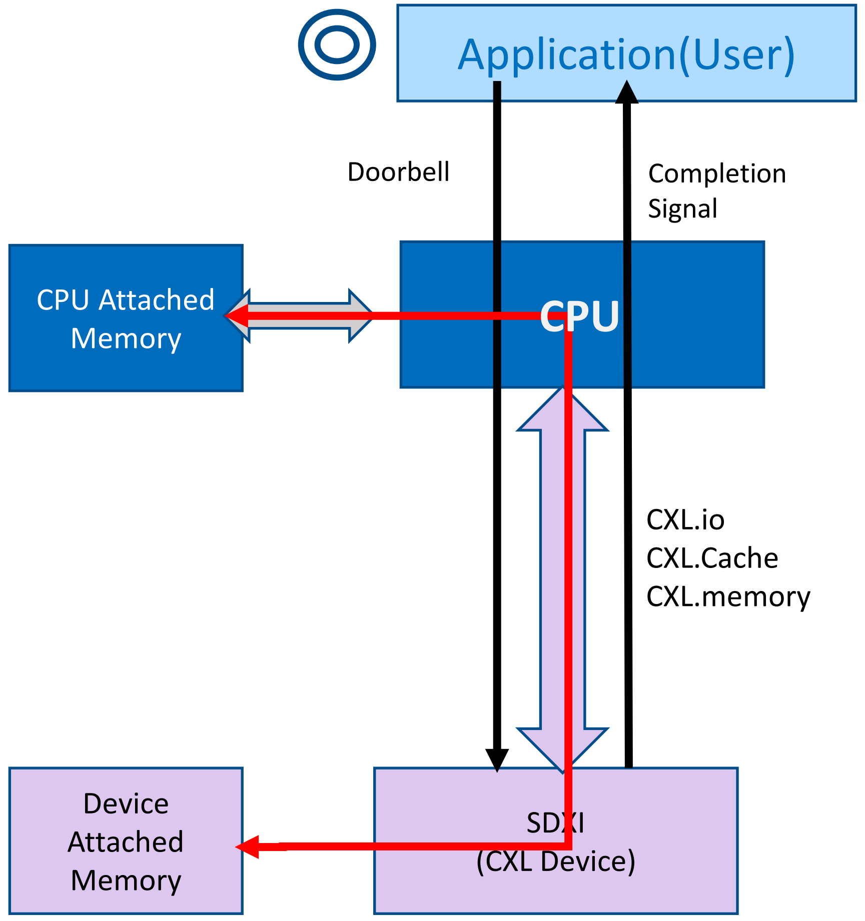
- SDXI v1.0 memory ordering
  - Write after Write – ‘seq’
  - Read after Completion of previous operation – ‘sync’
- v1.1 Memory ordering relaxations and clarifications
  - Read after Write
  - Valid bit checking
  - Flagged Write

