STORAGE DEVELOPER CONFERENCE

SDC 22 | Fremont, CA
September 12-15, 2022

BY Developers FOR Developers

A SNIA. Event

SDXI Internals and its Journey Towards Standardizing Memory to Memory Data Movement

Presented by: William Moyes

AMD Fellow, End-to-End Server Architecture

SNIA Legal Notice

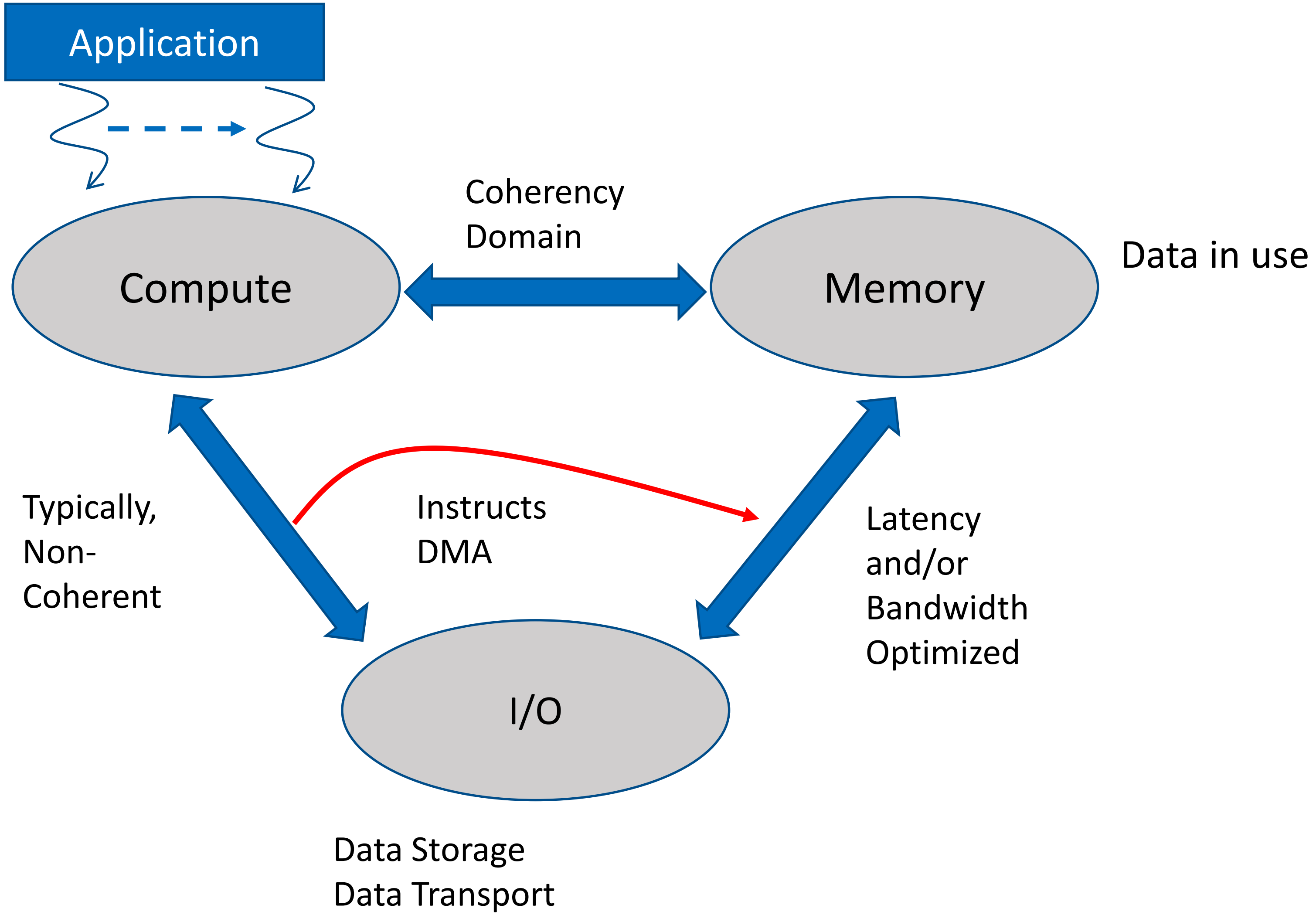
- This presentation is a project of SNIA, and the material contained in this presentation is copyrighted by the SNIA unless otherwise noted.
- SNIA member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used 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.

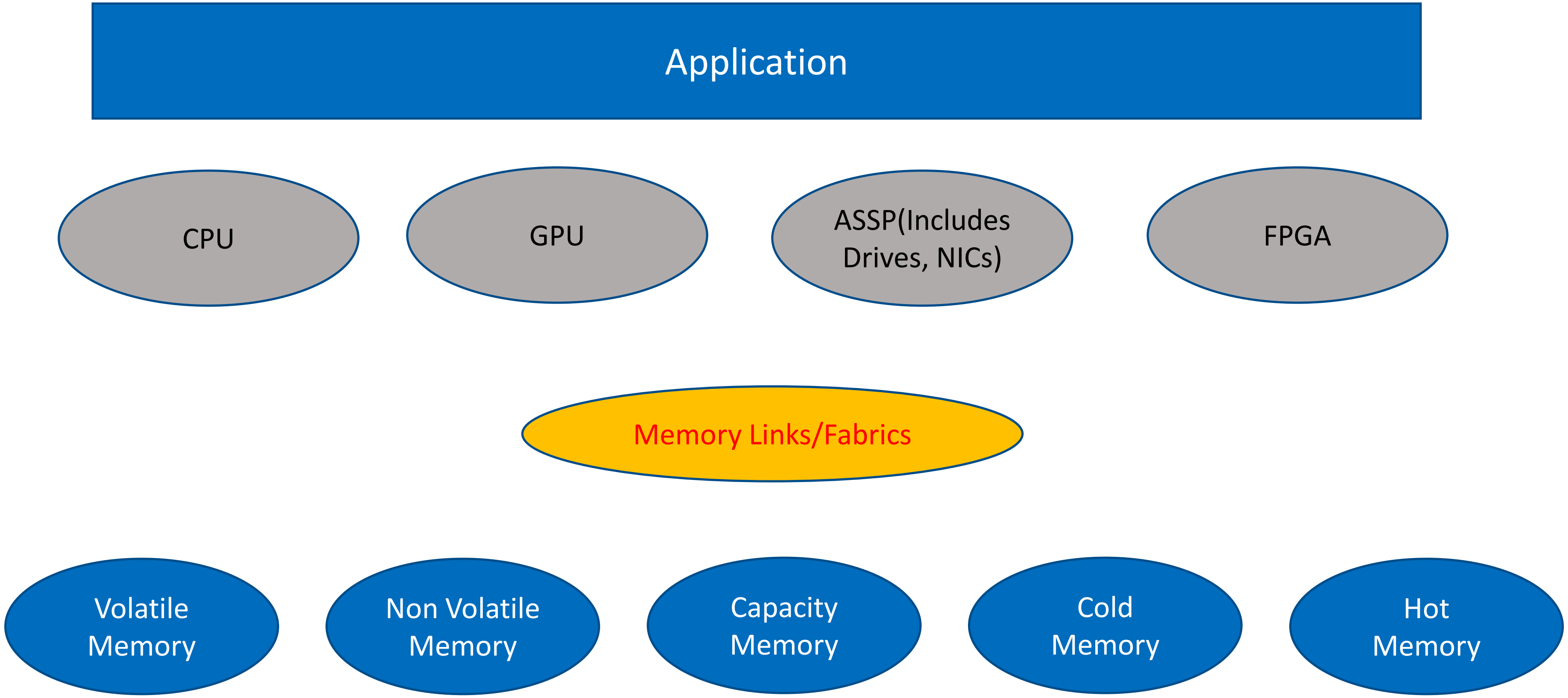
Agenda

- 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 Community
  - SDXI Futures
  - References, Links, and Announcements

