SDC
STORAGE DEVELOPER CONFERENCE
SNIA SANTA CLARA, 2016

RDMA VERBs Extensions for Persistency and Consistency

Idan Burstein

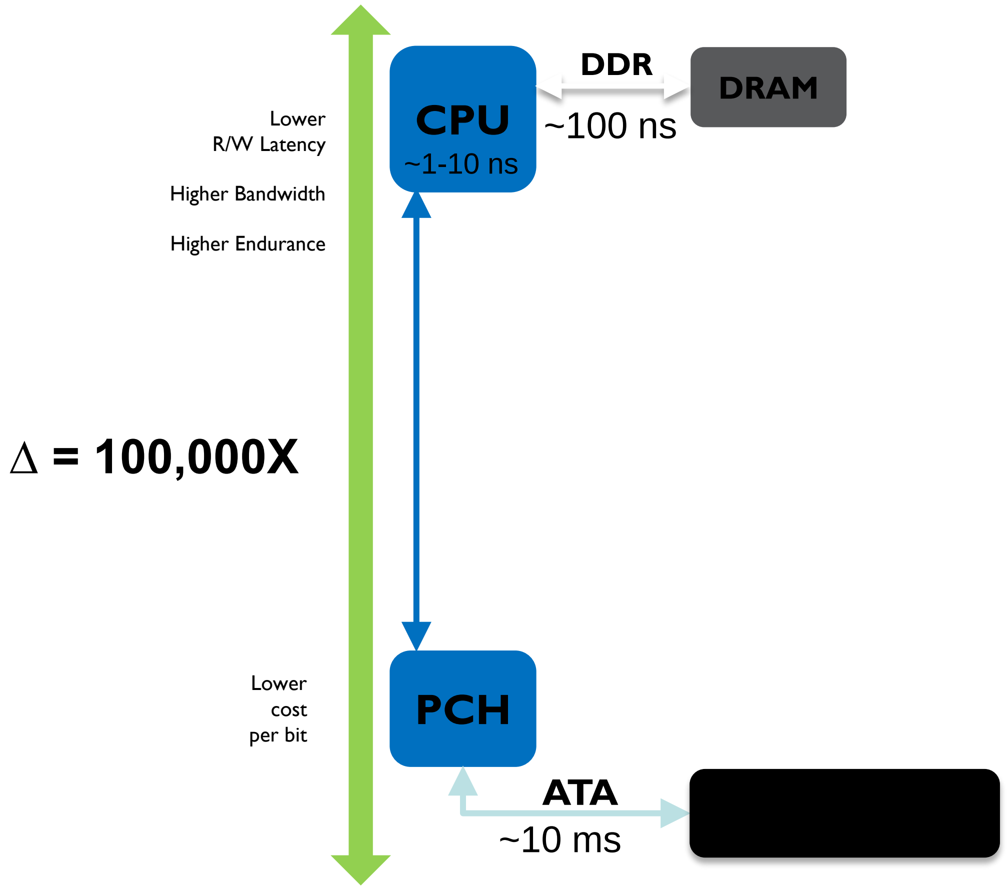
Mellanox Technologies

Outline

- Emerging NVM Technologies
- The NVM Programming Model
- RDMA – What? Why? How?
- RDMA Storage Protocols
- Latency Breakdown
- RDMA to Persistent Memory
- RDMA Reliability Extensions

