STORAGE DEVELOPER CONFERENCE

SDC 23

BY Developers FOR Developers

SNIA SDXI Specification v1.0 and beyond

Shyam Iyer

Chair, SNIA SDXI TWG

Member, SNIA Technical Council

Distinguished Engineer, Dell

SNIA Legal Notice

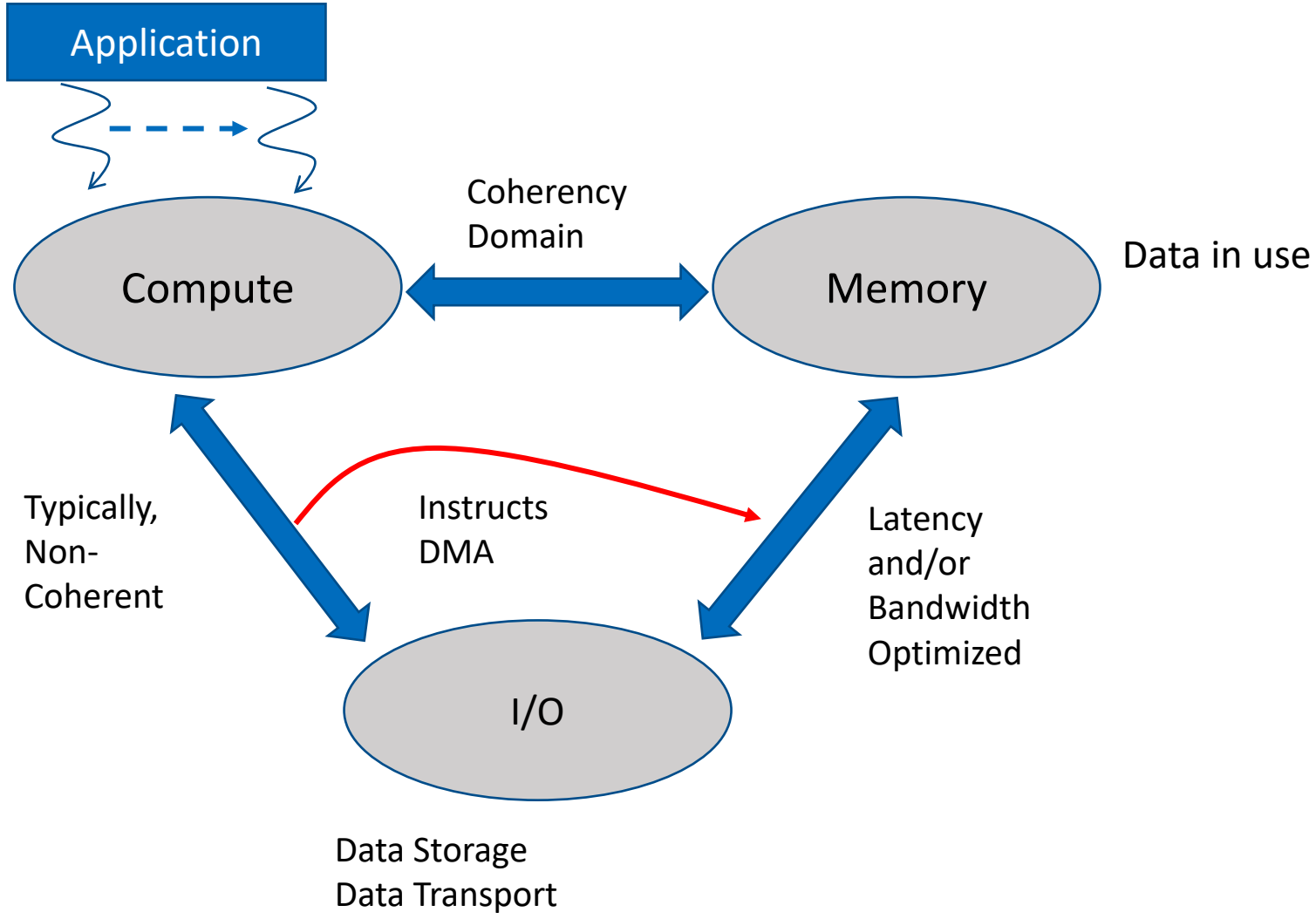
- This presentation is a project of SNIA, and the material contained in this presentation is copyrighted by the SNIA unless otherwise noted.
- SNIA member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations
- Neither the author nor the presenter is an attorney and nothing in this presentation is intended to be or should be construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion, please contact your attorney.
- The information presented herein represents the author's personal opinion and current understanding of the relevant issues involved.
- The author, the presenter, and SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

  NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

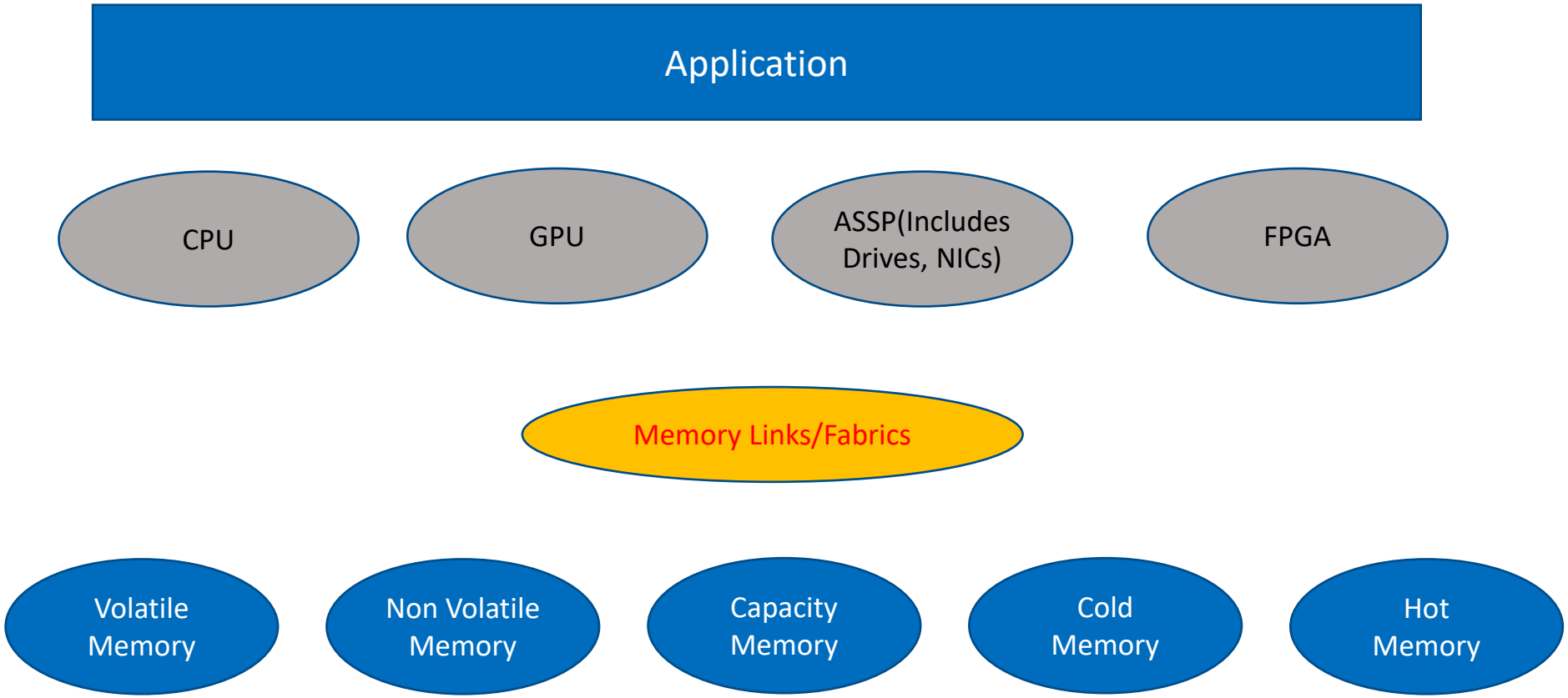
Agenda

- Compute, IO, Memory Bubble
  - Current Memory to Memory Data Movement Standard
- Use Cases
  - Application Patterns and benefits of Data Movement & Acceleration
- SNIA SDXI TWG
  - Goals and Tenets
  - A brief introduction to the internals of SDXI Specification
- SDXI Futures
- SDXI Community/Ecosystem
- Summary

Legacy Compute, Memory, IO Bubbles



Emerging Bubbles



Shared Design constraints

- Latency
- Bandwidth
- Coherency
- Control

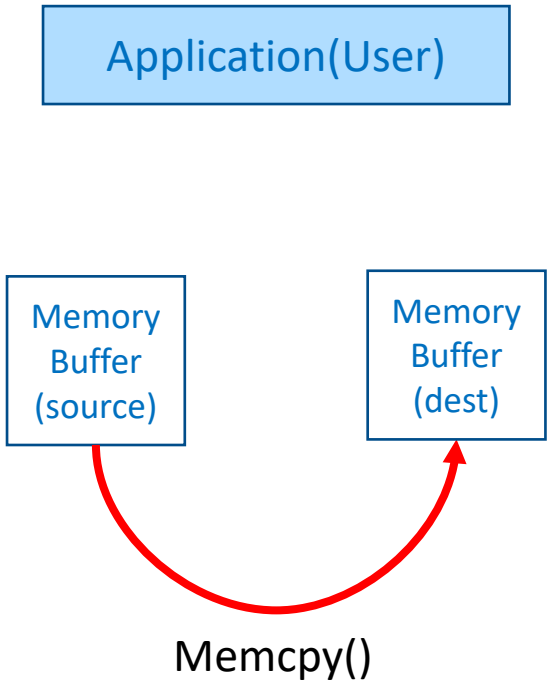
Current Data Movement Standard

- 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.

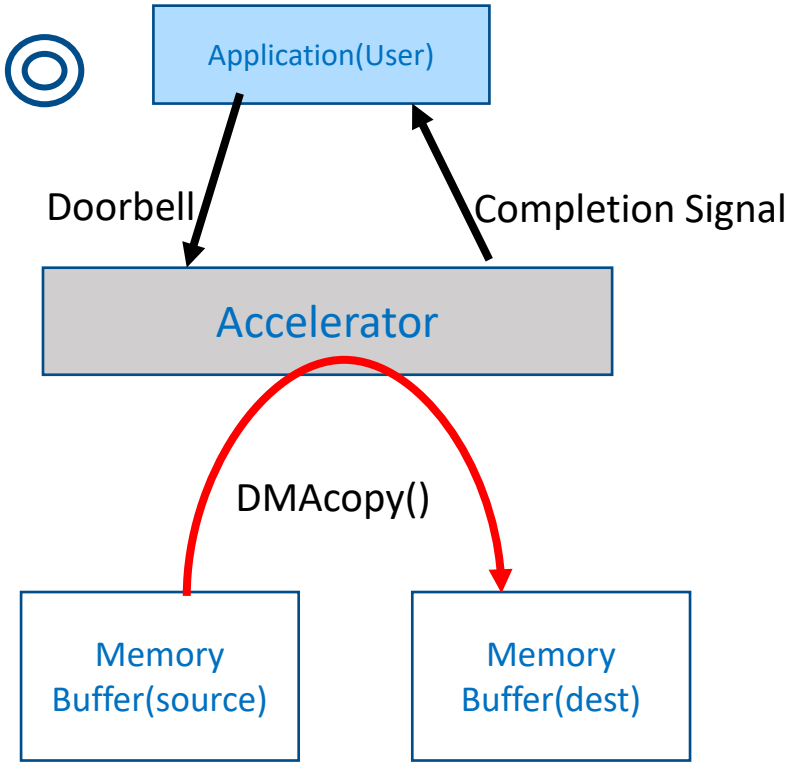
Agenda

- Compute, IO, Memory Bubble
  - Current Memory to Memory Data Movement Standard
- Use Cases
  - Application Patterns and benefits of Data Movement & Acceleration
- SNIA SDXI TWG
  - Goals and Tenets
  - A brief introduction to the internals of SDXI Specification
- SDXI Futures
- SDXI Community/Ecosystem
- Summary

Application Pattern 1 (Buffer Copies)