Point of view: Connection Manager

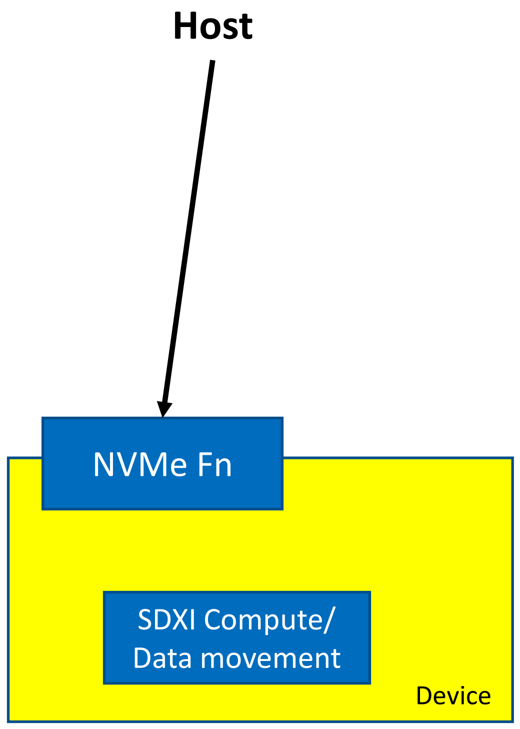


Point of view: CXL based Architectures

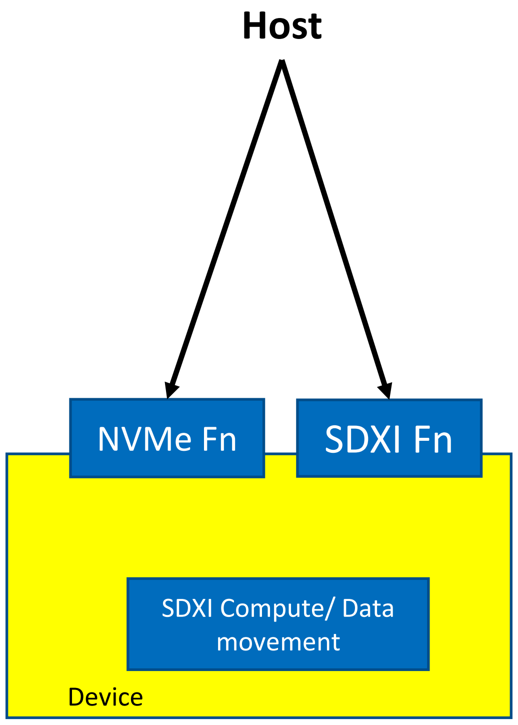




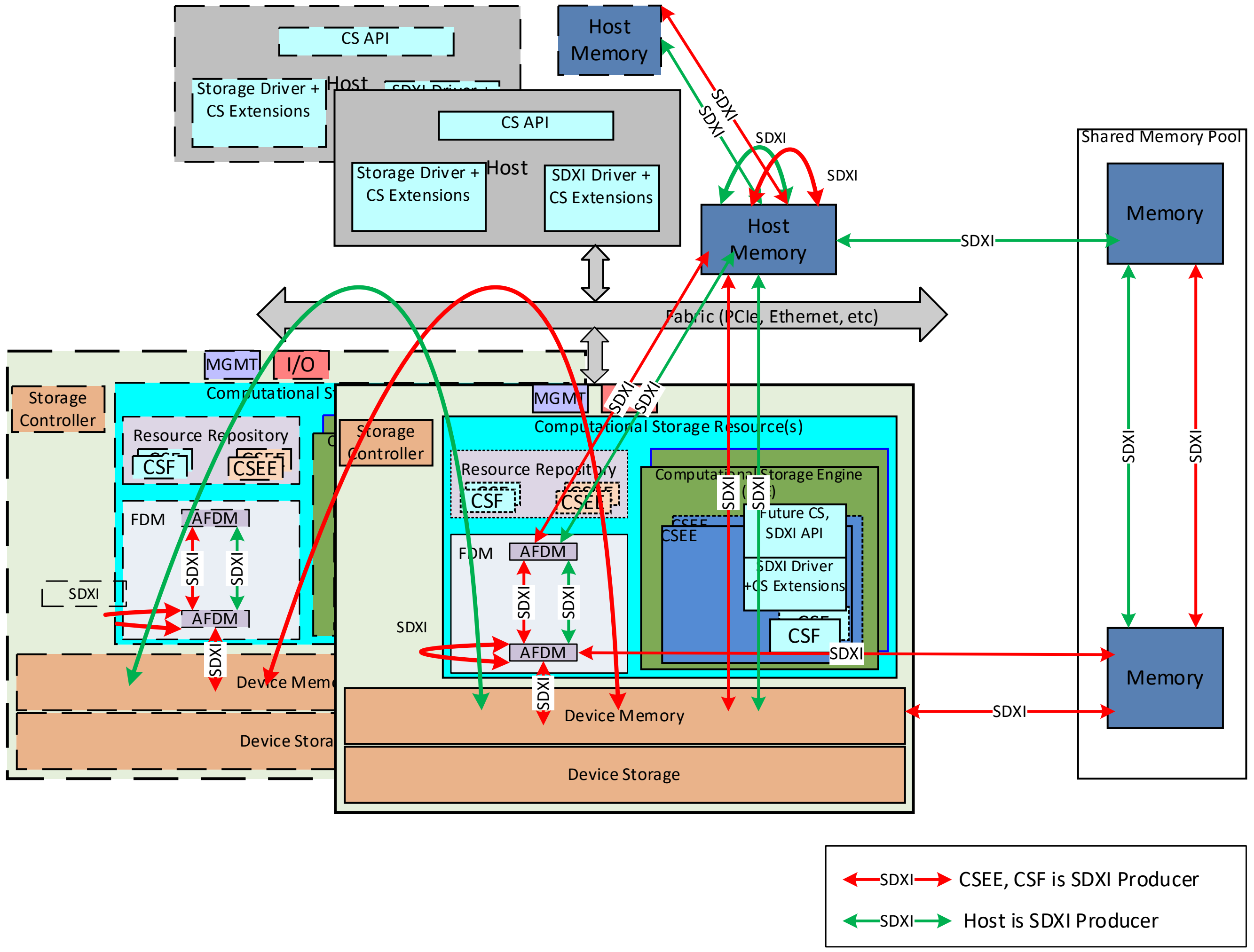
Point of View: Computational Storage, NVMe, and SDXI