Legacy Compute, Memory, IO Bubbles



Emerging Bubbles



Shared Design constraints

- Latency
- Bandwidth
- Coherency
- Control

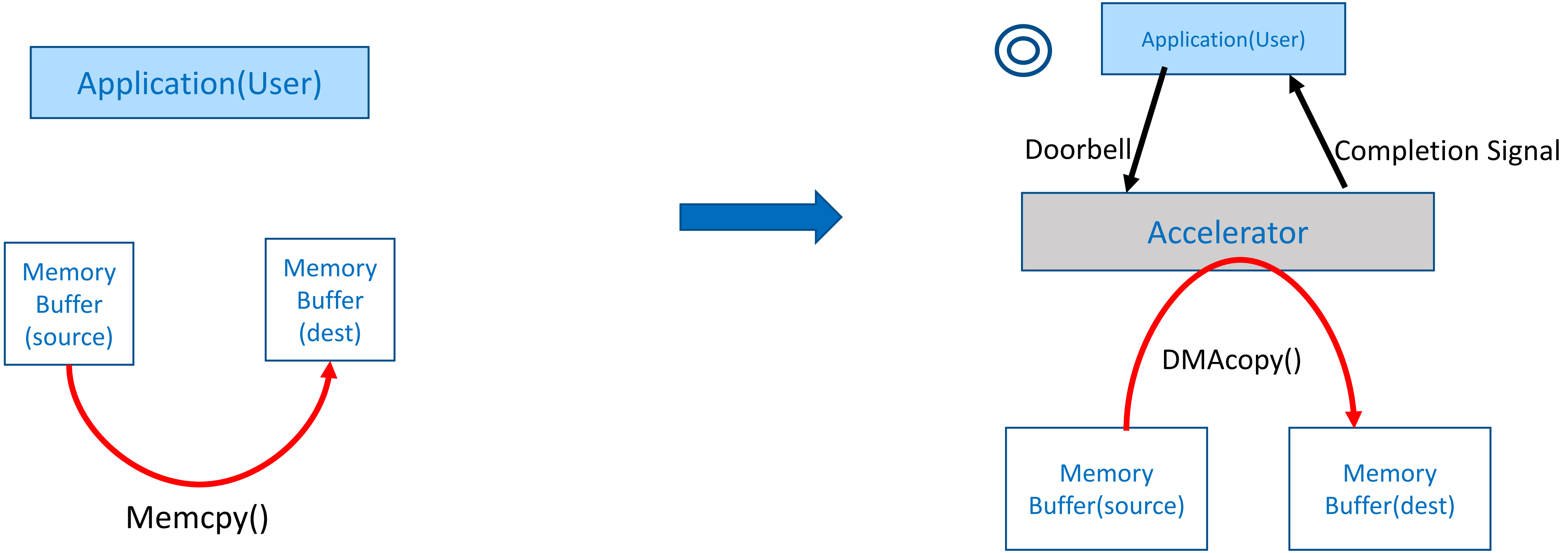
Current Data Movement Standard

- 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 Community
  - SDXI Futures
  - References, Links, and Announcements

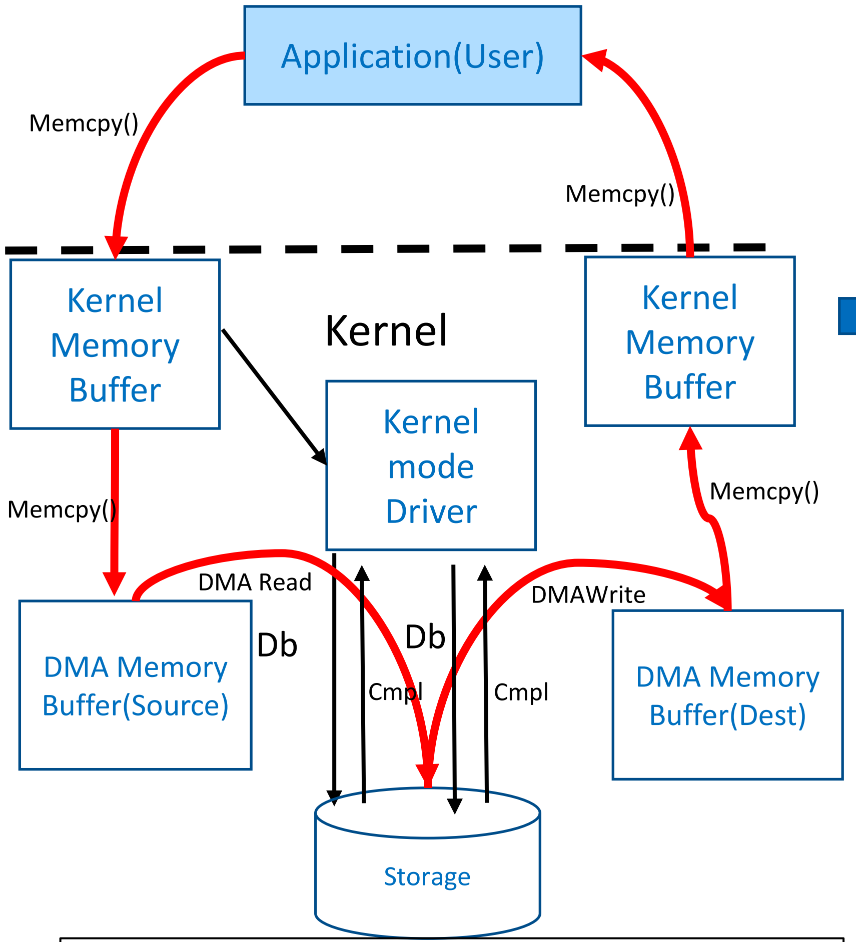
Application Pattern 1 (Buffer Copies)



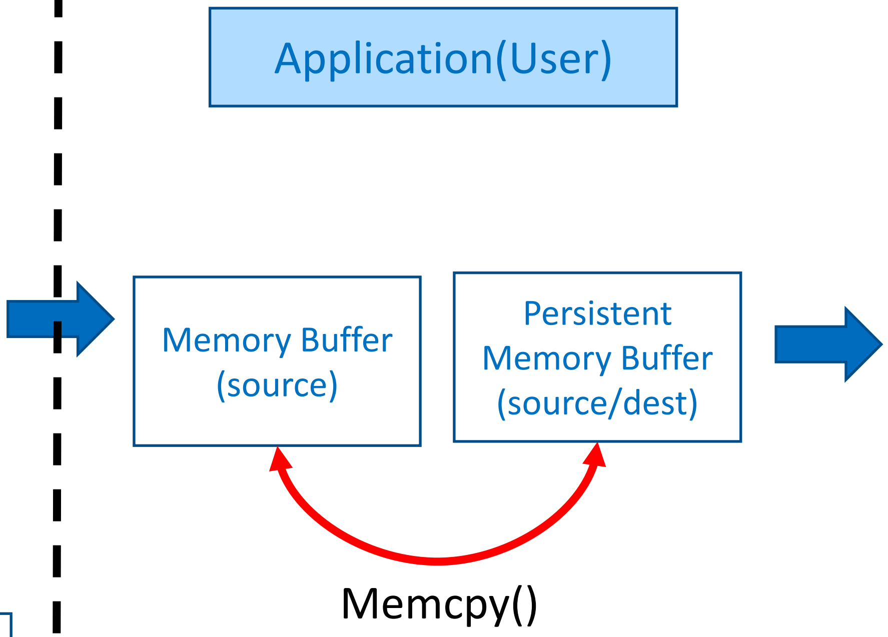
- Takes away from application performance