Emerging NVM Technologies

The Past: Nonvolatile Memories in Server Architectures



Lower
R/W Latency

Higher Bandwidth

Higher Endurance

Δ = 100,000X

Lower
cost
per bit

CPU

~1-10 ns

DDR

~100 ns

DRAM

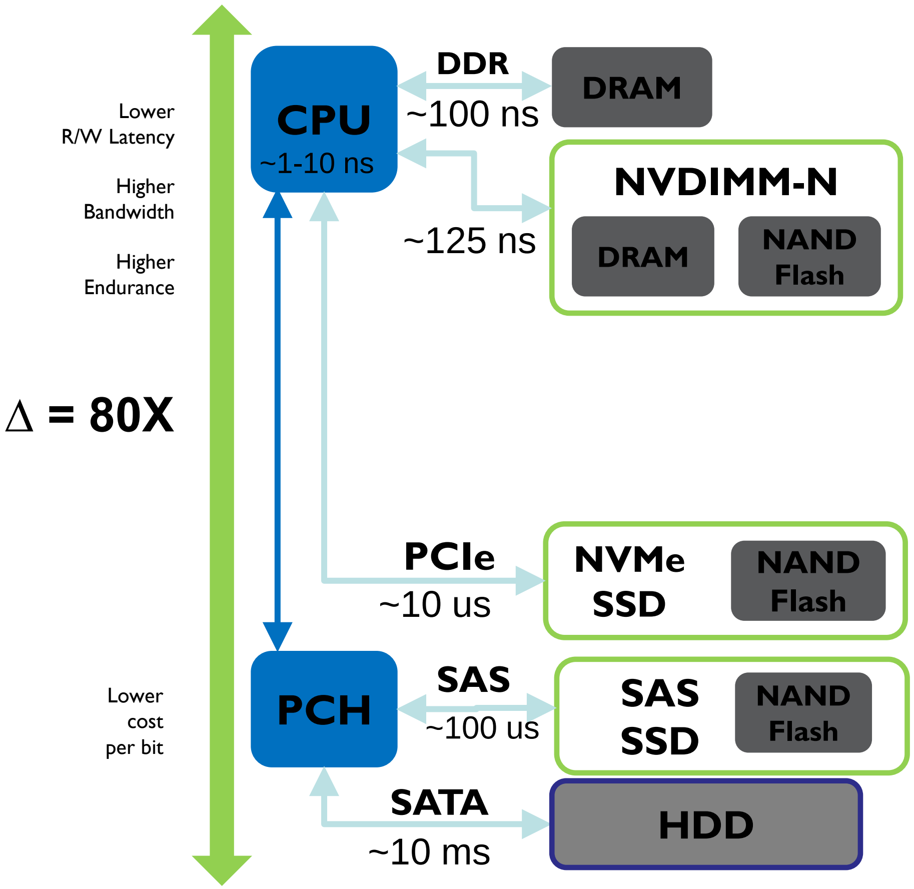
PCH

ATA

~10 ms

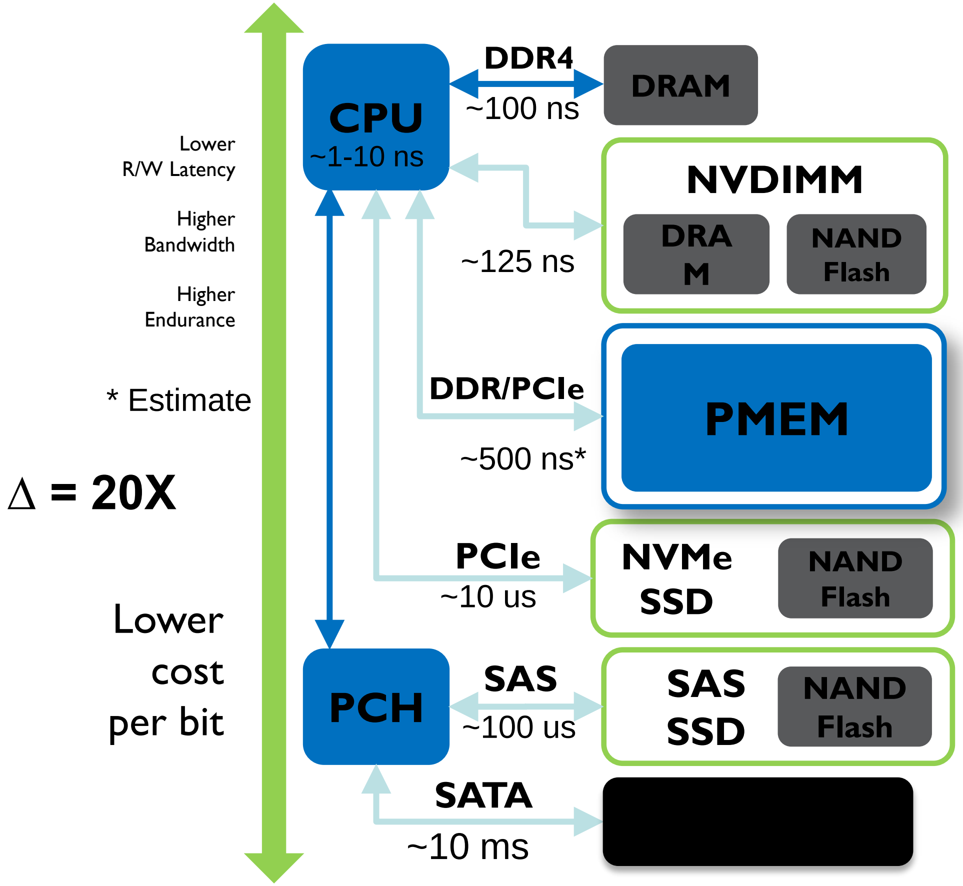
- For decades we've had two primary types of memories in computers: DRAM and Hard Disk Drive (HDD)
- DRAM was fast and volatile and HDDs were slower, but nonvolatile (aka persistent)
- Data moves from the HDD to DRAM over a bus where it is the fed to the processor
- The processor writes the result in DRAM and then it is stored back to disk to remain for future use
- ATA HDD is 100,000 times slower than DRAM (!!!)

The Present: 2D Hybrid Memory Server Architectures



- System performance increased as the speed of both the interface and the memory accesses improved
- NAND Flash considerably improved the nonvolatile response time
- SAS and PCIe made further optimizations to the storage interface
- NVDIMM provides battery- or ultra-capacitor-backed DRAM, operating at near-DRAM speeds and retains data when power is removed
- NVMe transport provides efficient use of PCI-Express bus (queues, etc.)

The Future: 3D Nonvolatile Memories in Server Architectures



- NVM technology provides the benefit in 'the middle' – reduces the gap
- Significantly faster than NAND Flash with much higher endurance
- Performance can be realized on PCIe or DDR buses – storage or memory
- Lower cost per bit than DRAM while being considerably more dense
    - Software-enabled via PMEM & others
- **Main Attributes:**
    - Byte addressable on the PCIe / coherent BUS domain
    - Latency at the scale of BUG latency

Local interfaces to Persistent Memory

- JDEC
  - NVDIMM-N/F
  - NVDIMM-P
    - Credit based
    - Non posted
    - Undeterministic latency
- NVMe
  - CMB (Persistent?)
- PCIe
  - Standardized commitment of data to persistency
- Others…

Software APIs for Persistent Memory

- NVM Programming Model
  - Describes application visible behaviors
    - Guarantees, Error semantics
  - Allows API's to align with OS's
  - Exposes opportunities in networks and processors
  - Accelerate the availability of software that enables NVM (Non-Volatile Memory) hardware.
- Implementations
  - Linux DAX extensions
  - PMEM.IO userspace library
    - Bypass the OS for persistency guarantees