Type A Device



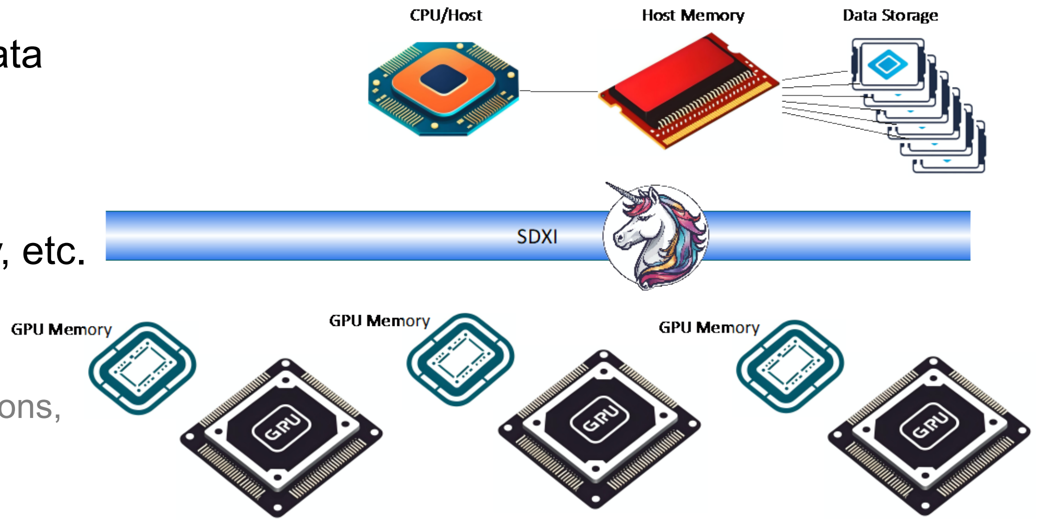
Type B Device


CS API
Storage Driver + CS Extensions
Host
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
Computational Storage Engine
Future CS, SDXI API
SDXI Driver +CS Extensions
FDM
AFDM
Device Memory
Device Storage
CSEE, CSF is SDXI Producer
Host is SDXI Producer

Point of view: Does it apply to AI? Yes!!!

- Varying data formats and intermediate data representations used in AI/ML data pipelines
  - E.g., file, Columnar, Binary, Text, Tabular, Nested, Array-based, Hierarchical
- Training/inferencing operations involve tensors in memory
- Tensors may be in different address spaces like Host Memor ry, etc.
- Need operations to be able to perform
  - Format Conversions
  - In memory Vector/Tensor transformations li ing, matrix operations, etc.
  - …
- Vendor-specific accelerat en TCO
- Possible Solution: S
  - Smart Dat ce (SDXI) is a SNIA standard for a memory to memory data eration interface that is
    - compatible
    - dependent of I/O interconnect technology
  - ata movement between different address spaces.
  - Standard extends to in-memory Offloads/transformations leveraging the architectural interface.