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

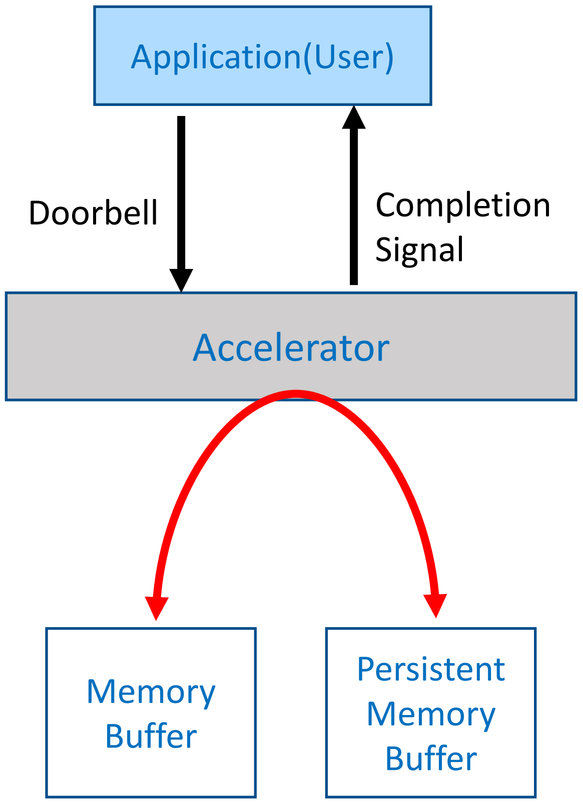
Application Pattern 2



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

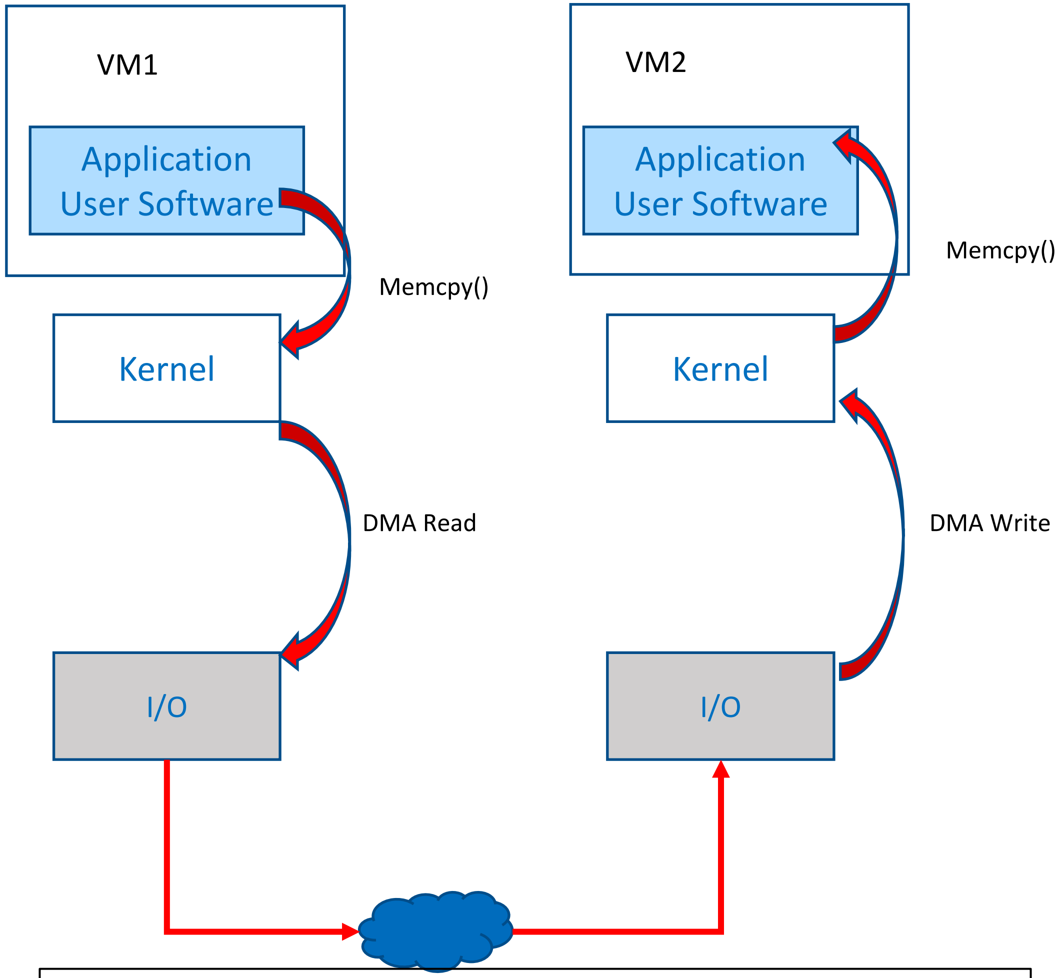


- Reduced buffer copies but still takes away from application performance

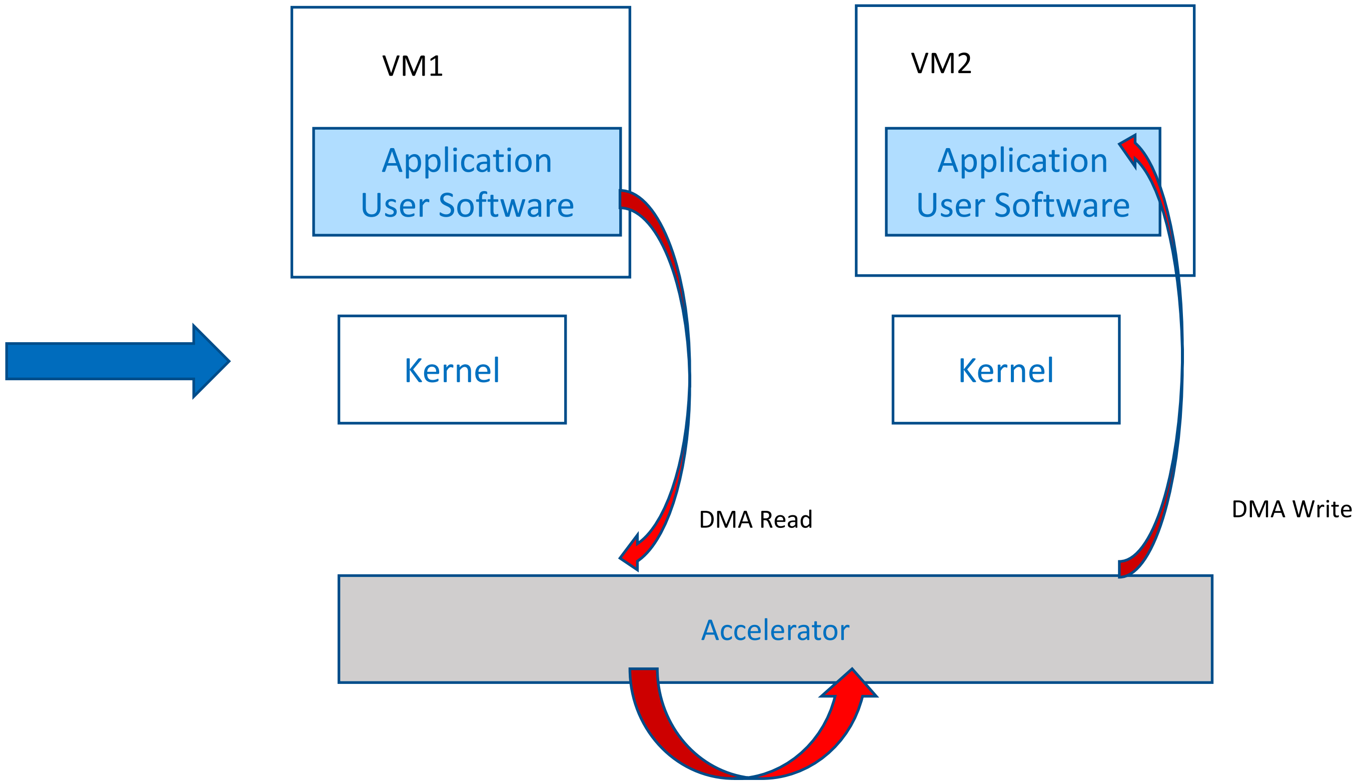


- Reduced buffer copies
- HW based offloaded memory copies

Application Pattern 3

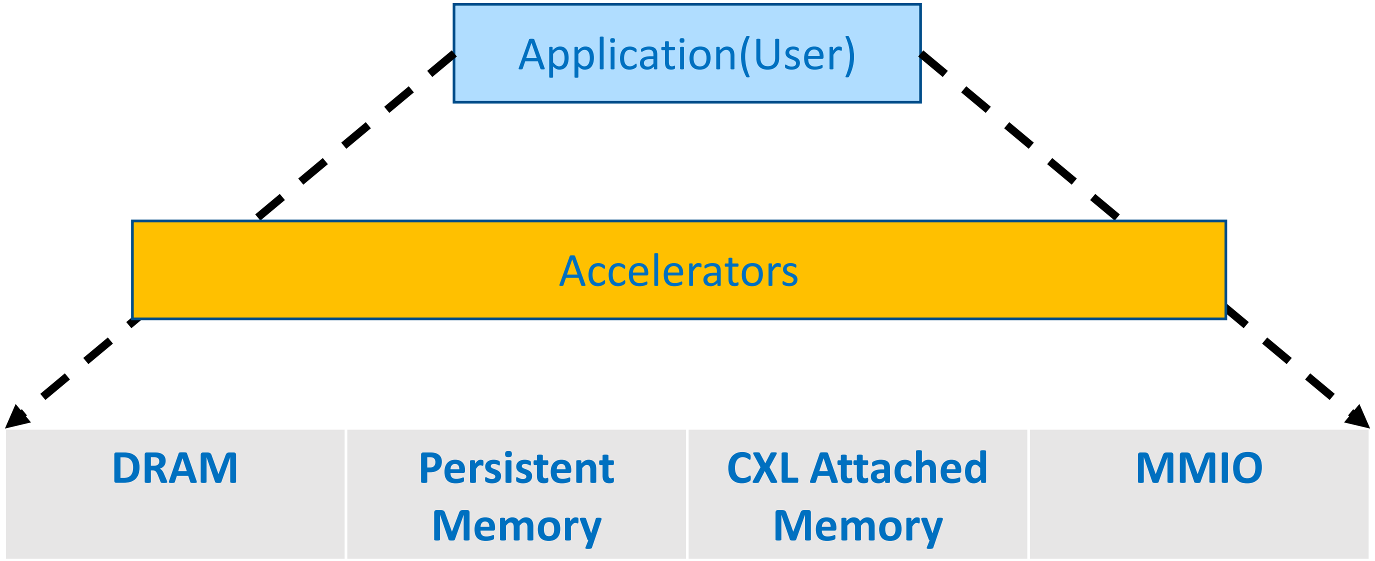


- Context isolation layers introduce multiple buffer copies



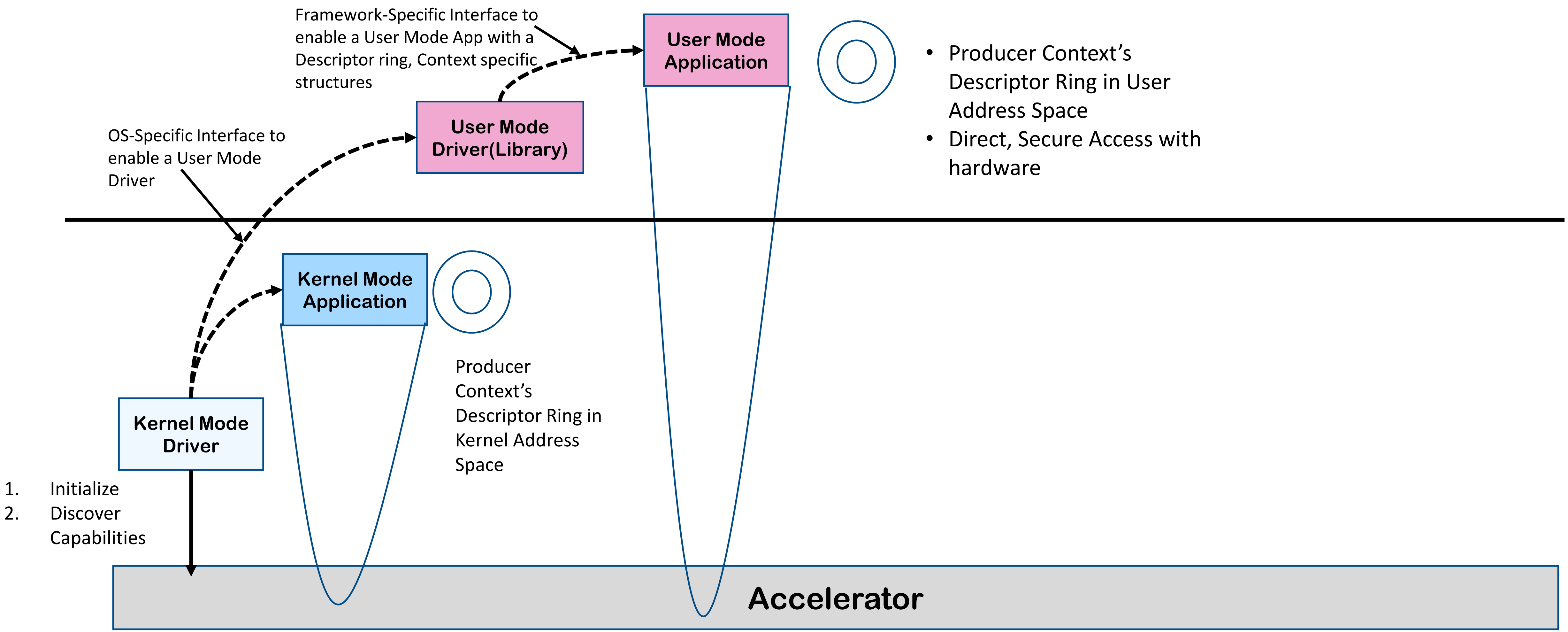
- Best of both: Context isolation layers and optimized HW based memory buffer copies

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



Framework-Specific Interface to enable a User Mode App with a Descriptor ring, Context specific structures