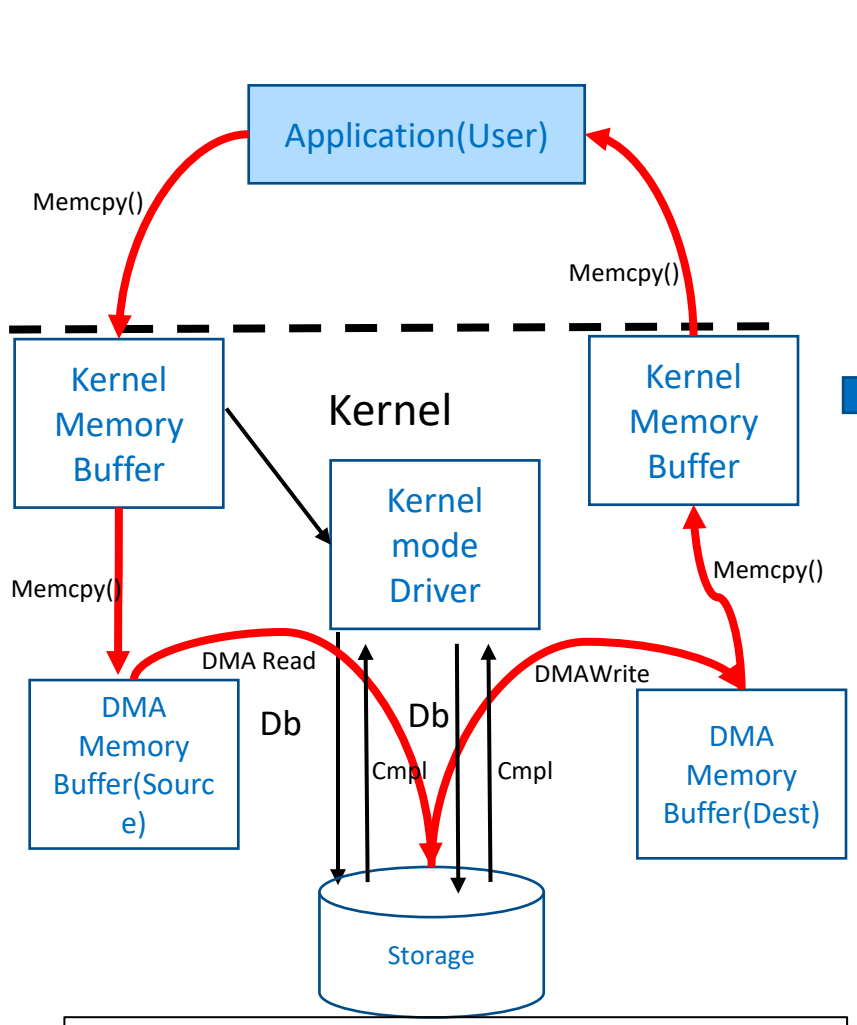


- Takes away from application performance

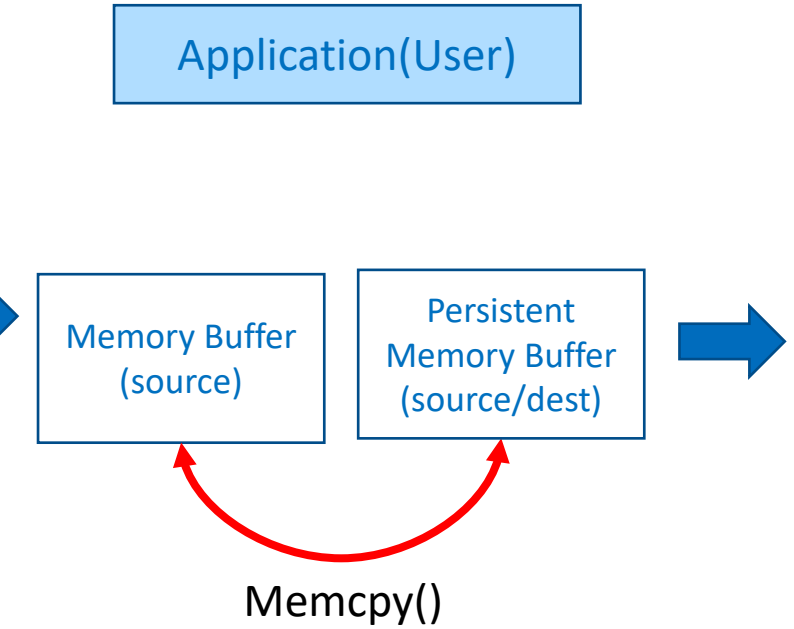


- HW based memory copies can be offloaded without affecting application performance