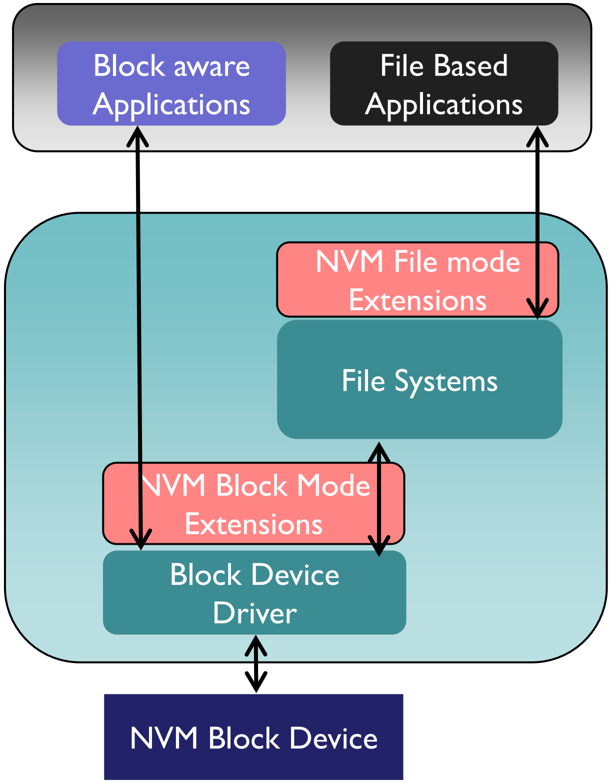
The NVM Programming Model

Programming Model Modes

- Block and File modes use IO
  - Data is read or written using RAM buffers
  - Software controls how to wait (context switch or poll)
  - Status is explicitly checked by software
- Volume and PM modes enable Ld/St
  - Data is loaded into or stored from processor registers
  - Processor makes software wait for data during instruction
  - No status checking – errors generate exceptions

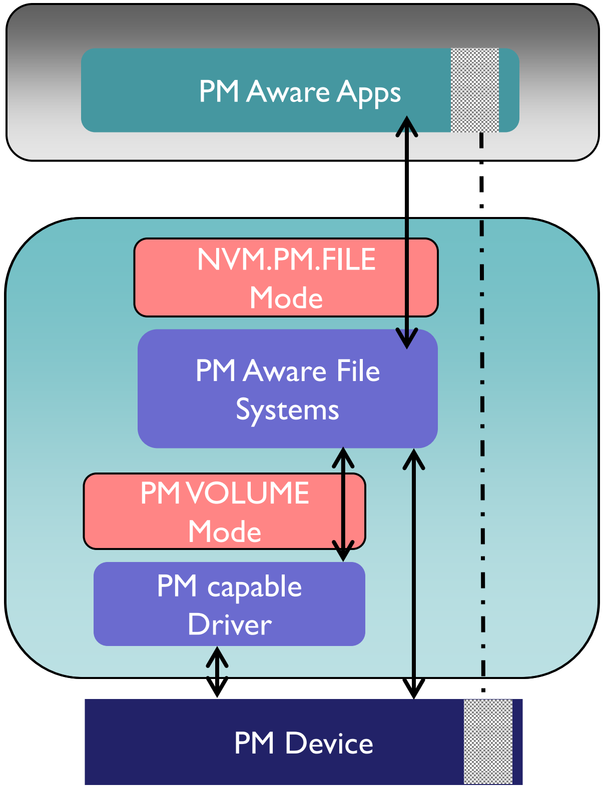
File and Block Mode Extensions

- NVM.BLOCK Mode
  - Targeted for file systems and block-aware applications
  - Atomic writes
  - Length and alignment granularities
  - Thin provisioning management
- NVM.FILE Mode
  - Targeted for file based apps.
  - Discovery and use of atomic write features
  - Discovery of granularities



Persistent Memory (PM) Modes

- NVM.PM.VOLUME
  - Software abstraction for persistent memory hardware
  - Address ranges
  - Thin provisioning management
- NVM.PM.FILE
  - Application behavior for accessing PM
  - Mapping PM files to application address space
  - Syncing PM files

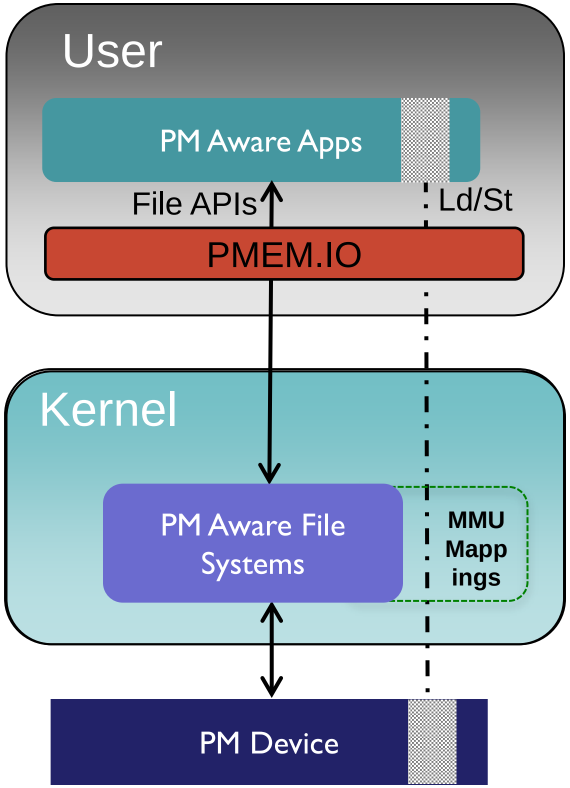


NVM.PM.FILE Basic Semantics

- MAP
  - Associates memory address with a file descriptor and offset
  - Specific address may be requested
- SYNC
  - Commit the read/write transactions to persistency
  - Types
    - Optimize flush
    - Optimize flush and verify

NVM PM TWG Latest Work

- Atomicity White Paper
  Transactional PM Libraries
- Remote Access for HA White Paper
  - High Availability PM - Remote Optimized Flush
- Asynchronous Optimize Flush



Remote Direct Memory Access (RDMA)

RDMA – What?