User Mode Application

User Mode Driver(Library)

OS-Specific Interface to enable a User Mode Driver

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

Kernel Mode Application

Producer Context's Descriptor Ring in Kernel Address Space

Kernel Mode Driver

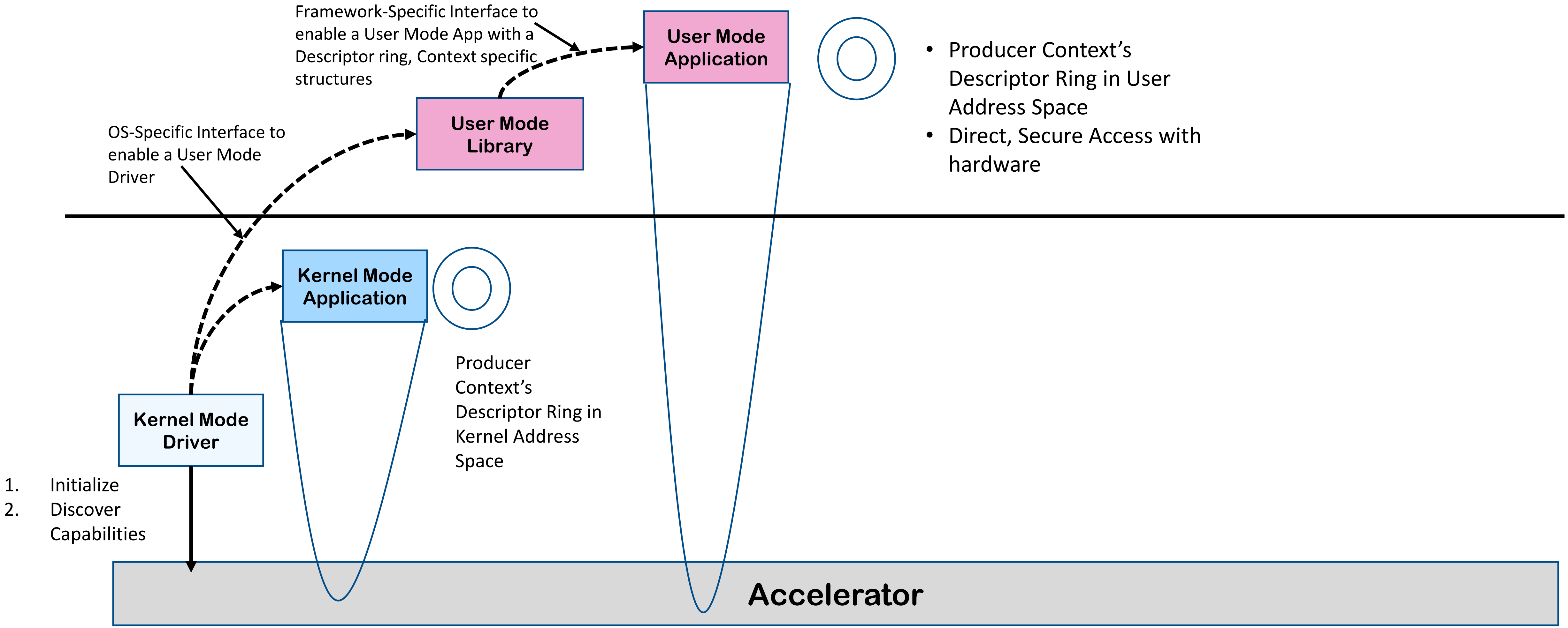
1. Initialize
2. Discover Capabilities

Accelerator

Direct HW Access, Access Memory Tiers

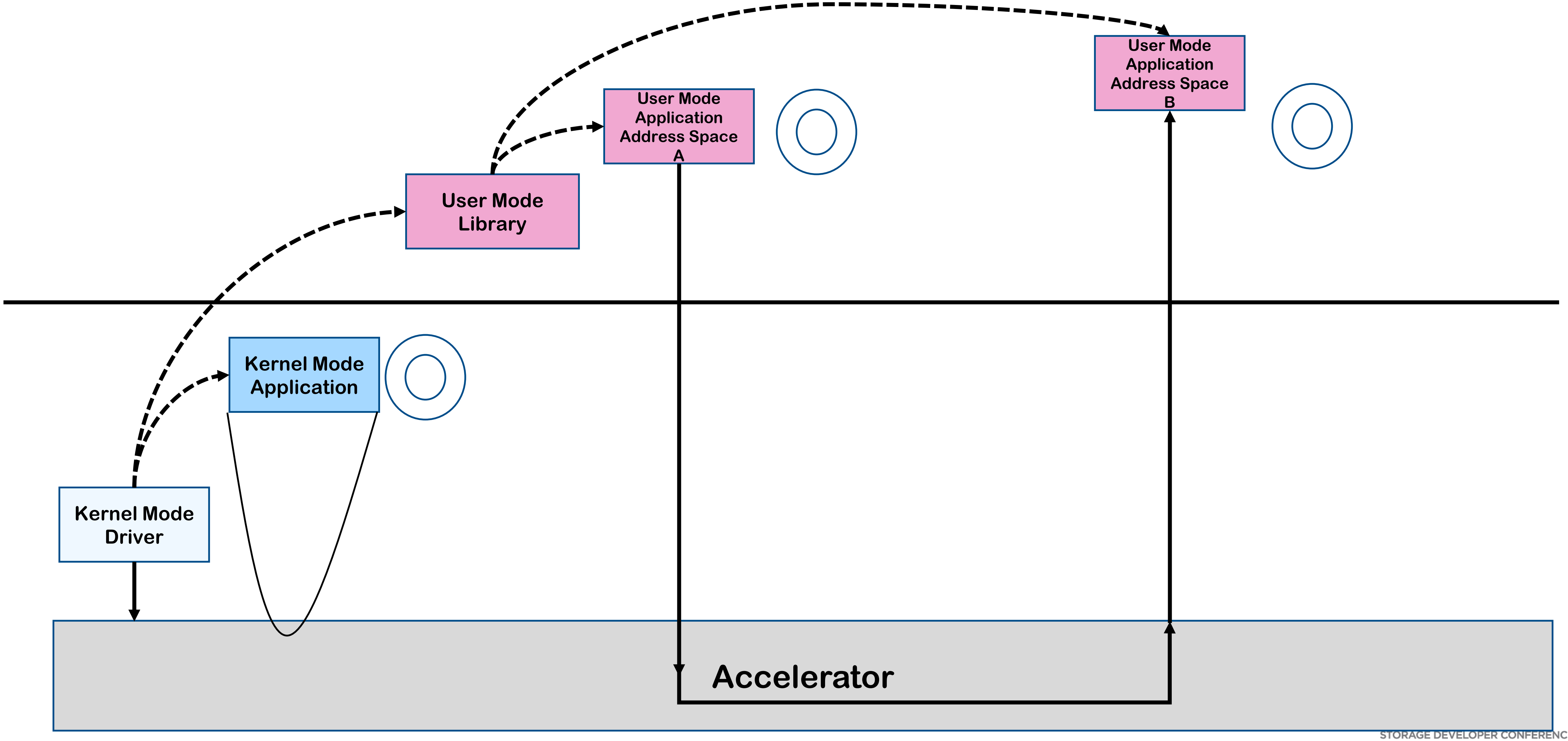
DRAM	PMEM	MMIO	CXL Memory

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


Framework-Specific Interface to enable a User Mode App with a Descriptor ring, Context specific structures
User Mode Application
User Mode Library
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