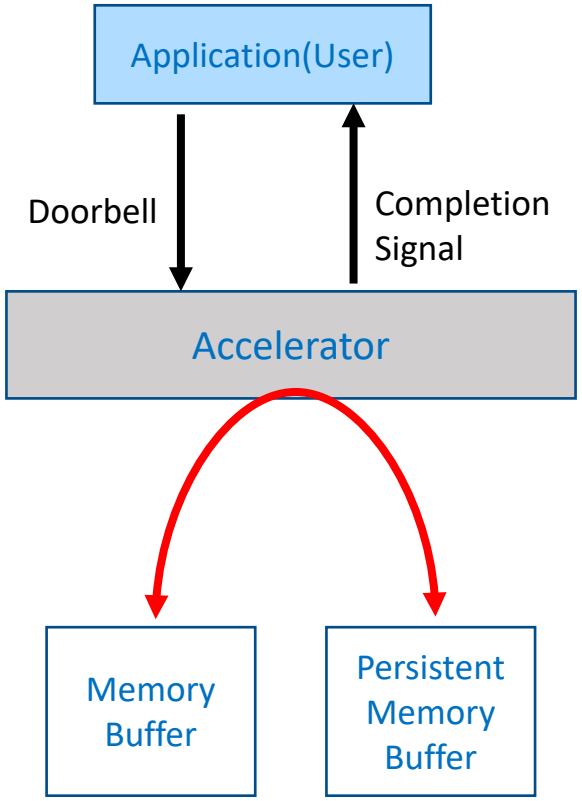
Application Pattern 2



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

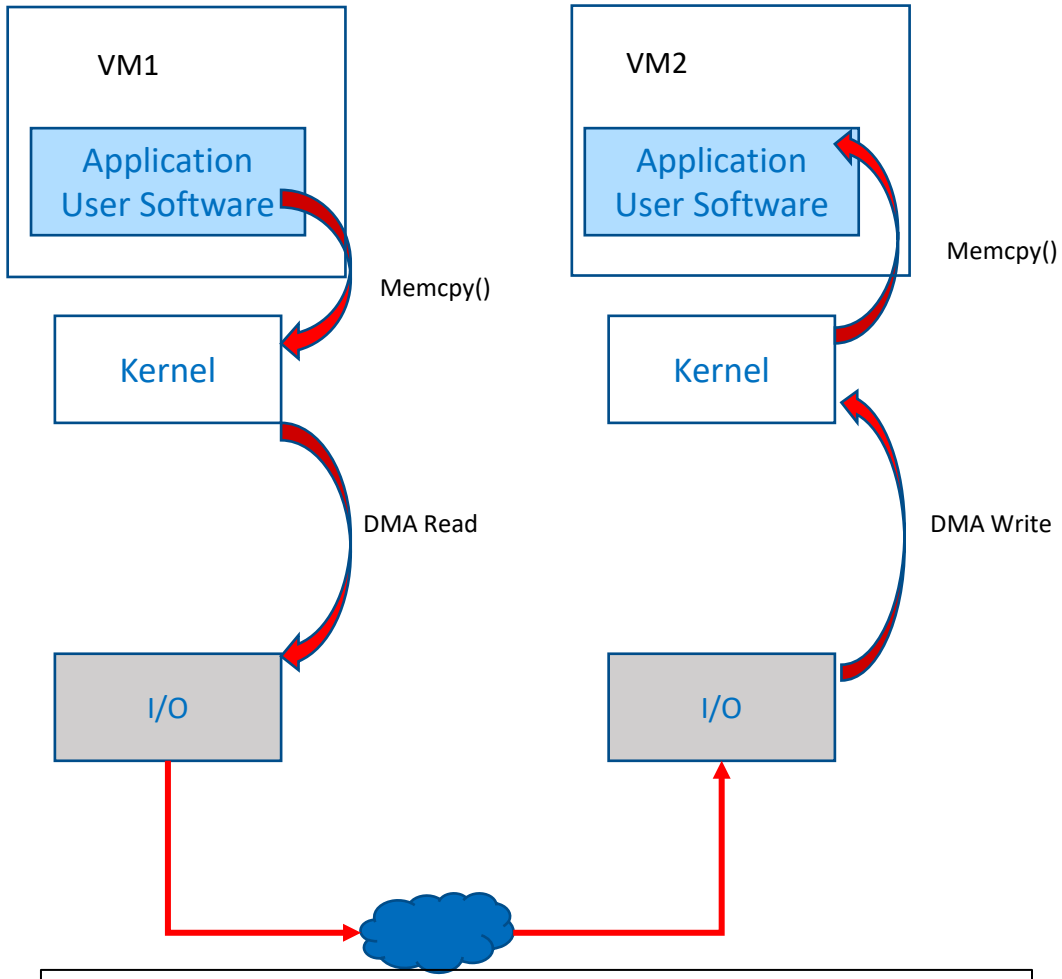


- Reduced buffer copies but still takes away from application performance

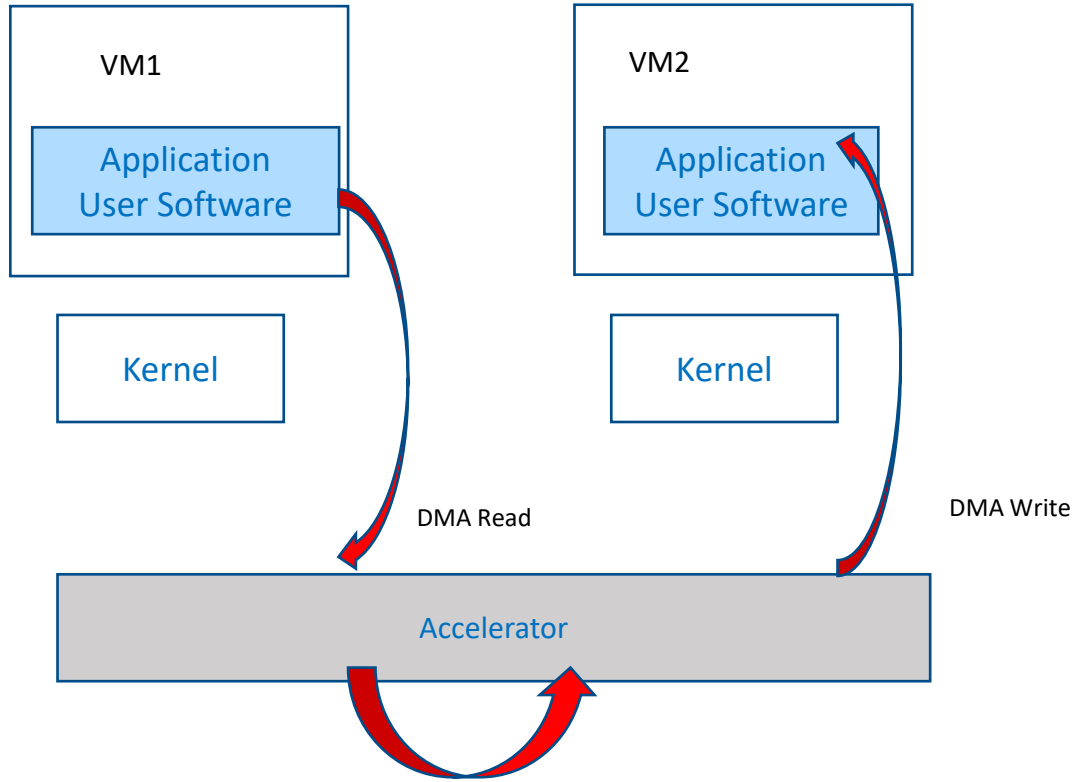


- Reduced buffer copies
- HW based offloaded memory copies

Application Pattern 3

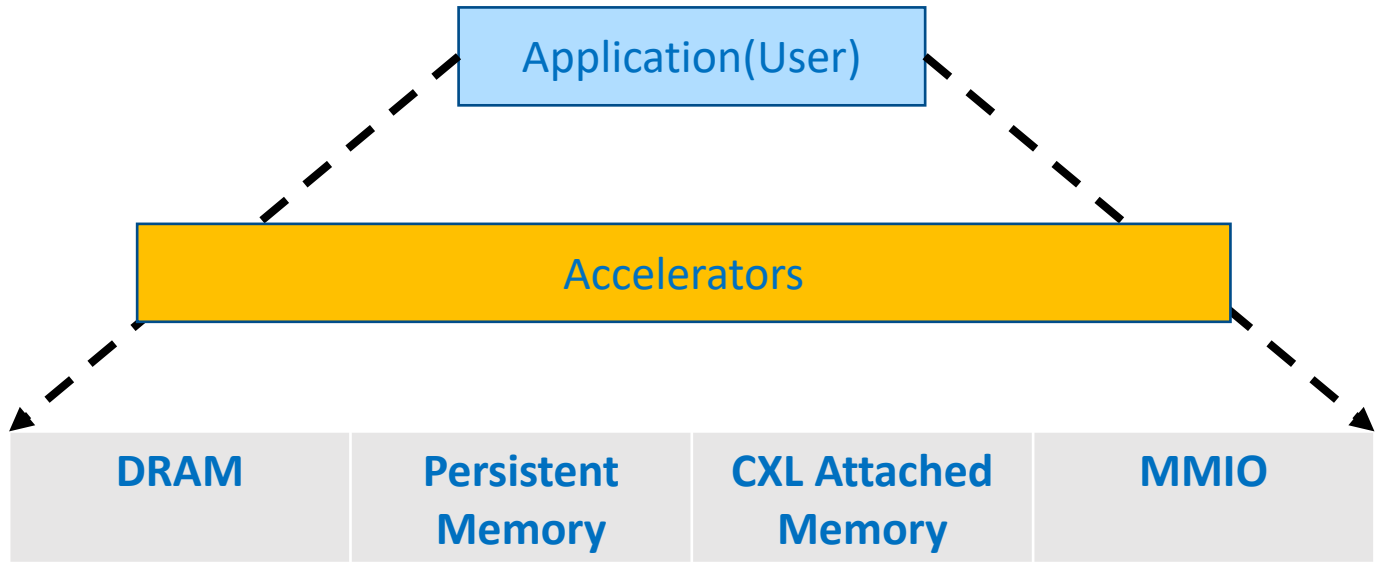


- Context isolation layers introduce multiple buffer copies



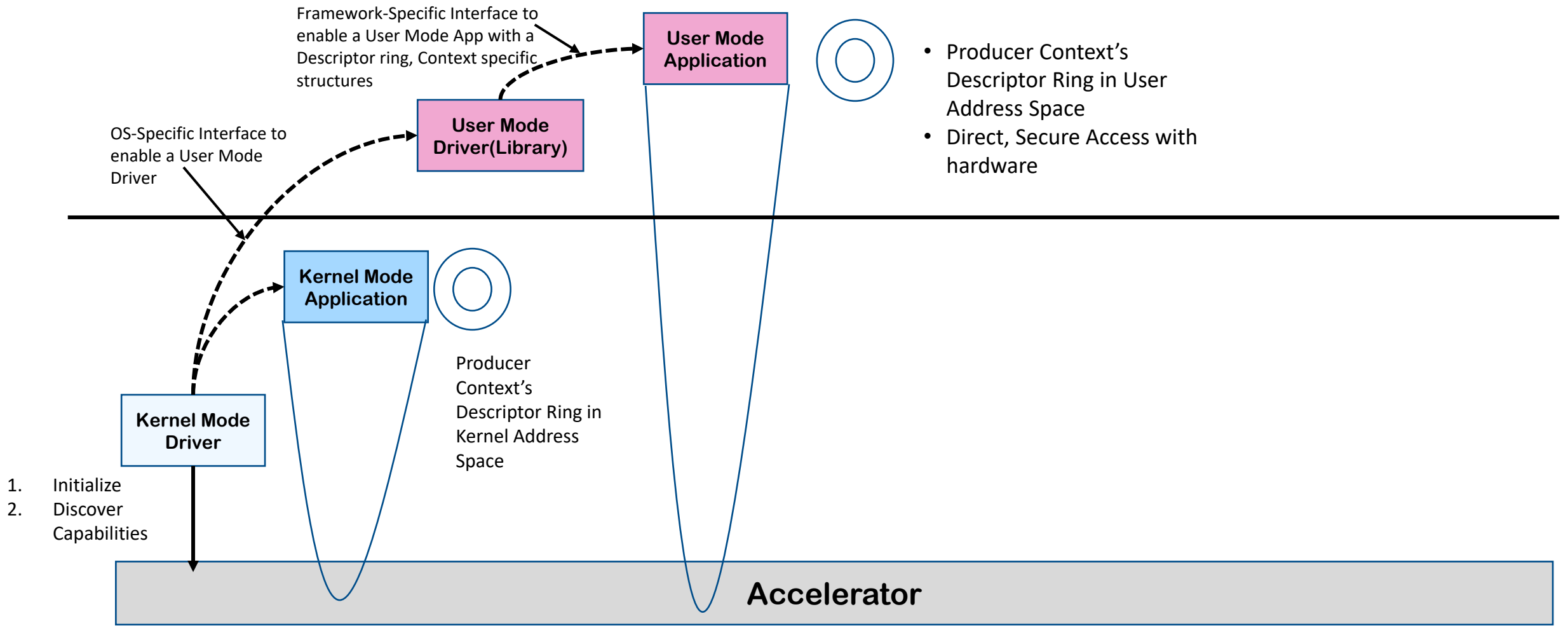
- Best of both: Context isolation layers and optimized HW based memory buffer copies

Data *in use* Memory Expansion



- Memory expansion expands the memory target surface area for accelerators
- Different tiers of memory
- Diversity in accelerator programming methods

Baremetal Stack View

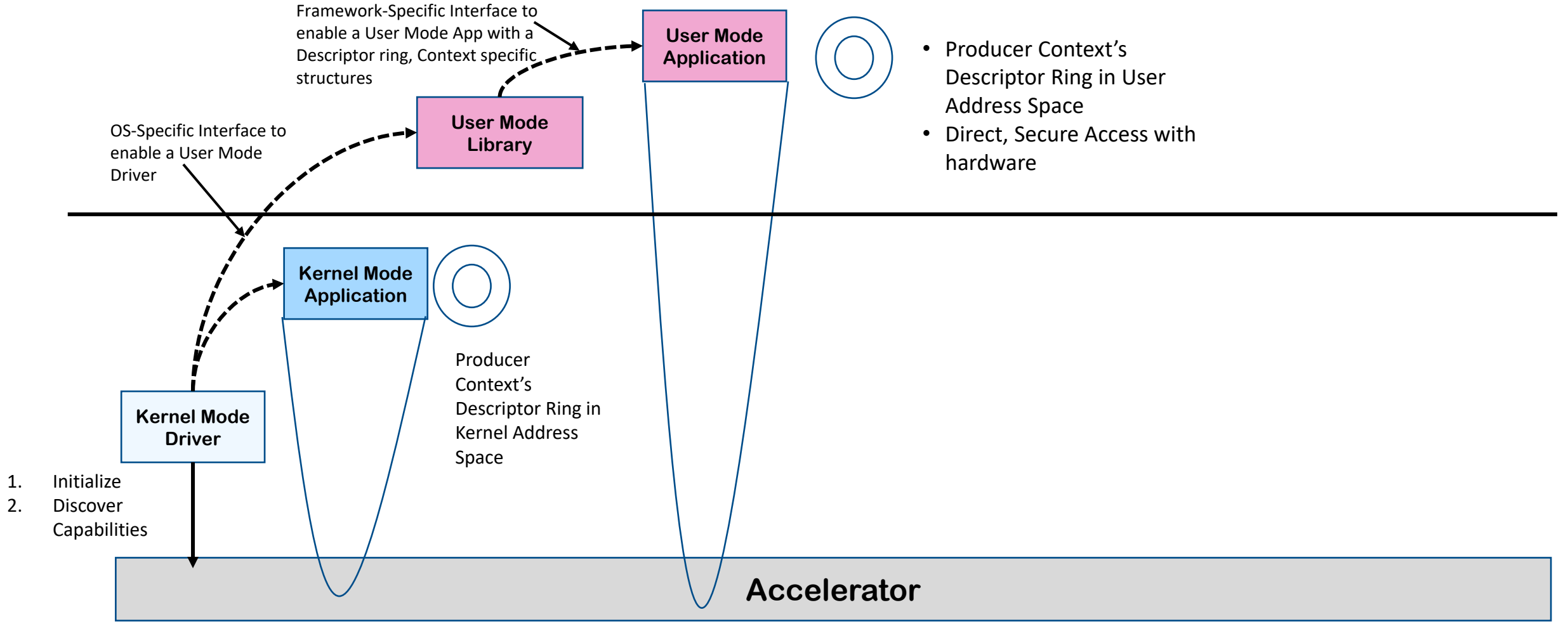


- Producer Context's Descriptor Ring in User Address Space
- Direct, Secure Access with hardware

Direct HW Access, Access Memory Tiers

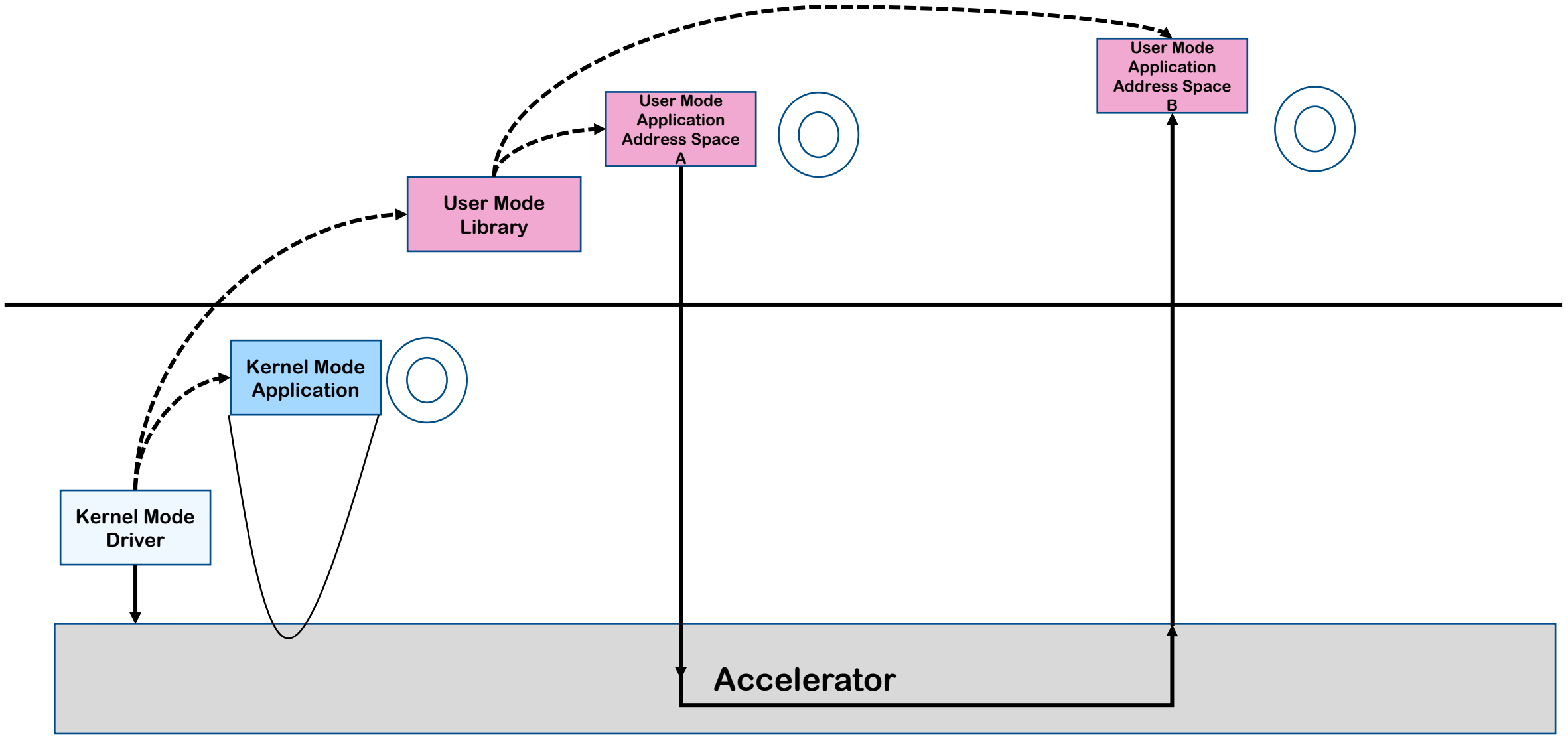
DRAM	PMEM	MMIO	CXL Memory

Source and Destination Memory Targets for Data transfer in System Physical Address Space