- Remote
  - data transfers between nodes in a network
- Direct
  - no Operating System Kernel involvement in transfers
  - everything about a transfer offloaded onto Interface Card
- Memory
  - transfers between user space application virtual memory
  - no extra copying or buffering
- Access
  - send, receive, read, write, atomic operations

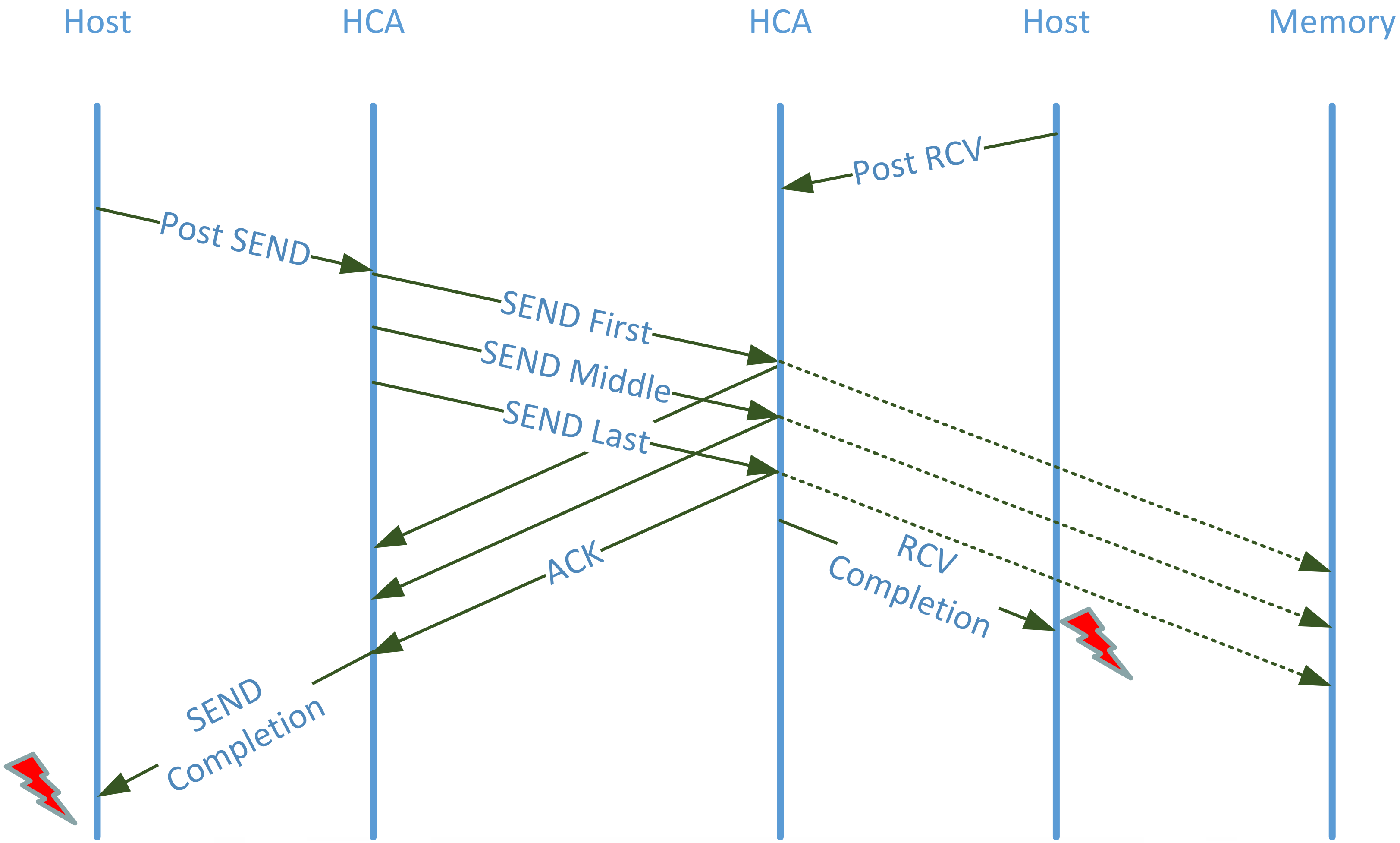
RDMA - Why?

- ❐ Latency (<usec)
- ❐ Zero-copy
- ❐ Hardware based one sided memory to remote memory operations
- ❐ OS and network stack bypasses
- ❐ Reliable credit base data and control delivery by hardware
- ❐ Network resiliency
- ❐ Scale out with standard converged network (Ethernet/InfiniBand)

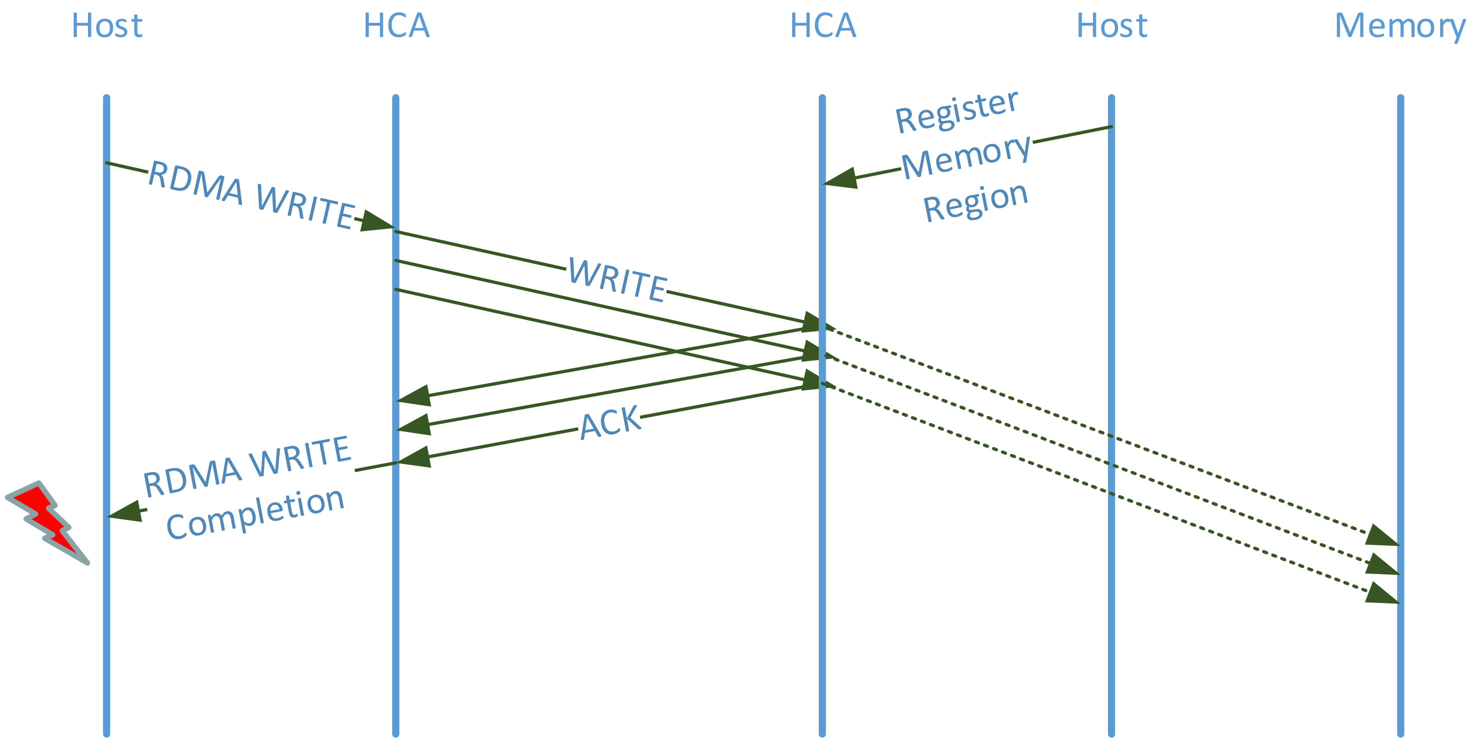
RDMA – How?