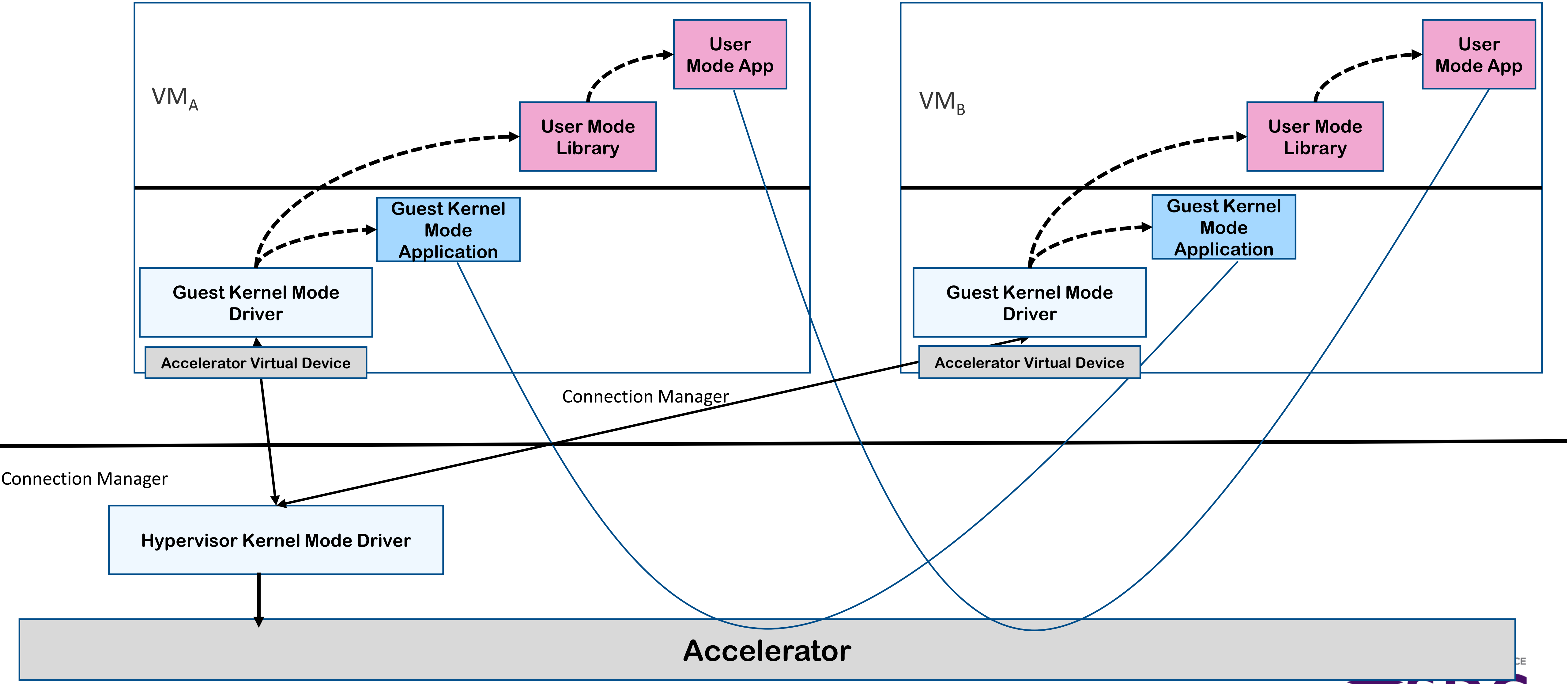
- Producer Context's Descriptor Ring in User Address Space
- Direct, Secure Access with hardware

Scale Baremetal Apps – Multi-Address Space



User Mode Application Address Space B
User Mode Application Address Space A
User Mode Library
Kernel Mode Application
Kernel Mode Driver
Accelerator

Scale with Compute Virtualization– Multi-VM address space



Agenda

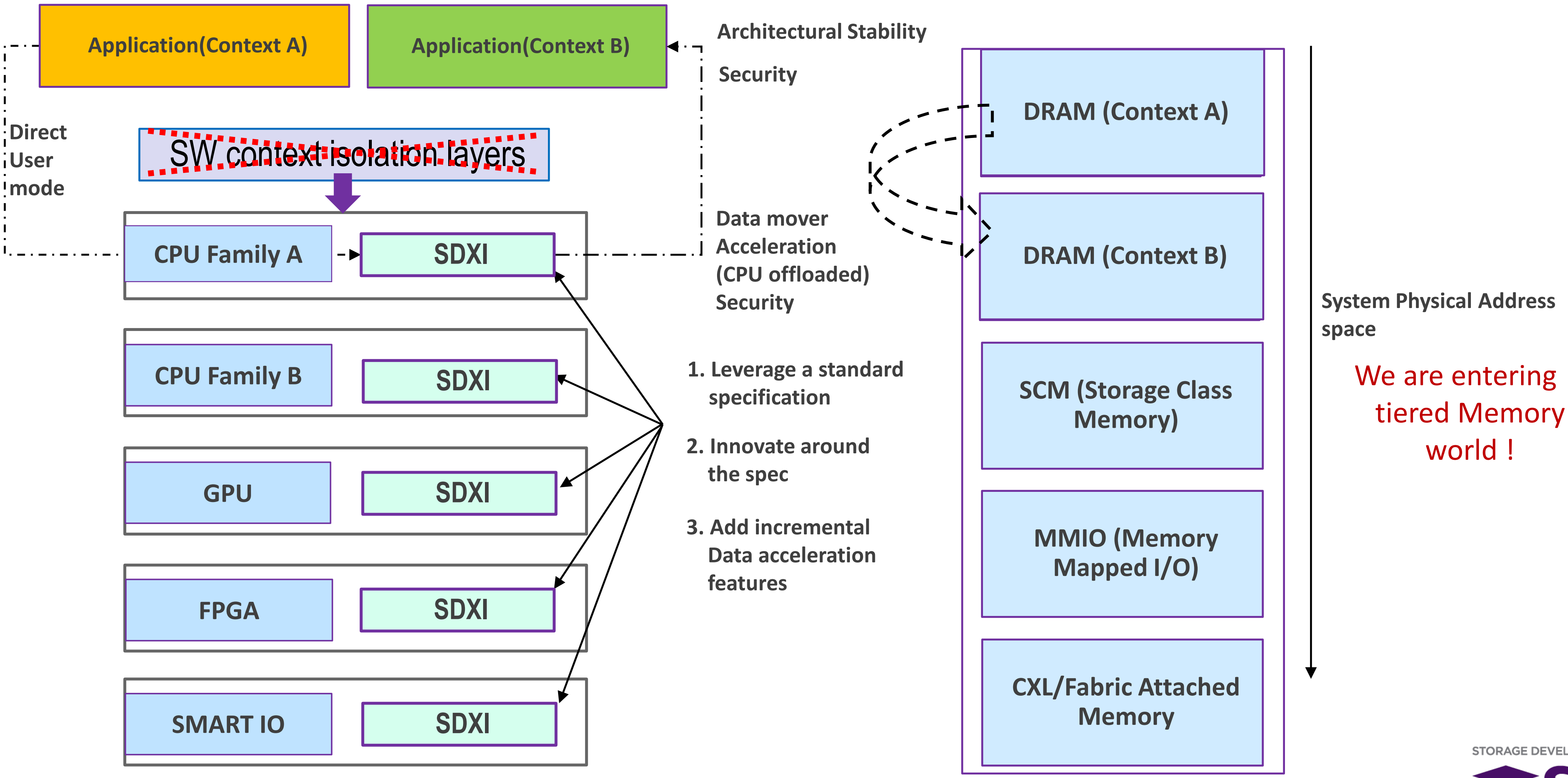
- Compute, IO, Memory Bubble
  - Current Memory to Memory Data Movement Standard
- Use Cases
  - Application Patterns and benefits of Data Movement & Acceleration
- SNIA SDXI TWG
  - Goals and Tenets
  - A brief introduction to the internals of SDXI Specification
  - SDXI Community
  - SDXI Futures
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

SDXI(Smart Data Accelerator Interface)

- Smart Data Accelerator Interface (SDXI) is a proposed 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
  - 28 member companies, 80+ individual members