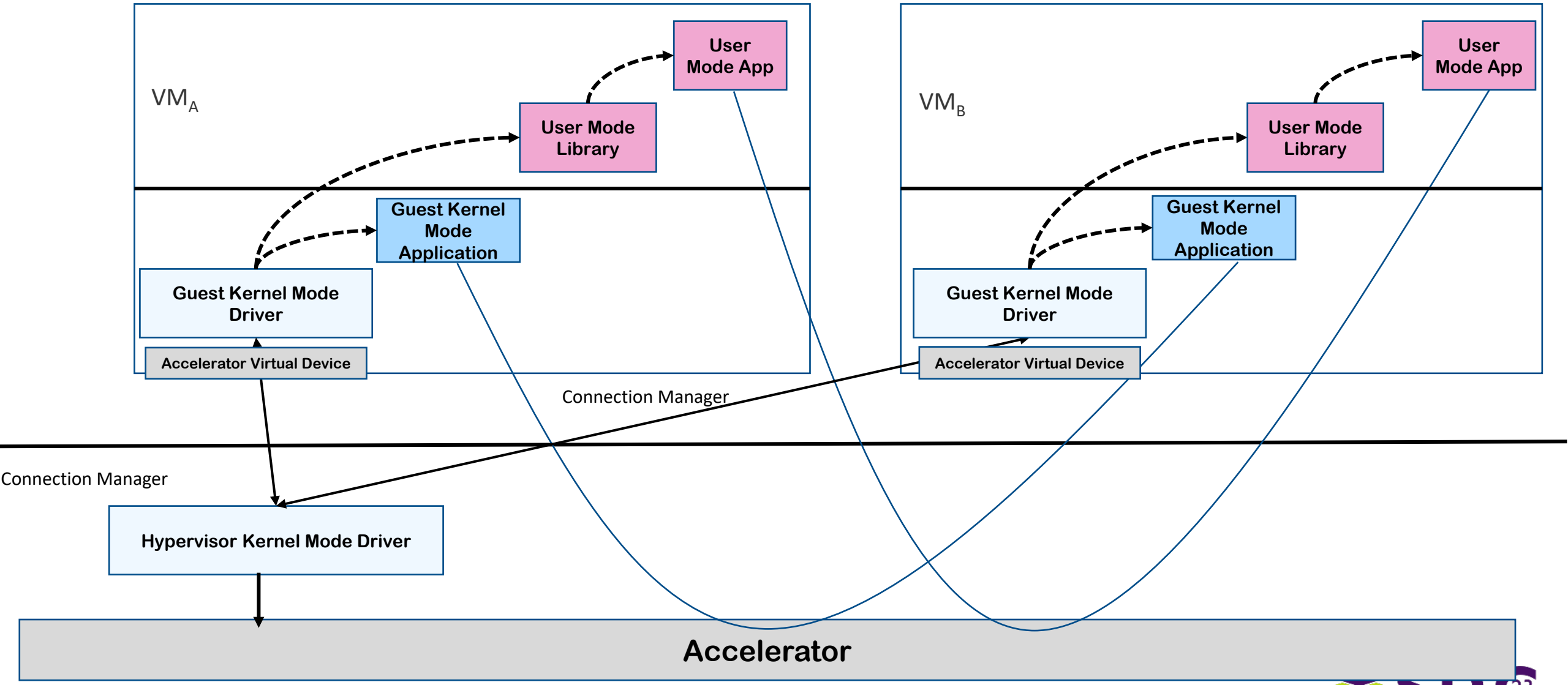


- Producer Context's Descriptor Ring in User Address Space
- Direct, Secure Access with hardware

Scale Baremetal Apps – Multi-Address Space



Scale with Compute Virtualization– Multi-VM address space



Agenda

- Compute, IO, Memory Bubble
  - Current Memory to Memory Data Movement Standard
- Use Cases
  - Application Patterns and benefits of Data Movement & Acceleration
- SNIA SDXI TWG
  - Goals and Tenets
  - A brief introduction to the internals of SDXI Specification
- SDXI Futures
- SDXI Community/Ecosystem
- Summary

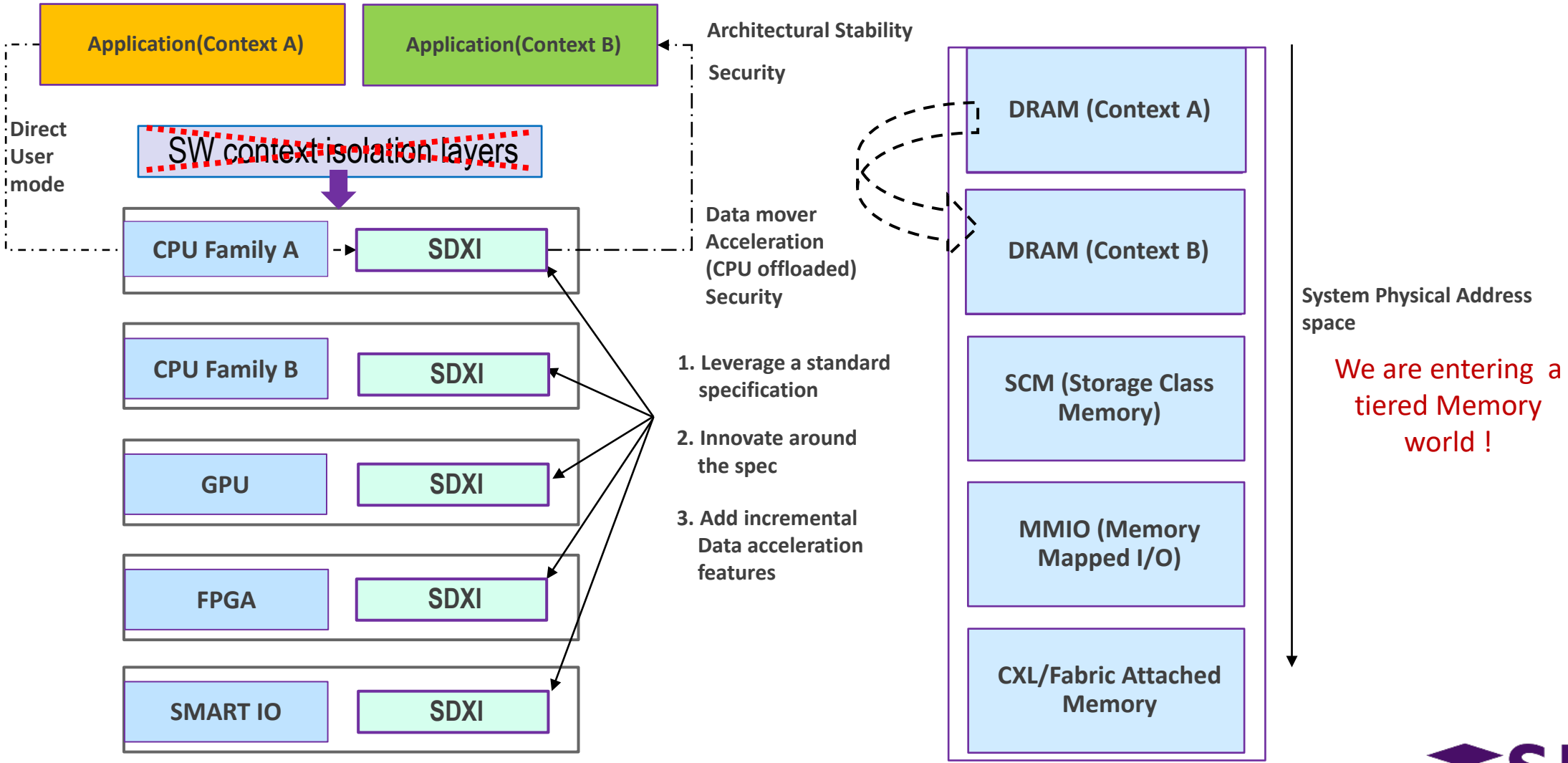
SDXI(Smart Data Accelerator Interface)

- 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

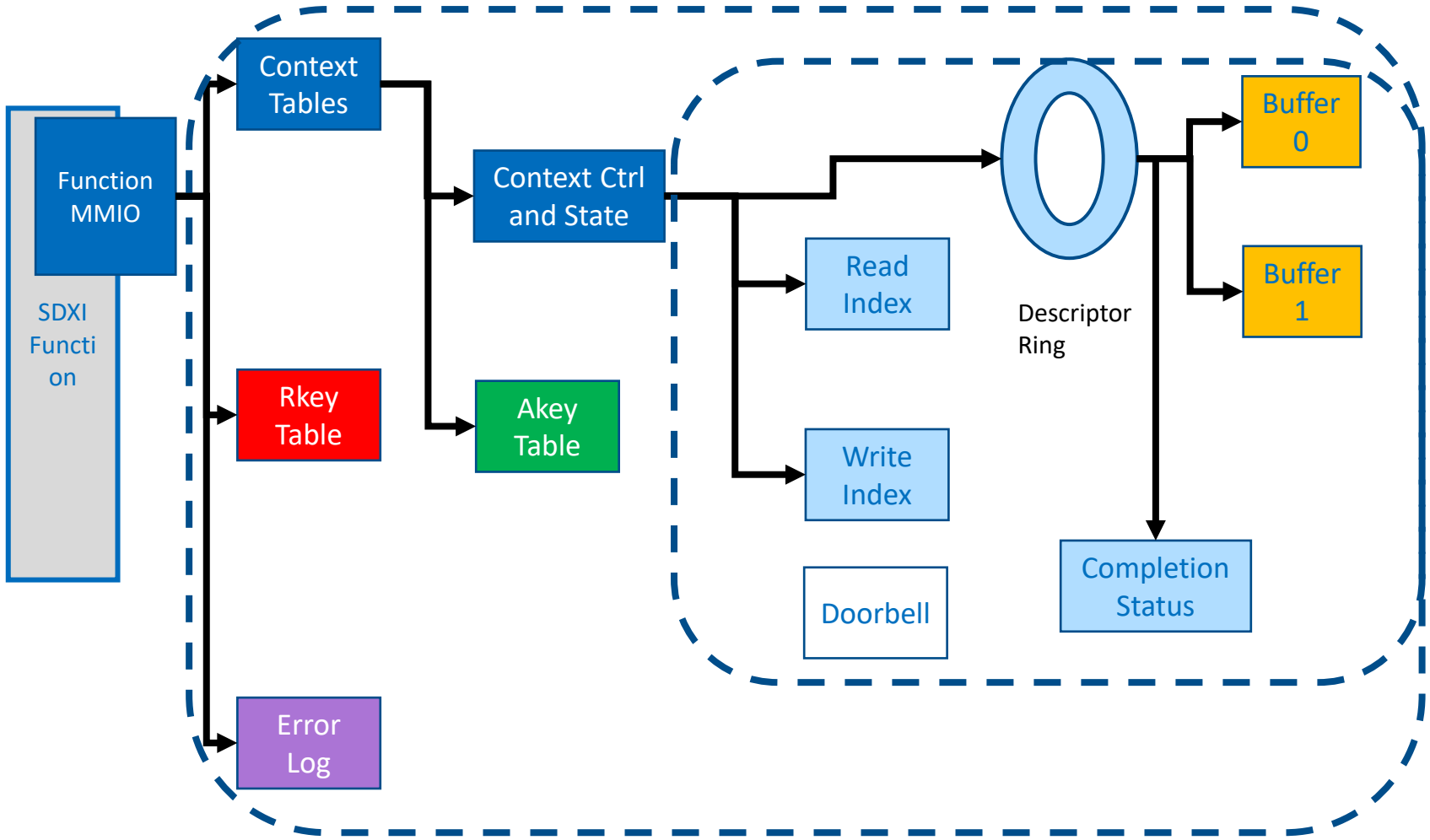
SDXI Design Tenets

- Data movement between different address spaces.
  - Includes user address spaces, different virtual machines
- Data movement without mediation by privileged software.
  - Once a connection has been established.
- Allows abstraction or virtualization by privileged software.
- Capability to quiesce, suspend, and resume the architectural state of a per-address-space data mover.
  - Enable “live” workload or virtual machine migration between servers.
- Enables forwards and backwards compatibility across future specification revisions.
  - Interoperability between software and hardware
- Incorporate additional offloads in the future leveraging the architectural interface.
- Concurrent DMA model.