- Transport built on simple primitives deployed for 15 years in the industry
  - **Queue Pair (QP)** – RDMA communication end point
  - **Connect** for establishing connection mutually
  - RDMA **Registration** of memory region (REG_MR) for enabling virtual network access to memory
  - **SEND** and **RCV** for reliable two-sided messaging
  - RDMA **READ** and RDMA **WRITE** for reliable one-sided memory to memory transmission

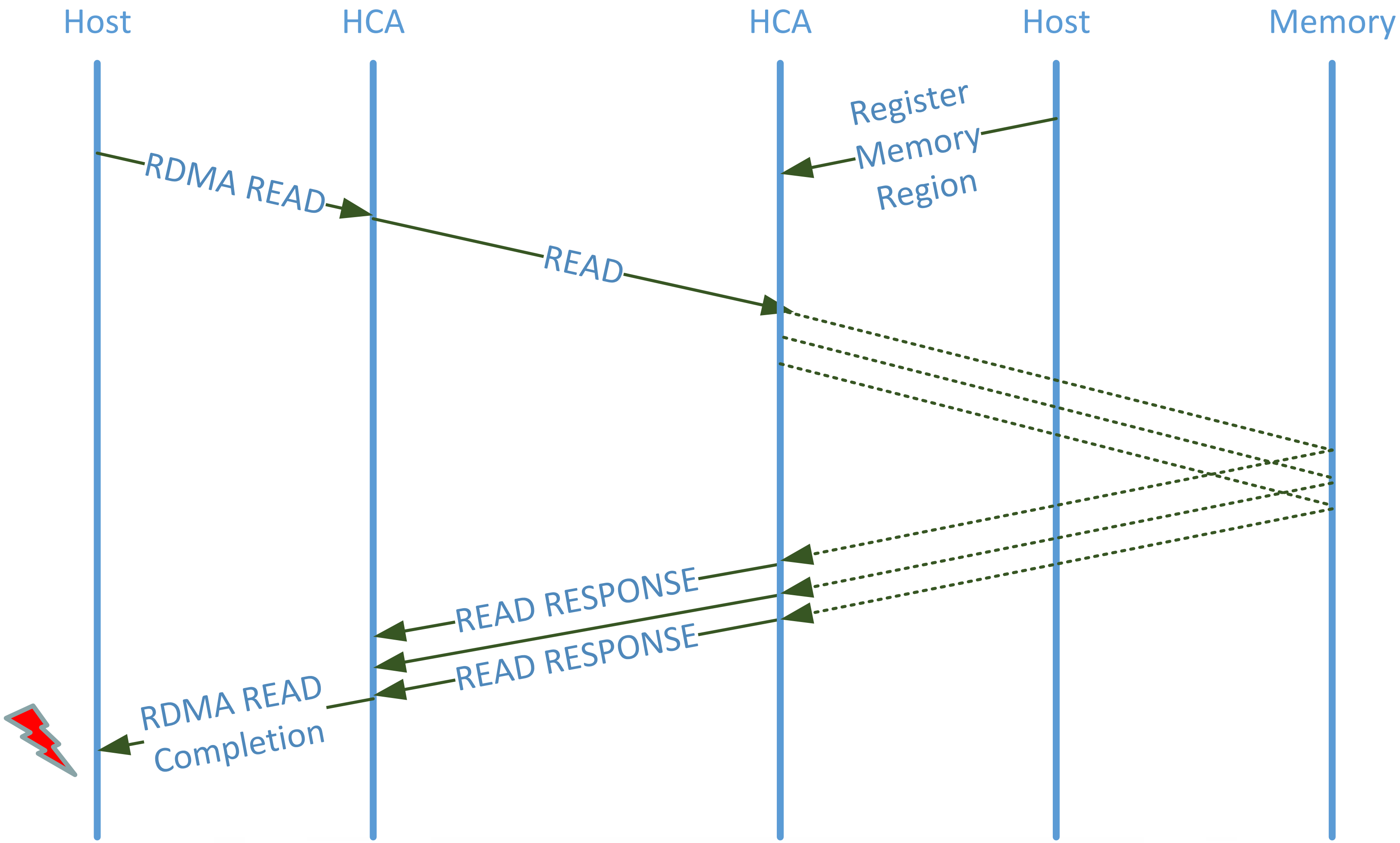
SEND/RCV


Host
HCA
HCA
Host
Memory
Post RCV
Post SEND
SEND First
SEND Middle
SEND Last
ACK
RCV Completion
SEND Completion

RDMA WRITE


Host
HCA
HCA
Host
Memory
Register Memory Region
RDMA WRITE
WRITE
ACK
RDMA WRITE Completion

RDMA READ


Host
HCA
HCA
Host
Memory
Register Memory Region
RDMA READ
READ
READ RESPONSE
READ RESPONSE
RDMA READ Completion