SDXI Memory-to-Memory Data Movement



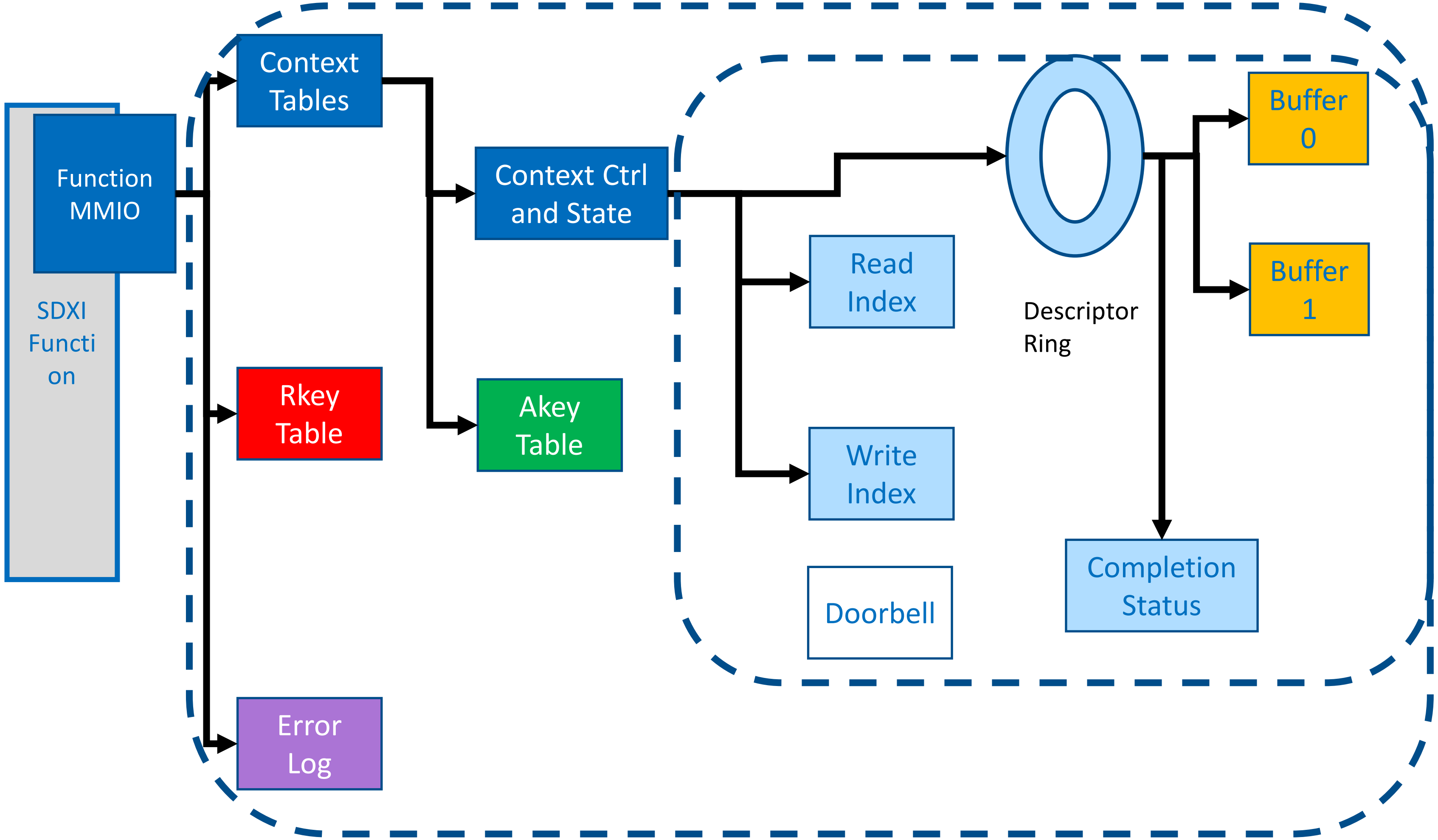
1. Leverage a standard specification
2. Innovate around the spec
3. Add incremental Data acceleration features

We are entering a tiered Memory world !

SDXI Design Tenets

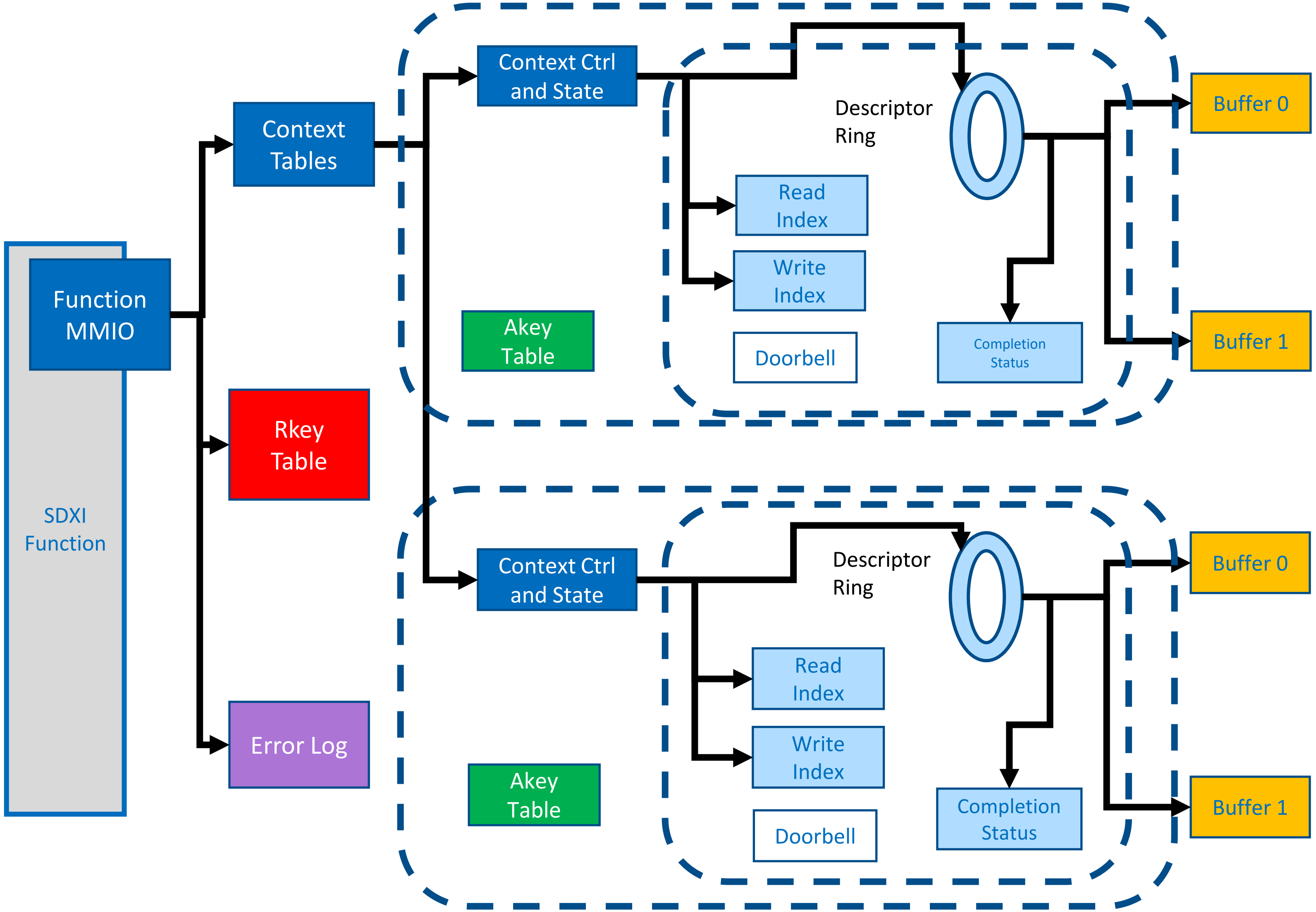
- Data movement between different address spaces.
  - Includes user address spaces, different virtual machines
- Data movement without mediation by privileged software.
  - Once a connection has been established.
- Allows abstraction or virtualization by privileged software.
- Capability to quiesce, 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.

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

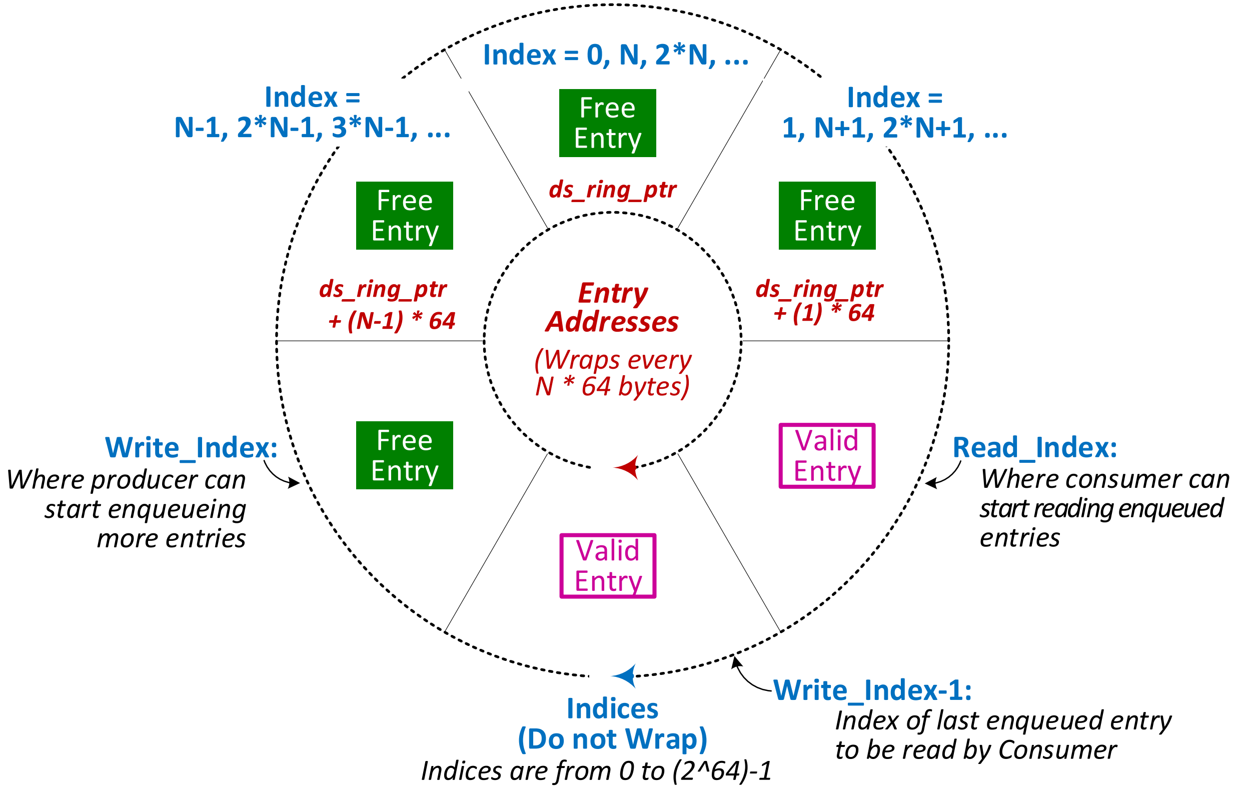
Memory Structures(2) – Multiple Contexts