SDXI has emerging AI use cases

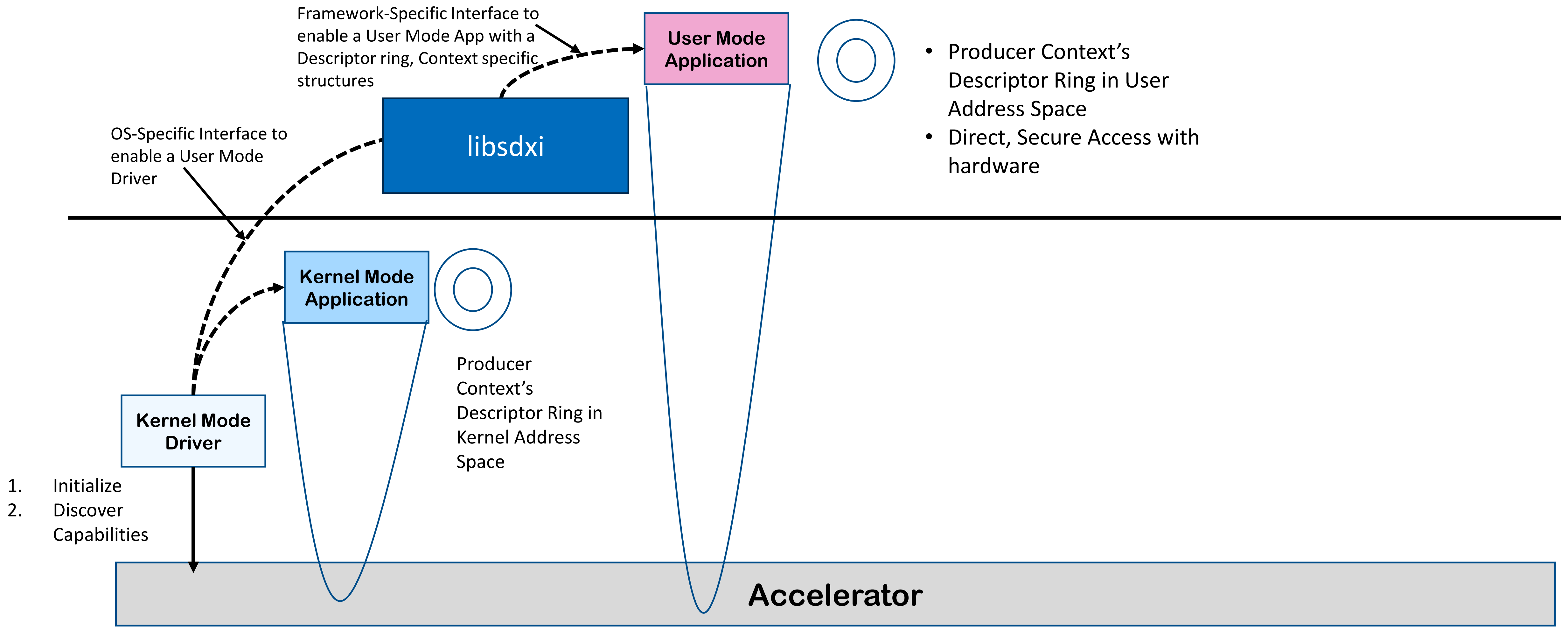


Software Enablement

Software Ecosystem

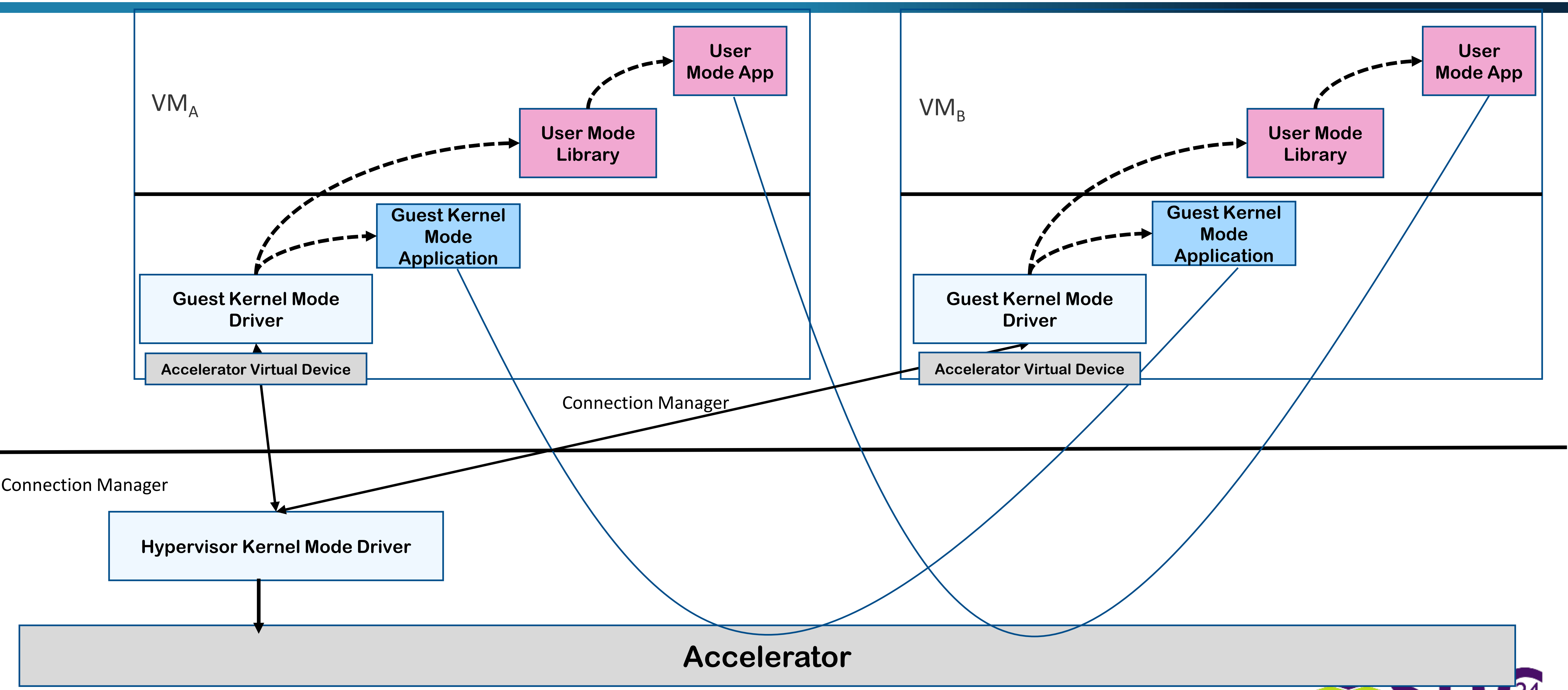
- SDXI TWG is working on libsdxi
  - OS-agnostic user space library
  - Helps user space applications use SDXI accelerated data movement operations
  - Control Plane API
    - Probing resource discovery
    - Context management
    - Connection management
  - Data Plane API
    - Memcpy
    - Zero Memfill
    - <Memory Operations>
- SDXI TWG is enabling SDXI driver work in various OSes
- SDXI Kernel mode Use cases
  - Linux DMA engine
  - Mem-zero
  - Autonuma aka numa page migration