Ordering Rules

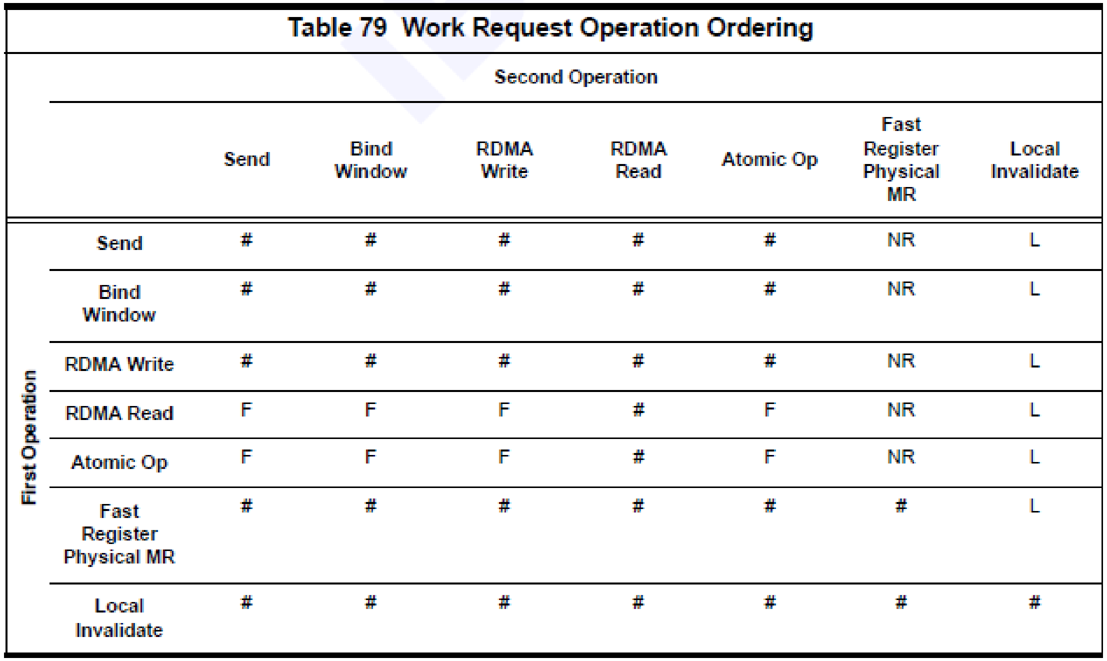


Table 79 Work Request Operation Ordering

		Second Operation						
		Send	Bind Window	RDMA Write	RDMA Read	Atomic Op	Fast Register Physical MR	Local Invalidate
First Operation	Send	#	#	#	#	#	NR	L
	Bind Window	#	#	#	#	#	NR	L
	RDMA Write	#	#	#	#	#	NR	L
	RDMA Read	F	F	F	#	F	NR	L
	Atomic Op	F	F	F	#	F	NR	L
	Fast Register Physical MR	#	#	#	#	#	#	L
	Local Invalidate	#	#	#	#	#	#	#

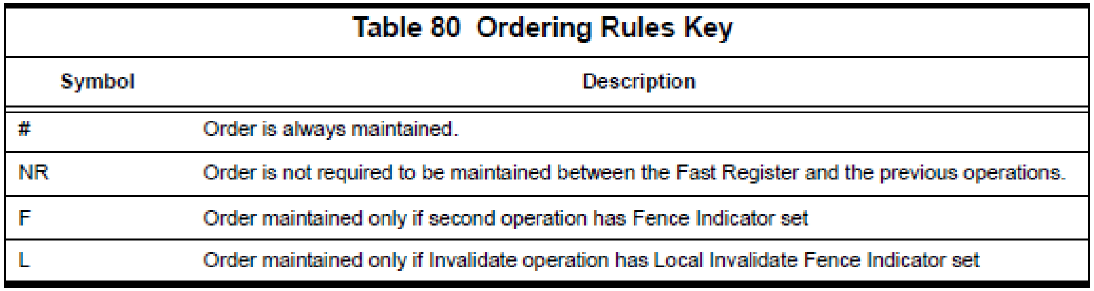
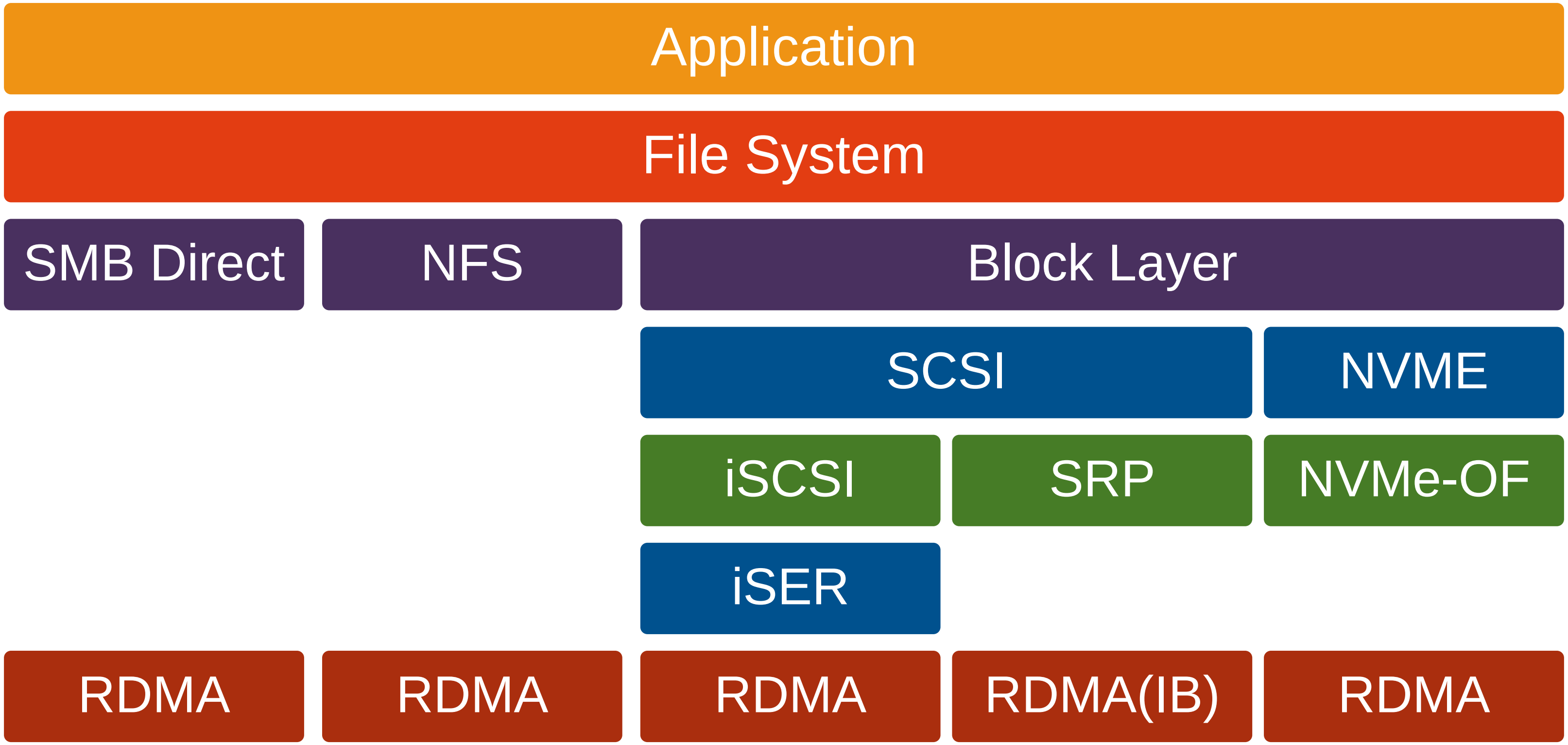