SDXI Memory-to-Memory Data Movement

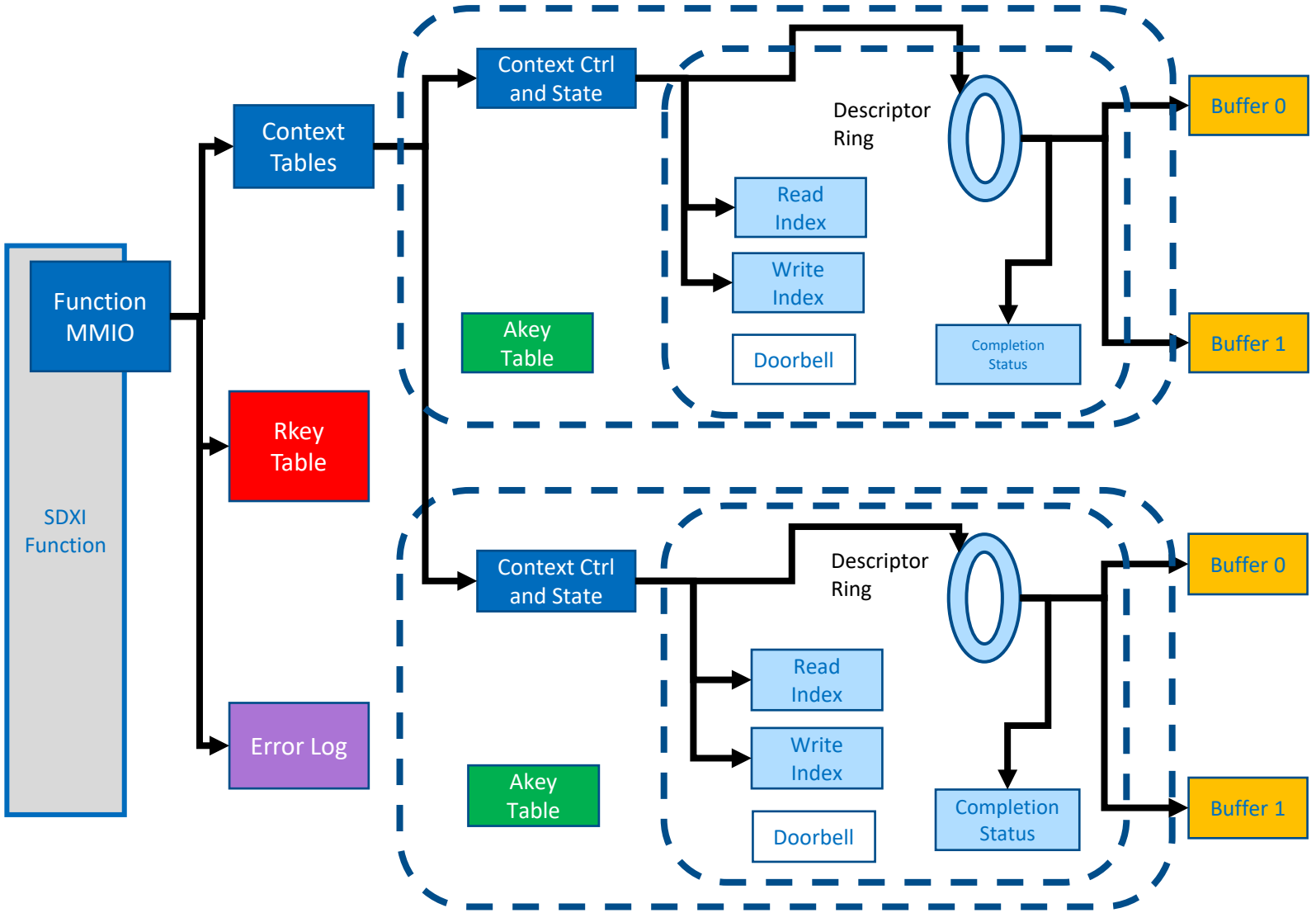


Memory Structures(1) – Simplified view



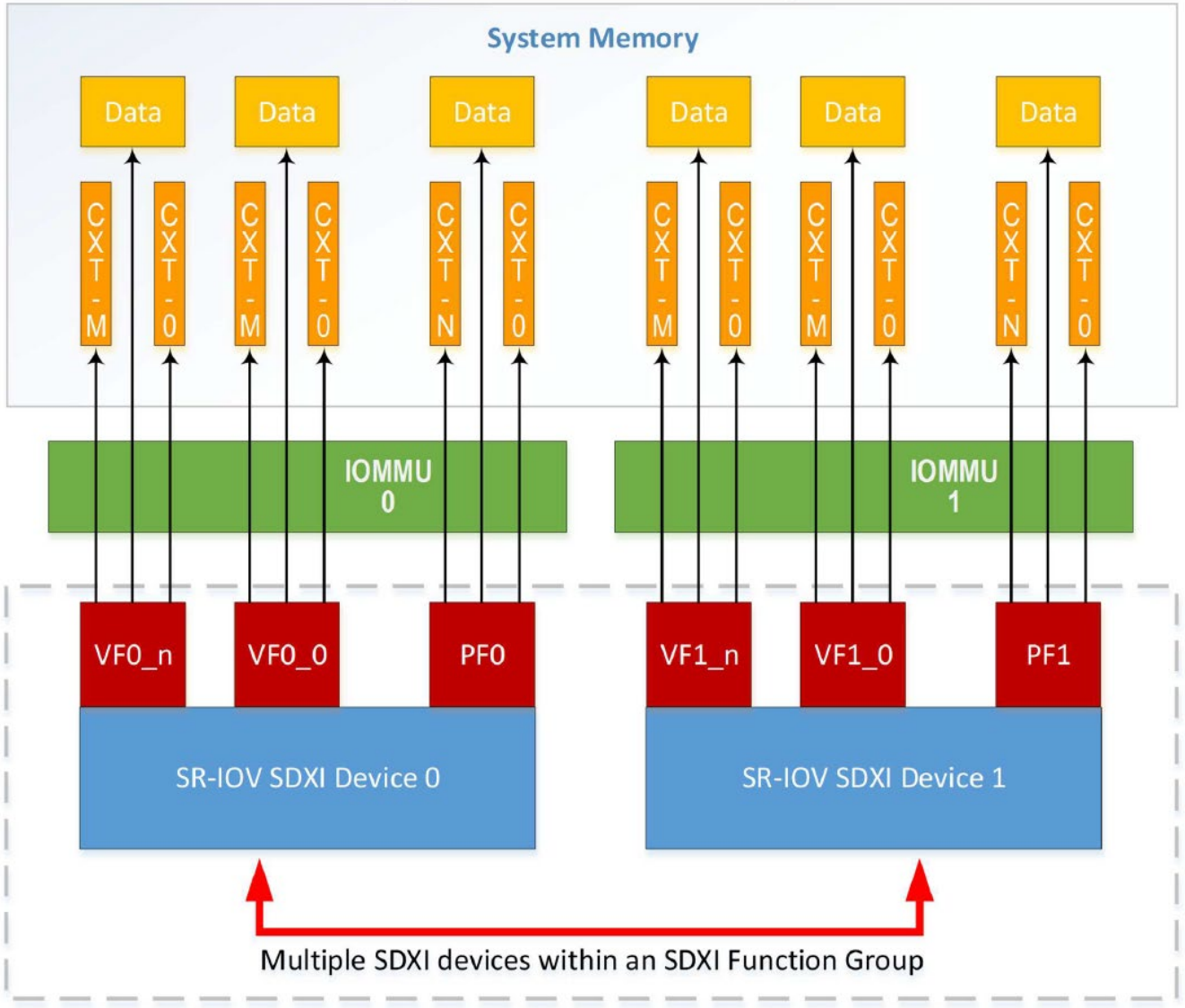
- All states in memory
- One standard descriptor format
- Easy to virtualize
- Architected function setup and control
  - *layered model for interconnect specific function management
  - SDXI class code registered for PCIe implementations

Memory Structures(2) – Multiple Contexts

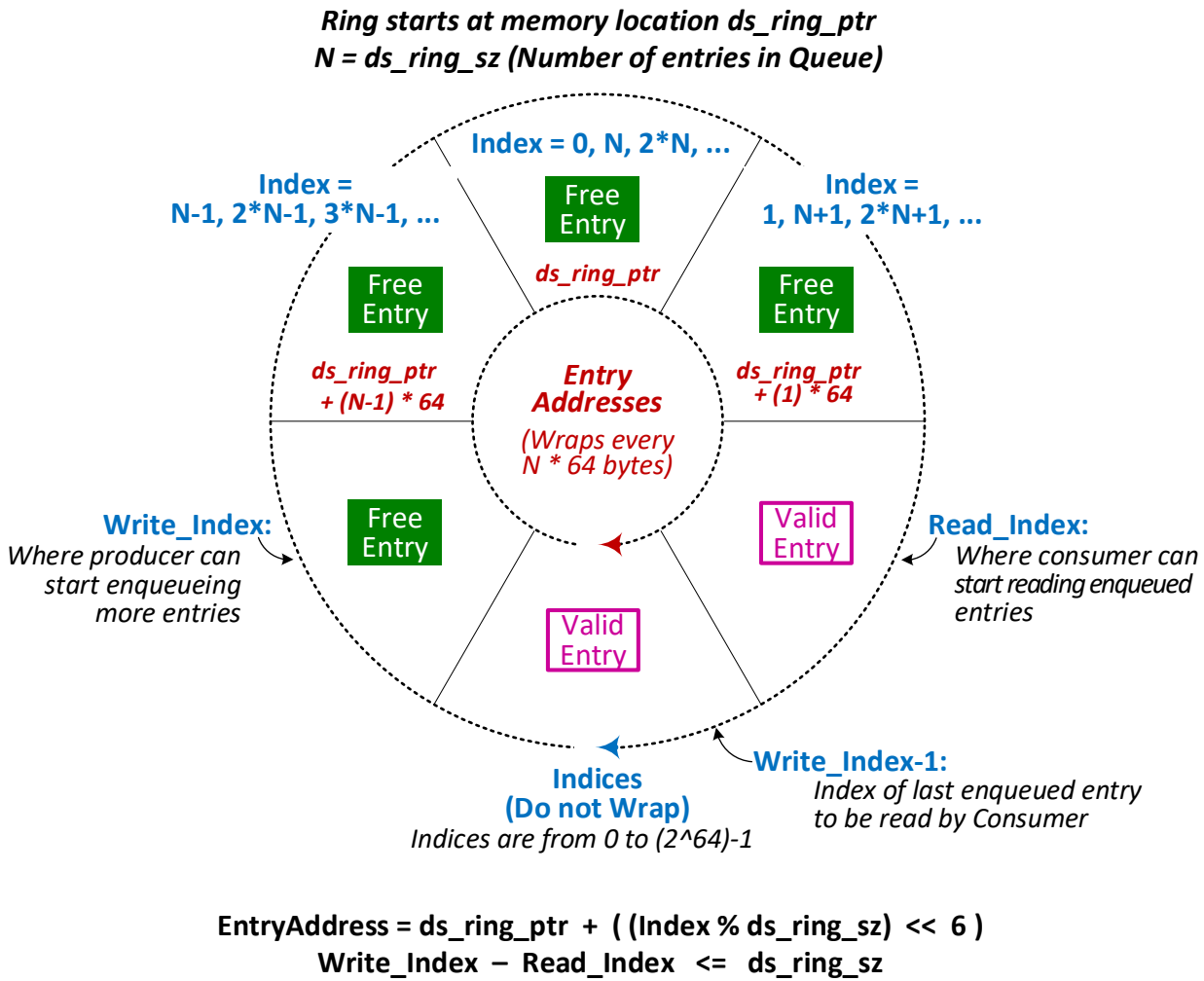


- Multiple Contexts per function
- Ring State directly managed by user space
- One way to log errors
- Per context access to target address spaces(Akey)
- One way to control access to local memory resources from remote functions(Rkey)
- One way to start, stop and administer contexts

Contexts and SDXI Function Groups



Descriptor Ring

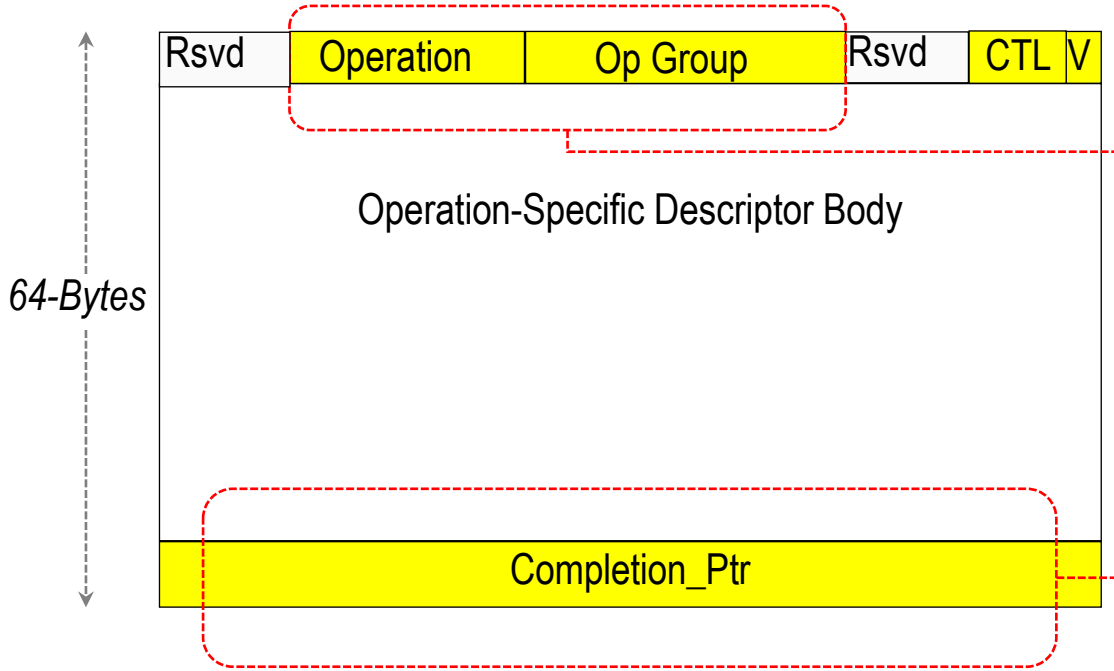


EntryAddress = ds_ring_ptr + ((Index % ds_ring_sz) << 6)

Write_Index – Read_Index <= ds_ring_sz

- Descriptors are processed (issued) in-order by function.
  - Executed out-of-order.
  - Completed out-of-order.
  - Read_Index is incremented by SDXI function
- Function may aggressively read valid descriptors…
  - Between Read & Write indices w/o waiting on Doorbells from producers.
  - Doorbell ensures new descriptors are recognized.
- Maximum parallelism of operations. Quiescing & Serializing state at well-defined boundaries.