- SDXI emulation project investigation for ecosystem development

Baremetal Stack View


Framework-Specific Interface to enable a User Mode App with a Descriptor ring, Context specific structures
User Mode Application
libsdxi
OS-Specific Interface to enable a User Mode Driver
Kernel Mode Application
Producer Context's Descriptor Ring in Kernel Address Space
Kernel Mode Driver
1. Initialize
2. Discover Capabilities
Accelerator

- Producer Context's Descriptor Ring in User Address Space
- Direct, Secure Access with hardware

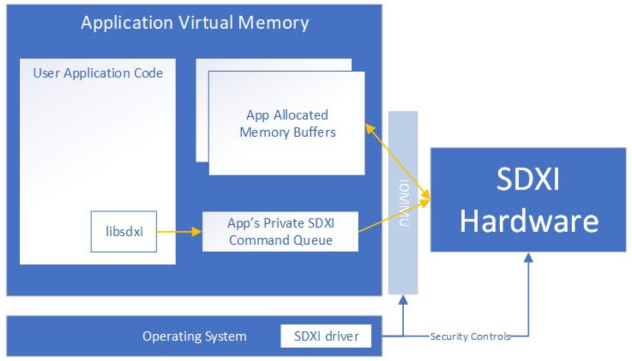
Scale with Compute Virtualization– Multi-VM address space