Table 80 Ordering Rules Key

Symbol	Description
#	Order is always maintained.
NR	Order is not required to be maintained between the Fast Register and the previous operations.
F	Order maintained only if second operation has Fence Indicator set
L	Order maintained only if Invalidate operation has Local Invalidate Fence Indicator set

RDMA Storage Protocols

Variety of RDMA Storage Protocols



Application

File System

SMB Direct

NFS

Block Layer

SCSI

NVME

iSCSI

SRP

NVMe-OF

iSER

RDMA

RDMA

RDMA

RDMA(IB)

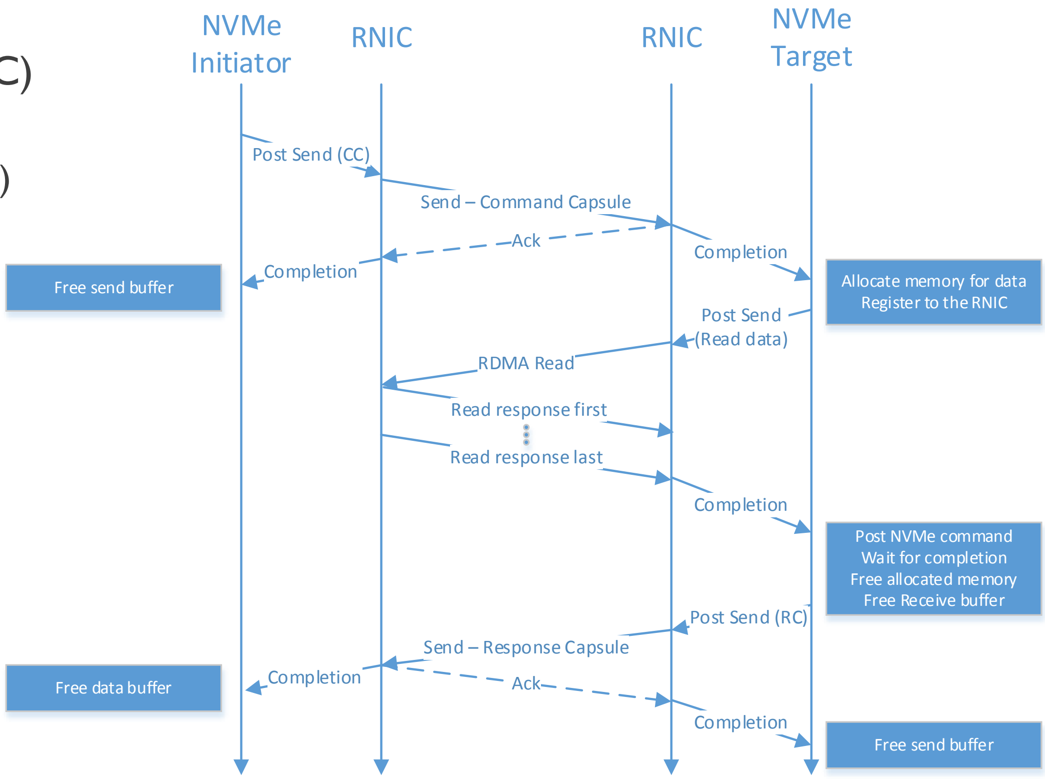
RDMA

Similar Exchange Model

Example: NVMe over Fabrics

NVMe-OF IO WRITE (PULL)