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

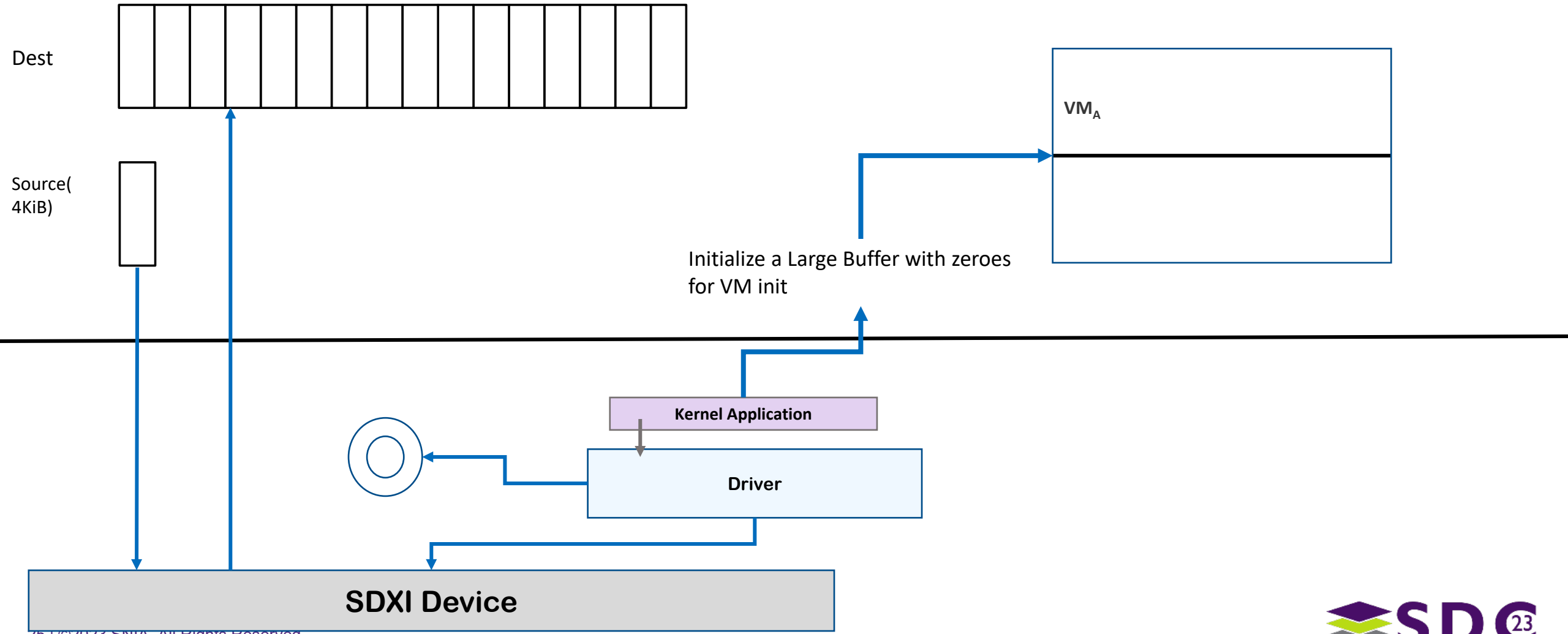
Admin: Start/Stop/Update/Sync, Interrupt Function & Contexts (easily virtualizable)

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

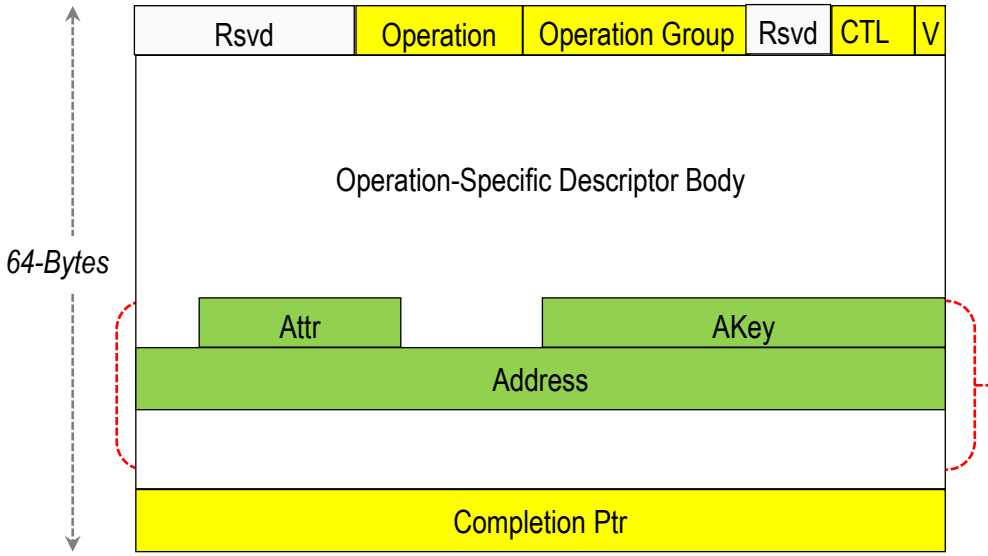
- Completion Signal
  - Initialized by SW, Decremented by Function on Success
- Error Bit(ER) to indicate the operation encountered an error
- Other bits in the 32-byte field are reserved to support future expansion of error codes

*Room for lots of future operations

RepCopy Example



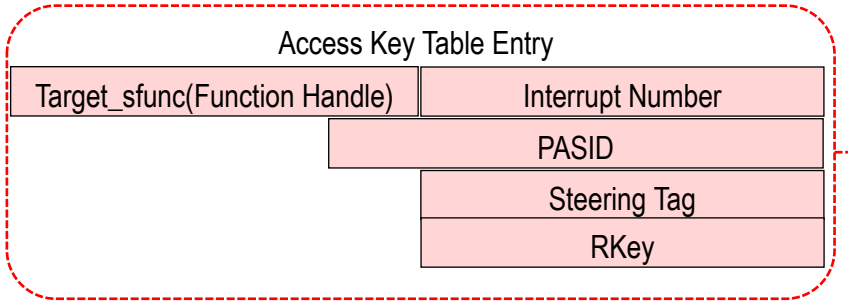
A Standard Descriptor Format (2)



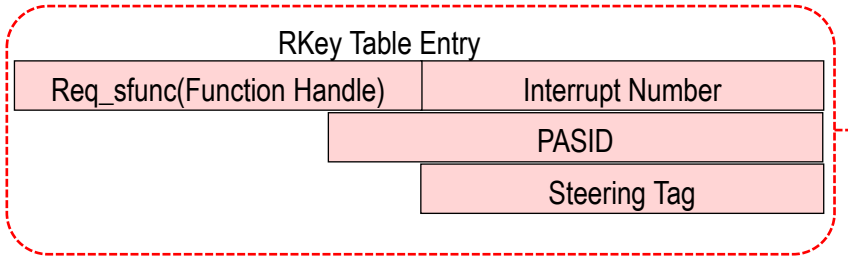
A memory location is always specified as a triple:

- Address Space ID: Index to Context Address Key Table Entry
- 64-bit Address
- Cacheability Attributes

Generated Address can be HPA, HVA, GPA, GVA and always translated through IOMMU.

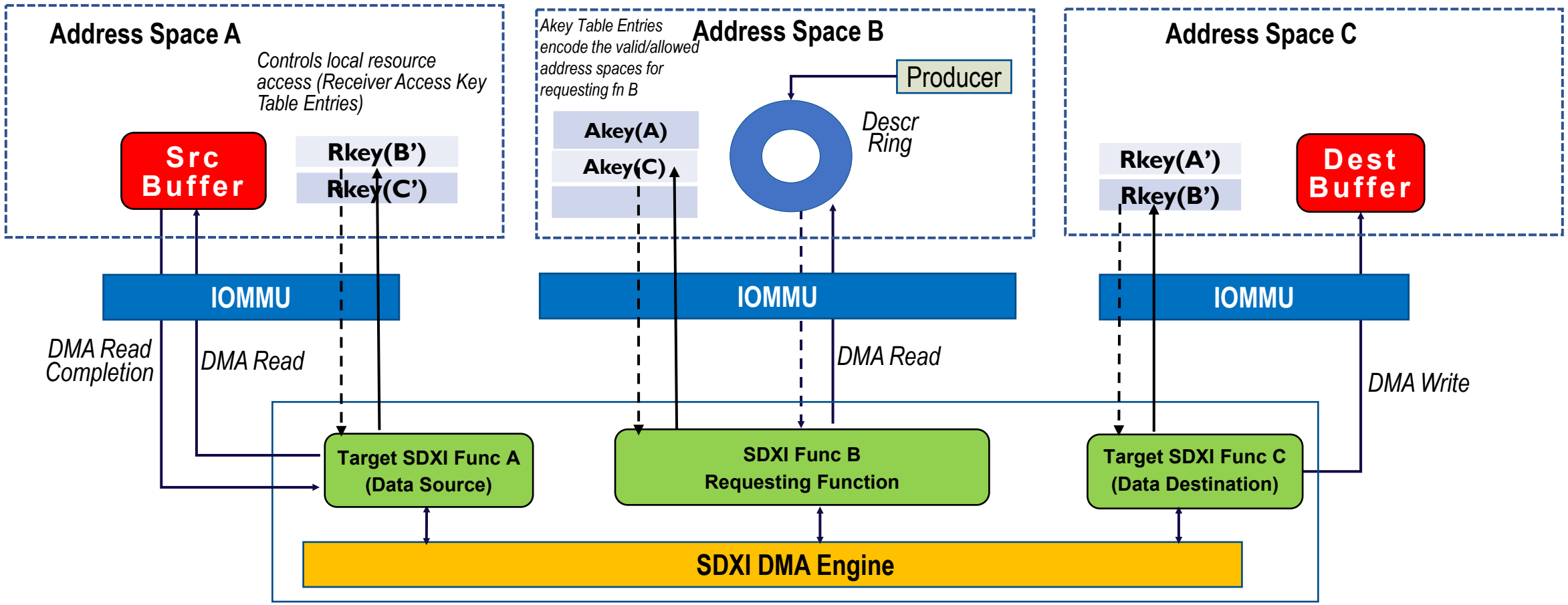


- An AKey table encodes all valid address spaces, PASIDs and interrupts available to the function context.
- Akey table entries enumerate each address space, pasids and interrupts resources
- Any descriptor within a context can reference an AKey table entry.
- An AKey is a requester side control.The Akey also encodes the Rkey to be used by the Target address space. The Target Function uses the supplied RKey value to index into its RKey Table and obtain an RKey Table Entry



- An RKey table entry controls target/receiver side resources
- Rkey index supplied by Akey table entry from requesting function's Akey table entry is used to index into RKey table entry in receiver's RKey table
- Req_sfunc handle in Rkey table entry should match the req_sfunc supplied by requesting function