- 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

Descriptor Ring

Ring starts at memory location ds_ring_ptr
N = ds_ring_sz (Number of entries in Queue)



Index = 0, N, 2*N, …

Free Entry

ds_ring_ptr

Index = 1, N+1, 2*N+1, …

Free Entry

*ds_ring_ptr + (1) * 64*

Index = N-1, 2*N-1, 3*N-1, …

Free Entry

*ds_ring_ptr + (N-1) * 64*

Entry Addresses
*(Wraps every N * 64 bytes)*

Free Entry

Valid Entry

Valid Entry

Write_Index: *Where producer can start enqueueing more entries*

Read_Index: *Where consumer can start reading enqueued entries*

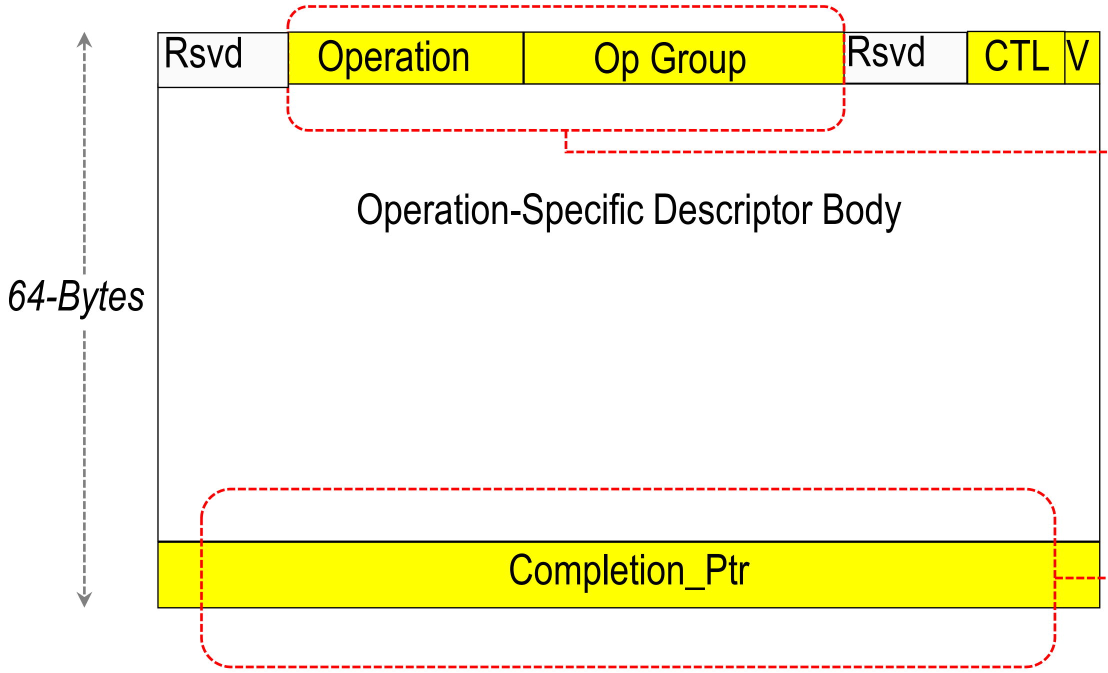
Write_Index-1: *Index of last enqueued entry to be read by Consumer*

Indices (Do not Wrap)
Indices are from 0 to (2^64)-1

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
Atomic	Connect	Others …

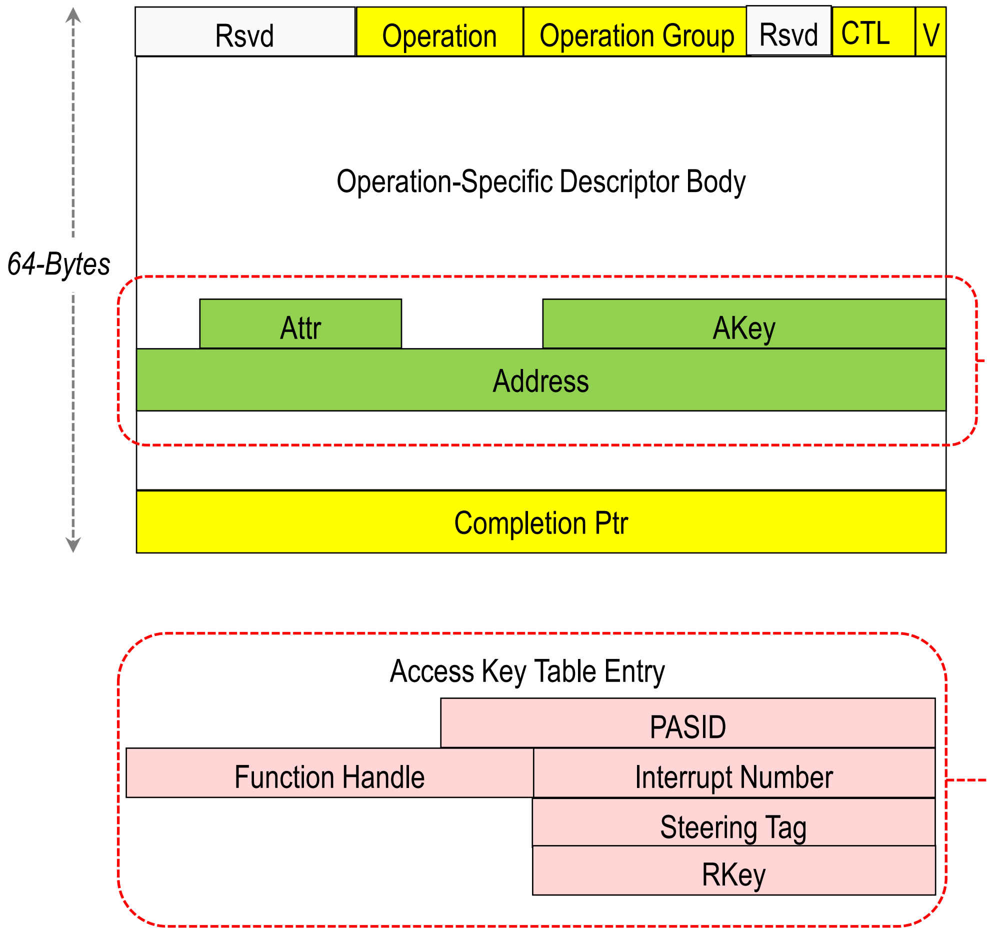
DMA:	Copy, RepCopy, WriteImm
Atomic:	Bitwise Ops, Add, Sub, Swap, Min, Max, CmpSwap, etc
Admin:	Start/Stop/Update/Sync, Interrupt Function & Contexts (easily virtualizable)
Connect:	Connect VMs through SDXI functions (easily virtualizable)

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

A Standard Descriptor Format (2)