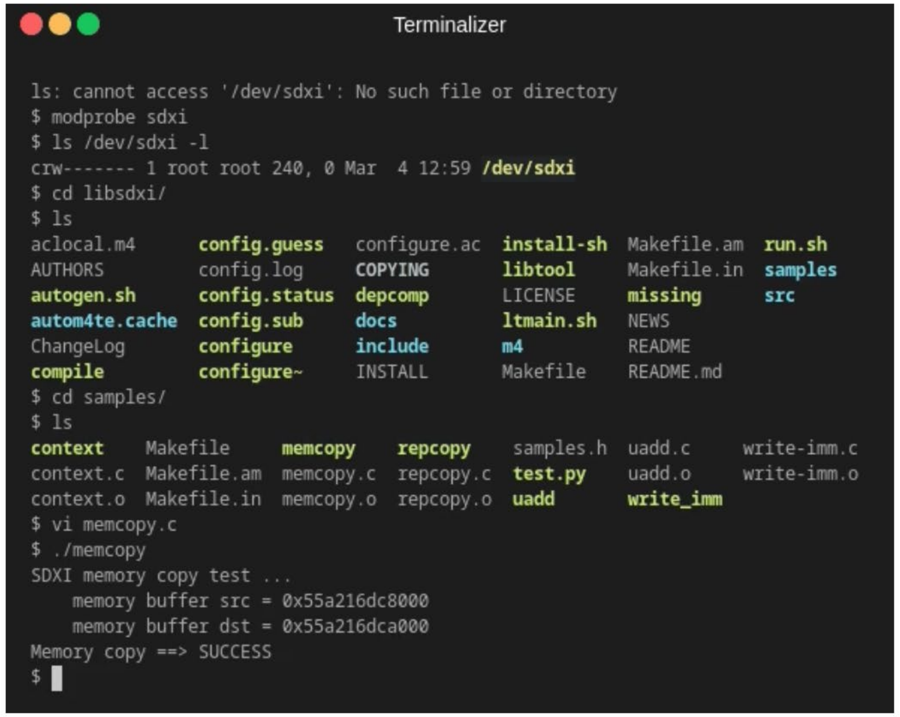


Proofpoints

Proofpoints: SDXI PoC Demo at Memcon 2024

SDXI Sample User Mode application with Linux





```
ls: cannot access '/dev/sdxi': No such file or directory
$ modprobe sdxi
$ ls /dev/sdxi -l
crw------- 1 root root 240, 0 Mar  4 12:59 /dev/sdxi
$ cd libsdxi/
$ ls
aclocal.m4      config.guess   configure.ac  install-sh  Makefile.am  run.sh
AUTHORS         config.log     COPYING       libtool     Makefile.in  samples
autogen.sh      config.status  depcomp       LICENSE     missing      src
autom4te.cache  config.sub     docs          ltmain.sh   NEWS
ChangeLog       configure      include       m4          README
compile         configure~     INSTALL       Makefile    README.md
$ cd samples/
$ ls
context    Makefile     memcopy    repcopy    samples.h  uadd.c     write-imm.c
context.c  Makefile.am  memcopy.c  repcopy.c  test.py    uadd.o     write-imm.o
context.o  Makefile.in  memcopy.o  repcopy.o  uadd       write_imm
$ vi memcopy.c
$ ./memcopy
SDXI memory copy test ...
    memory buffer src = 0x55a216dc8000
    memory buffer dst = 0x55a216dca000
Memory copy ==> SUCCESS
$ █
```

5 | © SNIA. All Rights Reserved.

SNIA

Summary and Call to Action

- SNIA is developing SDXI a memory to memory data movement standard
  - v1.0 released!
- Multiple companies involved in the effort
- SDXI standard continues to improve with new features and use cases
  - SDXI TWG working v1.1 specification
  - TWG has a framework and roadmap for v1.1, v1.2, and v2.0
- SDXI software ecosystem is developing, and proof points are emerging
- Learn more:
  - https://www.snia.org/sdxi

Q&A

Please take a moment to rate this session.

Your feedback is important to us.