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



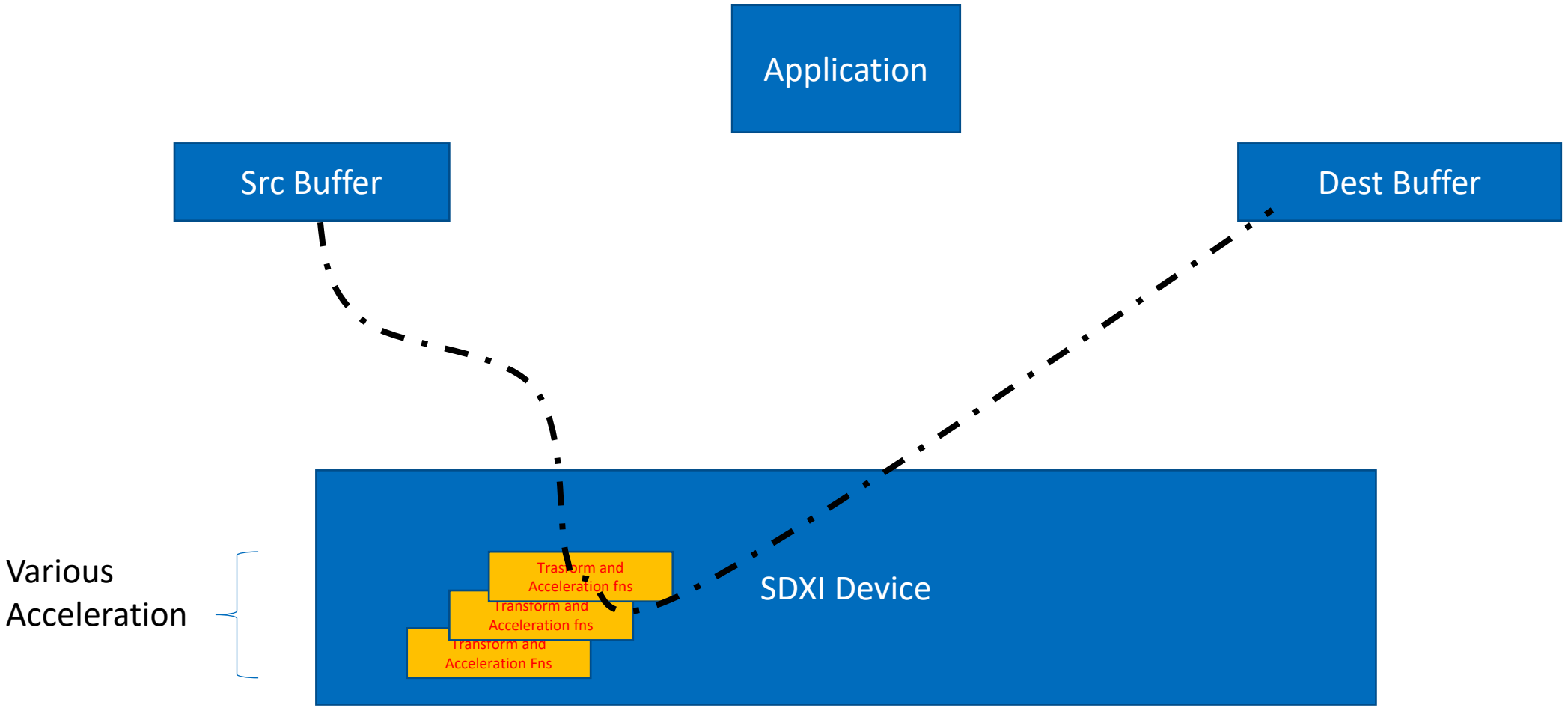
Agenda

- Compute, IO, Memory Bubble
  - Current Memory to Memory Data Movement Standard
- Use Cases
  - Application Patterns and benefits of Data Movement & Acceleration
- SNIA SDXI TWG
  - Goals and Tenets
  - A brief introduction to the internals of SDXI Specification
- SDXI Futures/SDXI v1.1
- SDXI Community/Ecosystem
- Summary