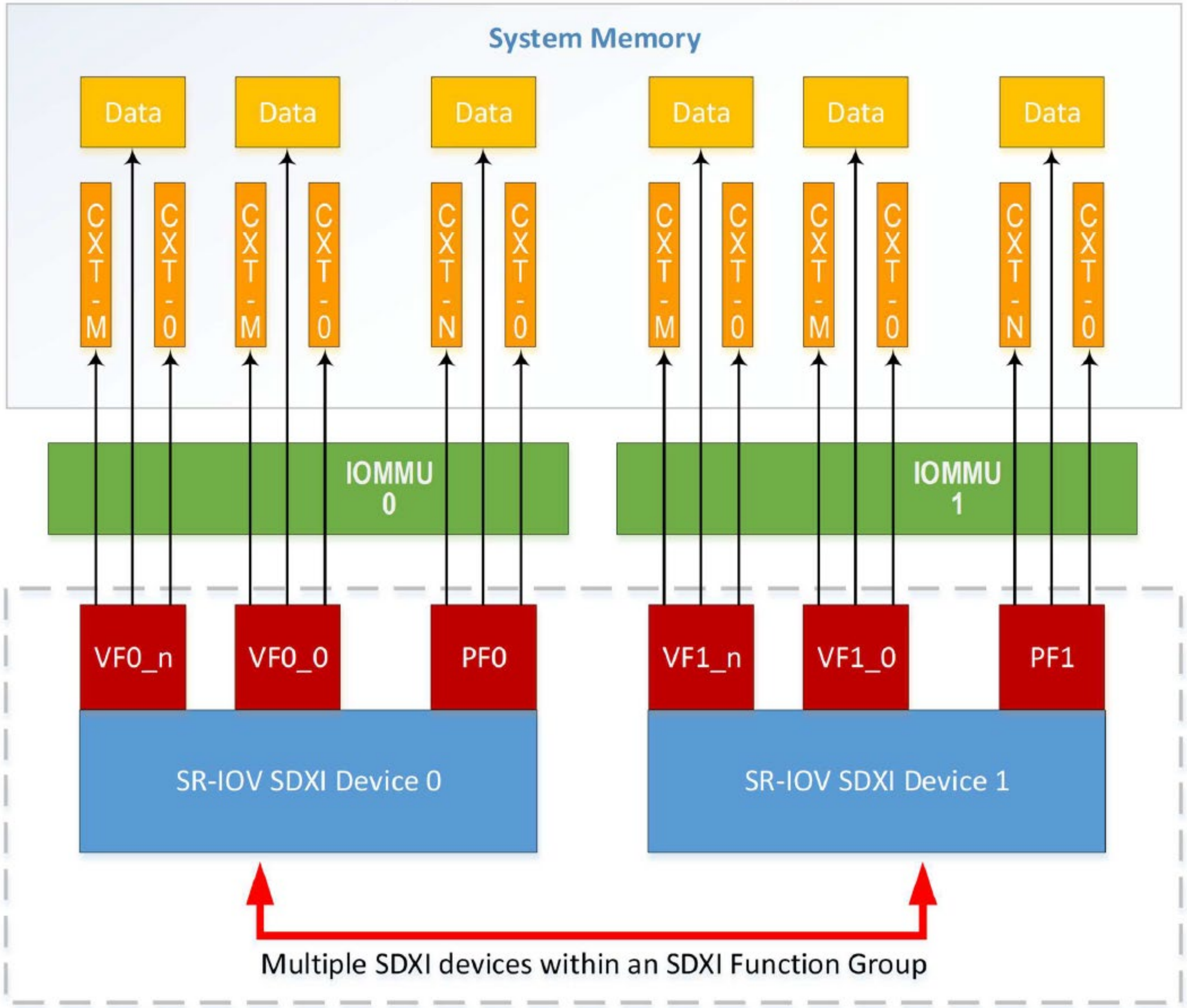
A memory location is always specified as a triple:

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

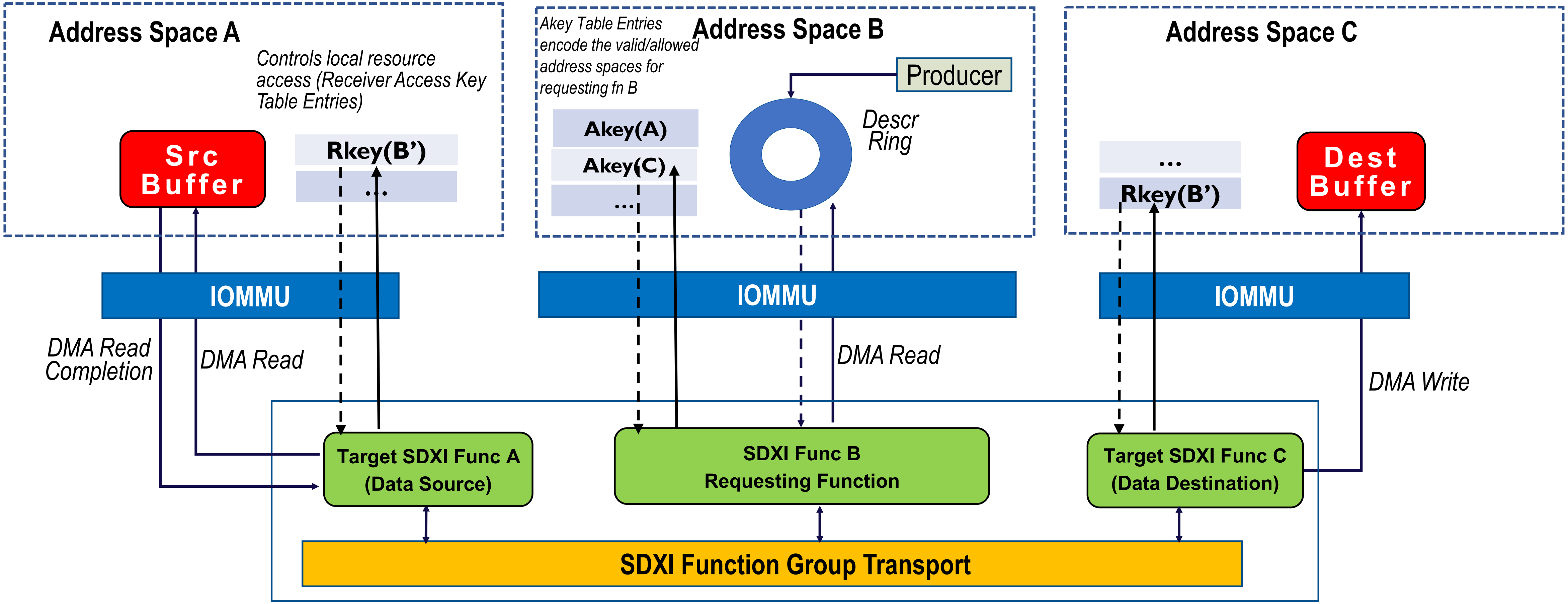
Generated Address can be HPA, HVA, GPA, GVA and always translated through IOMMU.

- An AKey table entry encodes all valid address spaces, PASIDs and interrupts available to the function context.
- 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. RKey is target side control.

Contexts and SDXI Function Groups



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)
…
Descr 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 Function Group Transport

Active Community of SDXI TWG members

- TWG members have contributed in various ways towards improving the specification since initial contribution by founding members
- The contributions have resulted in improvements in many areas like -
  - Architectural behavior of Administrative Start/Stop operations
  - Architectural Context States and Function States
  - Cache Injection mechanisms
  - Deterministic discovery of SDXI Function Groups
  - Rich Completion Status and Error Reporting
  - Improved mechanisms to control local memory access from remote functions
  - Usage models
  - Interconnect agnostic specification language

Active Contributors

- Curtis Ballard, HPE
- Beau Beachamp, MemVerge
- Richard Brunner, VMware
- Xiangping Chen, Dell
- Don Dutile, IBM
- Paul Hartke, AMD
- Shyam Iyer, Dell Inc
- Travis Hamilton, Arm
- Brian Hirano, Micron
- Frederick Knight, NetApp
- Santosh Kumar, SK Hynix
- James Leighton, Western Digital
- Bill Martin, Samsung
- John Maroney, Micron
- J Metz, AMD
- William Moyes, AMD
- Philip Ng, AMD
- Murali Ravirala, Microsoft
- Dwight Riley, HPE
- Alexandre Romana, Arm
- Glen Sescila, Dell Inc
- Paul Von Stamwitz, Fujitsu
- Jason Wohlgemuth, Microsoft

What to expect: SDXI Futures

- Release v1.0
- Plan post v1.0 activities. The current charter includes:
  - New data mover operations for smart acceleration
  - Data mover operations involving persistent memory targets
  - Cache coherency models for data movers
  - Security Features involving data movers
  - Management architecture for data movers(includes connection manager)
- Some additional discussion topics being considered post-v1.0
  - QoS improvements
  - Latency improvements
  - RAS improvements
  - CXL related discussions
  - Heterogenous environments

Draft: Subject to Change

Links

- SDXI Specification v0.9-rev1 available for Public Review
  - https://www.snia.org/tech_activities/publicreview
- SNIA SDXI Page
  - https://www.snia.org/sdxi
- PM + CS Summit 2021
  - https://www.snia.org/educational-library/new-path-better-data-movement-within-system-memory-computational-memory-sdxi
- SDC 2020 presentation
  - https://www.youtube.com/watch?app=desktop&v=iv2GUfnxG-A
- In memory compute summit
  - https://www.youtube.com/watch?v=iv2GUfnxG-A
- SDC 2021 panel session
  - https://www.youtube.com/watch?v=PrlQZF2a4YI
- New Subgroup! SDXI + CS Subgroup
  - Membership Criteria:
    - Should be a member of SNIA's SDXI TWG & CS TWG

Conclusion

- Data in use performance needs are increasing
- Variety of accelerator and memory technologies, standardizing memory to memory data movement and acceleration is gaining ground
- SNIA's SDXI (Smart Data Accelerator Interface) TWG standardizing memory to memory data movement and acceleration
- v0.9-rev1 version of the specification out for public review
  - Various use cases, features enabled
  - https://www.snia.org/tech_activities/publicreview
- TWG members have discussed their motivation towards contributing to this specification
- Enables persistent memory technologies and computational storage use cases

Q&A

Please take a moment to rate this session.

Your feedback is important to us.