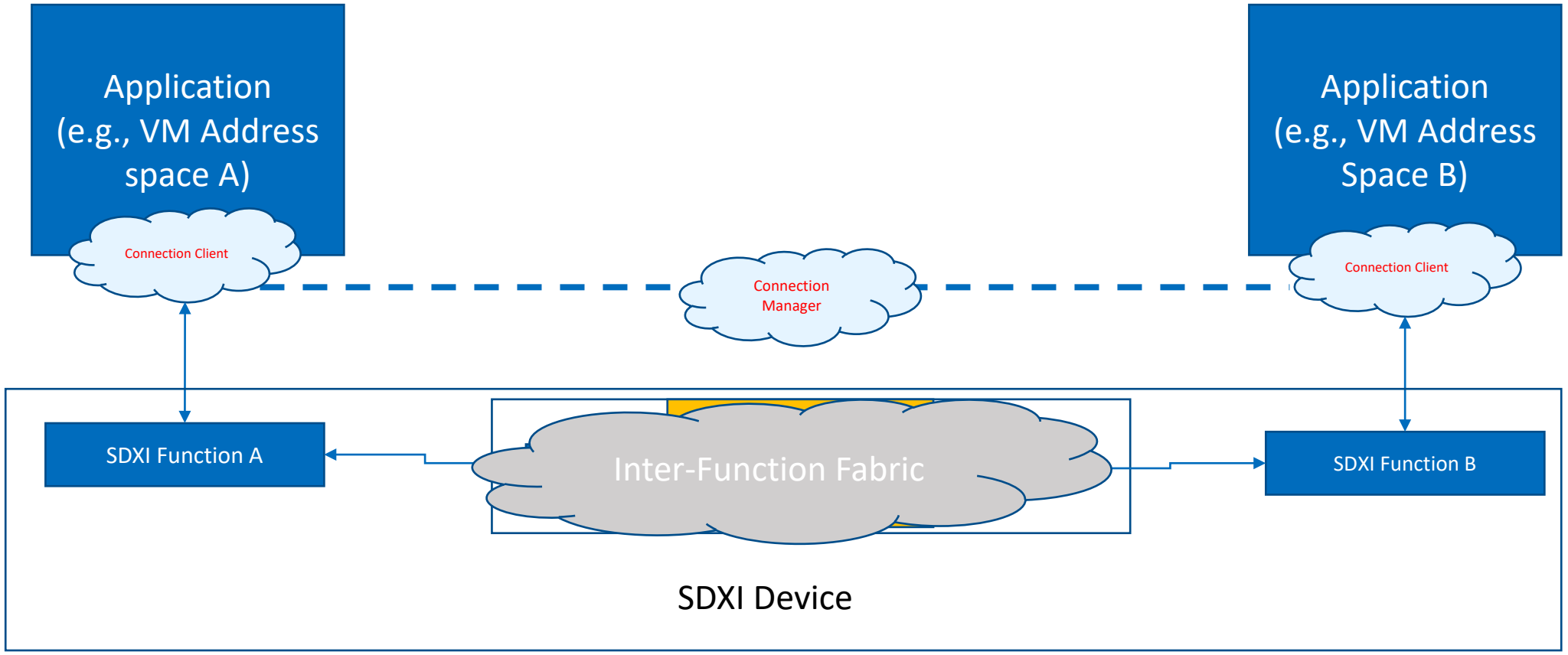
SDXI v1.1 investigations

- 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

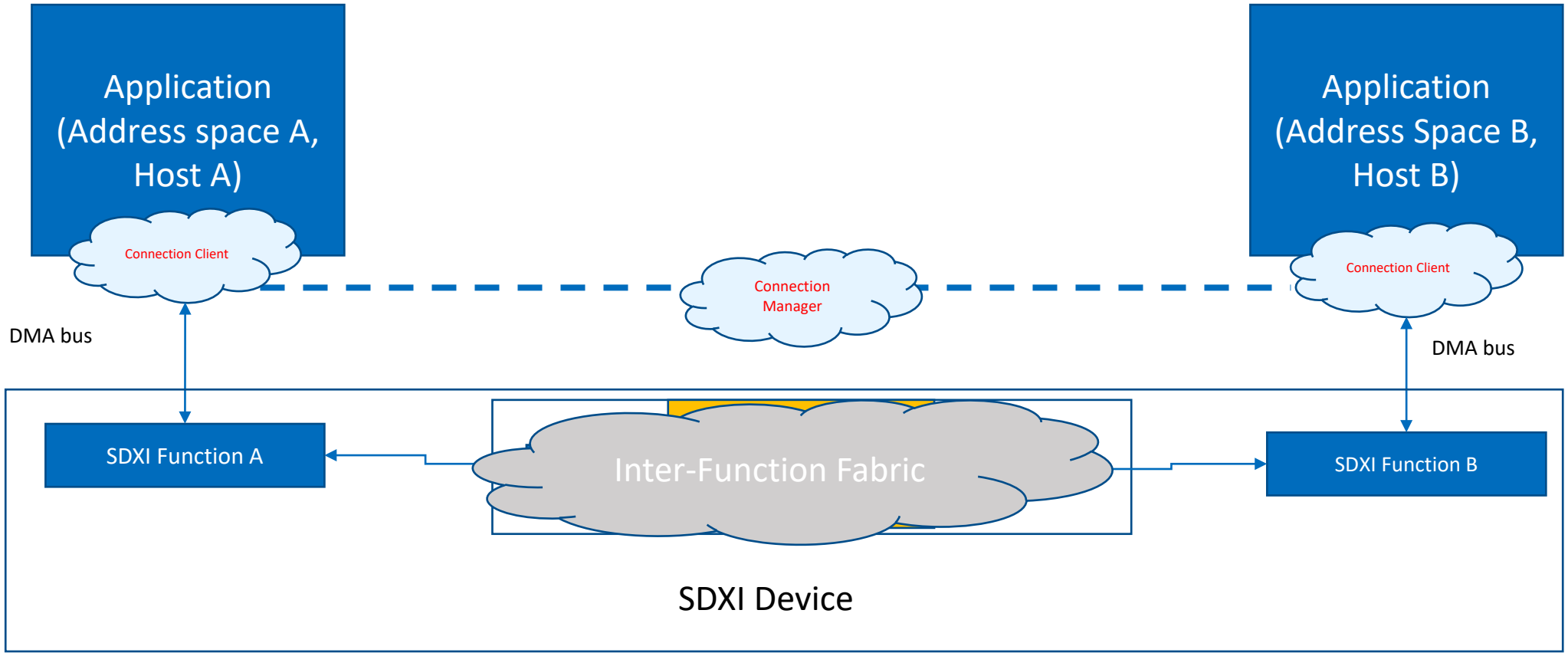
New Data Mover Operations



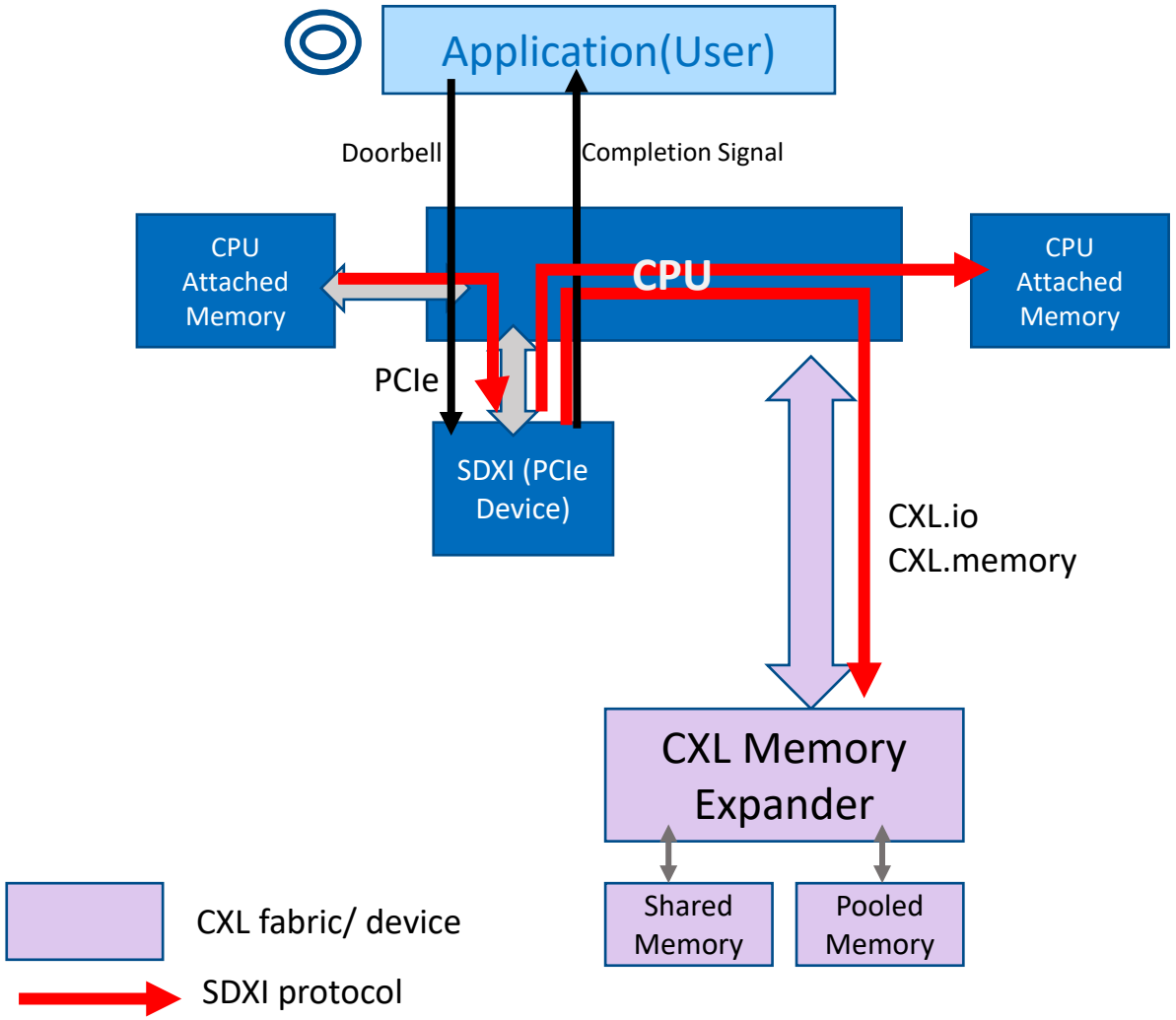
Connection Manager

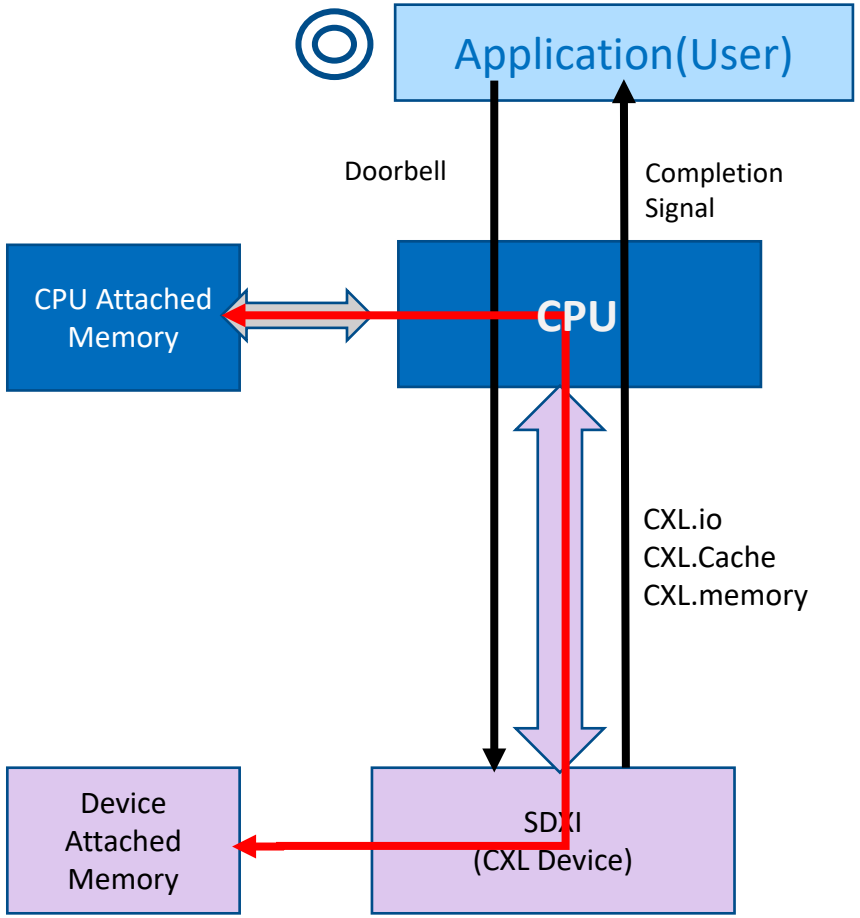


Host to Host

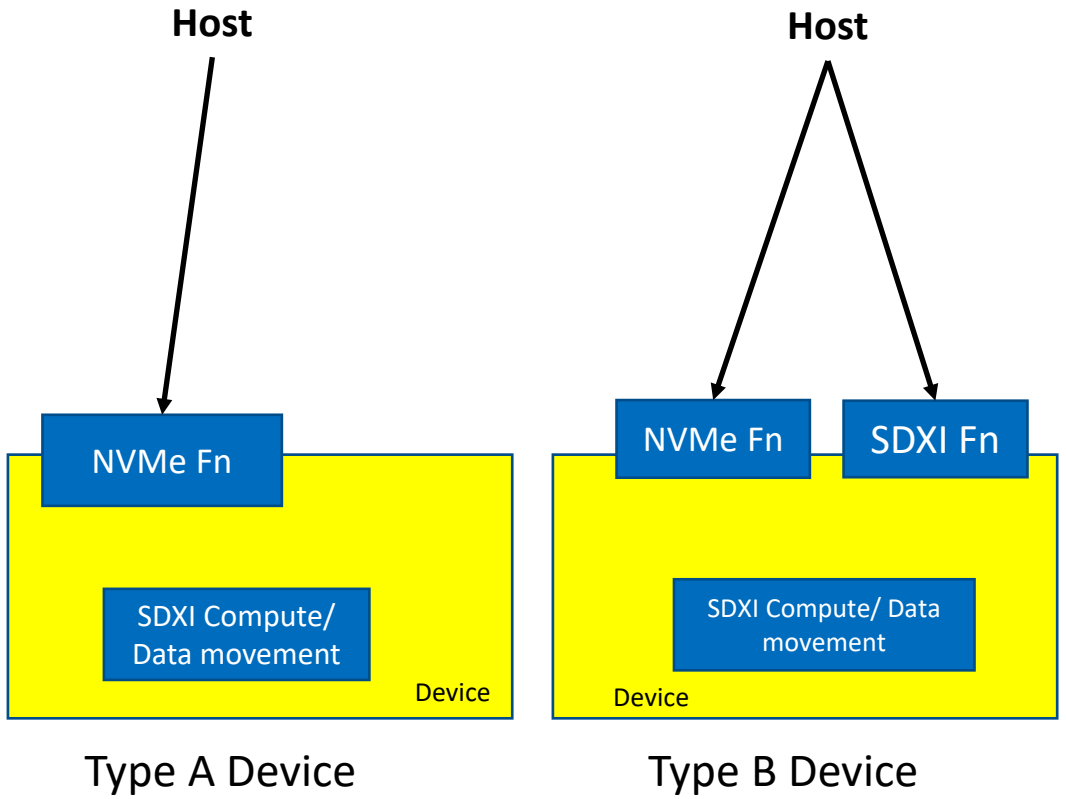


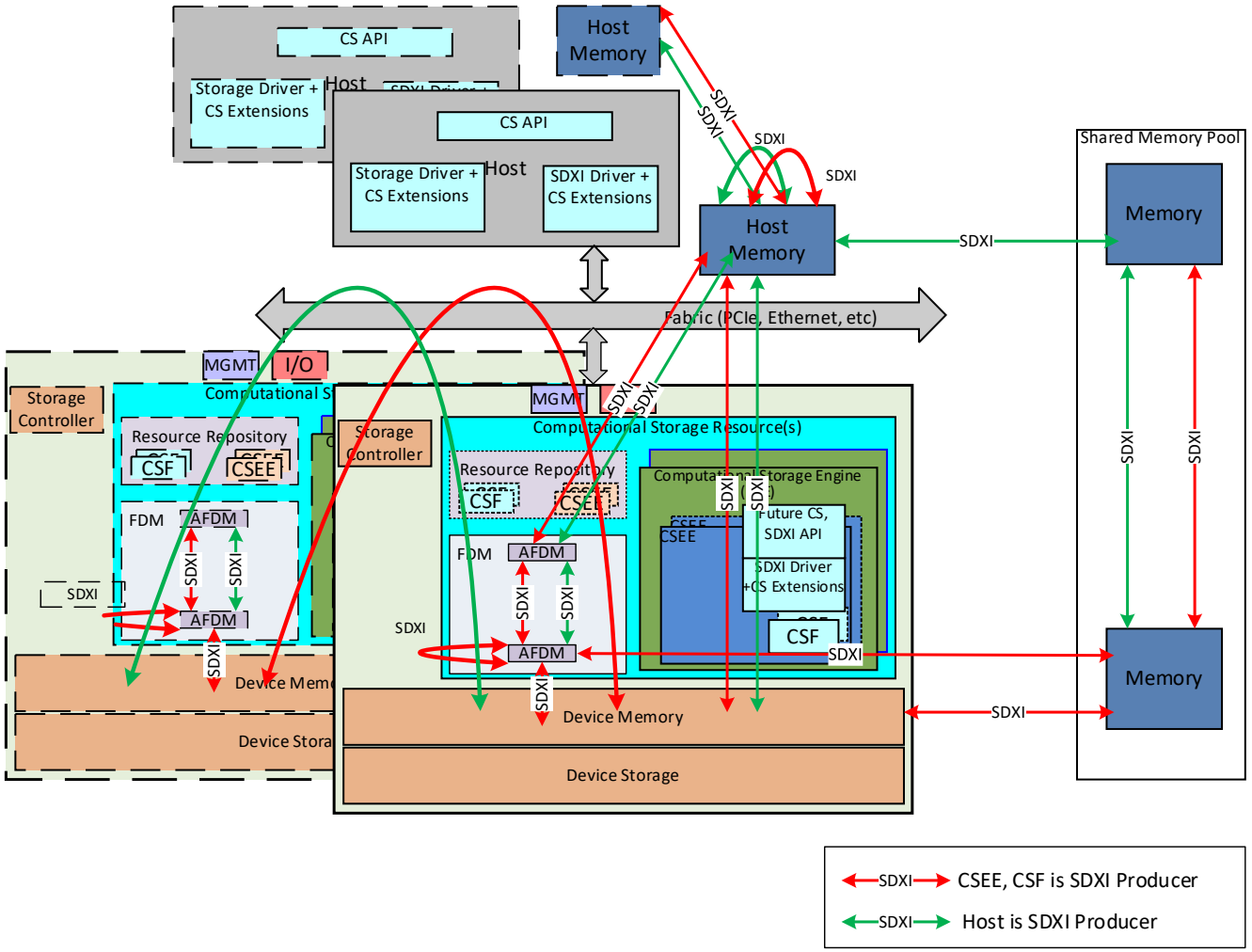
CXL based Architectures





Computational Storage, NVMe, and SDXI





Agenda

- Compute, IO, Memory Bubble
  - Current Memory to Memory Data Movement Standard
- Use Cases
  - Application Patterns and benefits of Data Movement & Acceleration
- SNIA SDXI TWG
  - Goals and Tenets
  - A brief introduction to the internals of SDXI Specification
- SDXI Futures/SDXI v1.1
- SDXI Community/Ecosystem
- Summary

Additional SDXI Ecosystem activities

- SDXI Software group within SDXI TWG
  - 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

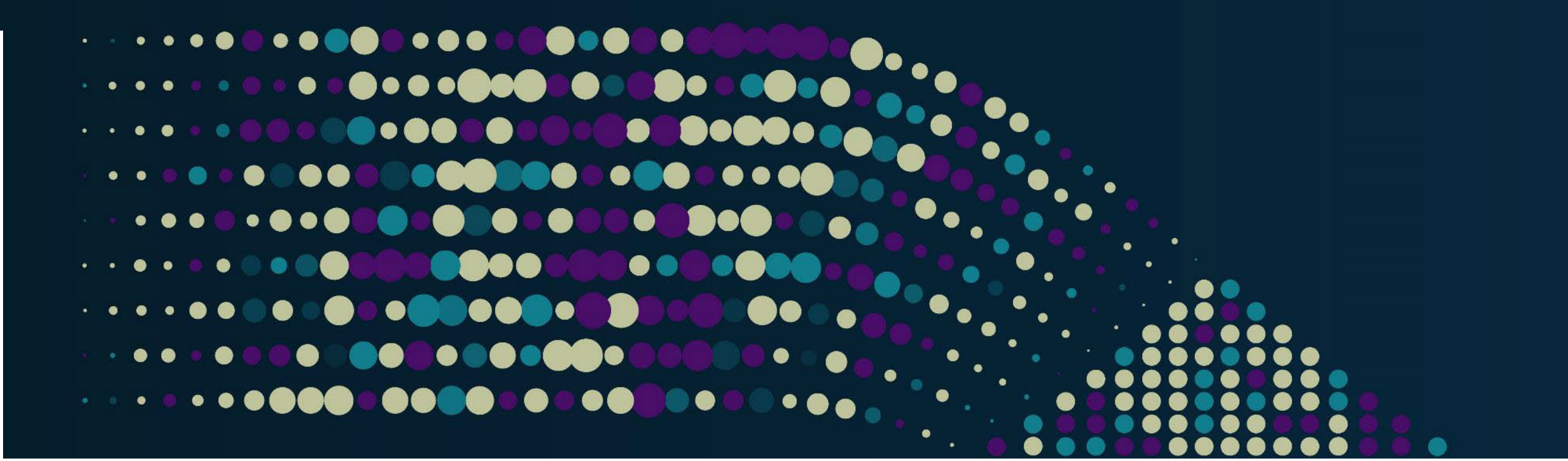
Agenda

- Compute, IO, Memory Bubble
  - Current Memory to Memory Data Movement Standard
- Use Cases
  - Application Patterns and benefits of Data Movement & Acceleration
- SNIA SDXI TWG
  - Goals and Tenets
  - A brief introduction to the internals of SDXI Specification
- SDXI Futures/SDXI v1.1
- SDXI Community/Ecosystem
- Summary

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 is working v1.1 specification
- SDXI Software work
  - SDXI TWG is working on libsdxi, an OS-agnostic library to help user space applications use SDXI accelerated data movement operations
- Learn More:
  - https://www.snia.org/sdxi

Q&A



Please take a moment to rate this session.

Your feedback is important to us.