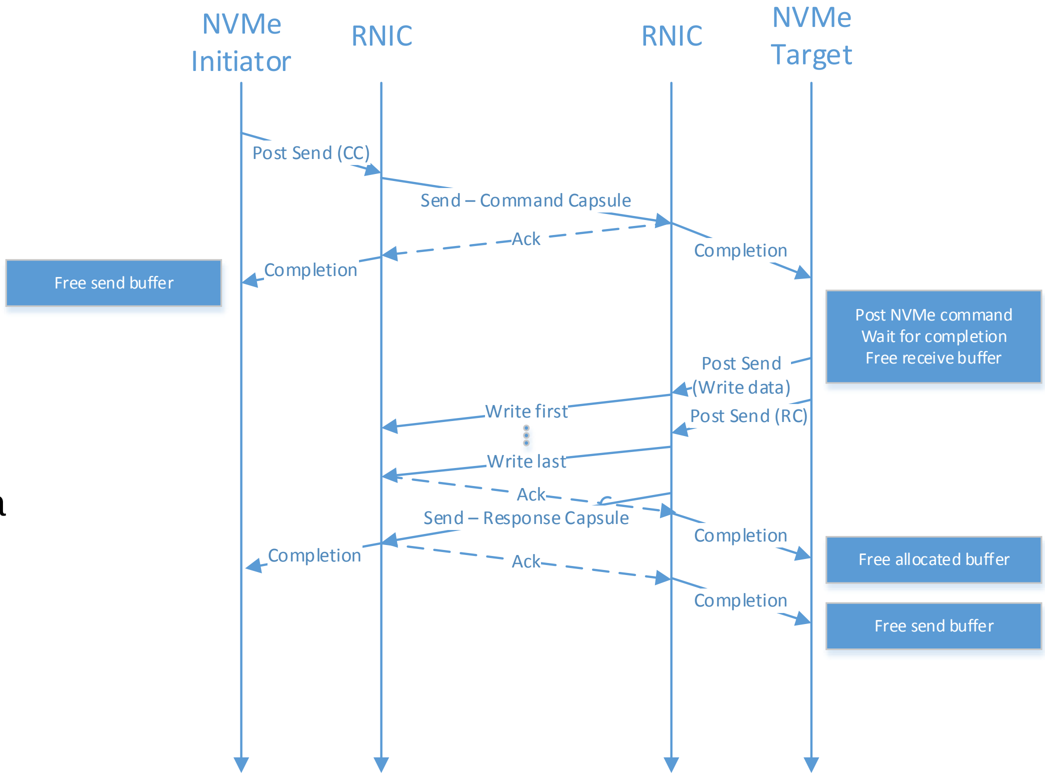
- Host
  - SEND carrying Command Capsule (CC)
  - Upon RCV completion
    - Free the data buffer (invalidate if needed)
- Subsystem
  - Upon RCV Completion
    - Allocate Memory for Data
    - Post RDMA READ to fetch data
  - Upon READ Completion
    - Post command to backing store
  - Upon SSD completion
    - Send NVMe-OF RC
    - Free memory
  - Upon SEND Completion
    - Free CC and completion resources



Additional Round Trip to Fetch Data

NVMe-OF IO READ

- Host
  - Post SEND carrying Command Capsule (CC)
- Subsystem
  - Upon RCV Completion
    - Allocate Memory for Data
    - Post command to backing store
  - Upon SSD completion
    - Post RDMA Write to write data back to host
    - Send NVMe RC
  - Upon SEND Completion
    - Free memory
    - Free CC and completion resources



NVMe Initiator

RNIC

RNIC

NVMe Target

Post Send (CC)

Send – Command Capsule

Ack

Completion

Completion

Free send buffer

Post NVMe command
Wait for completion
Free receive buffer

Post Send (Write data)

Post Send (RC)

Write first

Write last

Ack

Send – Response Capsule

Completion

Completion

Free allocated buffer

Ack

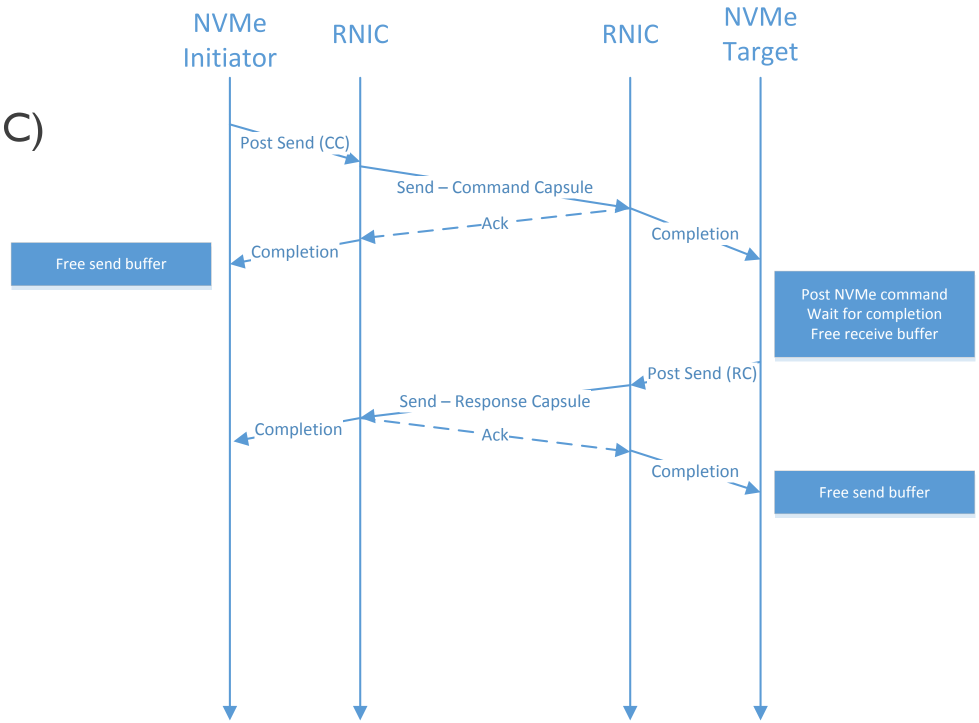
Completion

Free send buffer

Interrupts Required for Reading

NVMe-OF IO WRITE In-Capsule (PUSH)

- Host
  - Post SEND carrying Command Capsule (CC)
  - Upon RCV completion
    - Free the data buffer (invalidate if needed)
- Subsystem
  - Upon RCV Completion
    - Allocate Memory for Data
  - Upon SSD completion
    - Send NVMe-OF RC
    - Free memory
  - Upon SEND Completion
    - Free CC and completion resources



Large Buffers Should be Posted in Advance

Data is being places in undeterministic location with respect to the requester

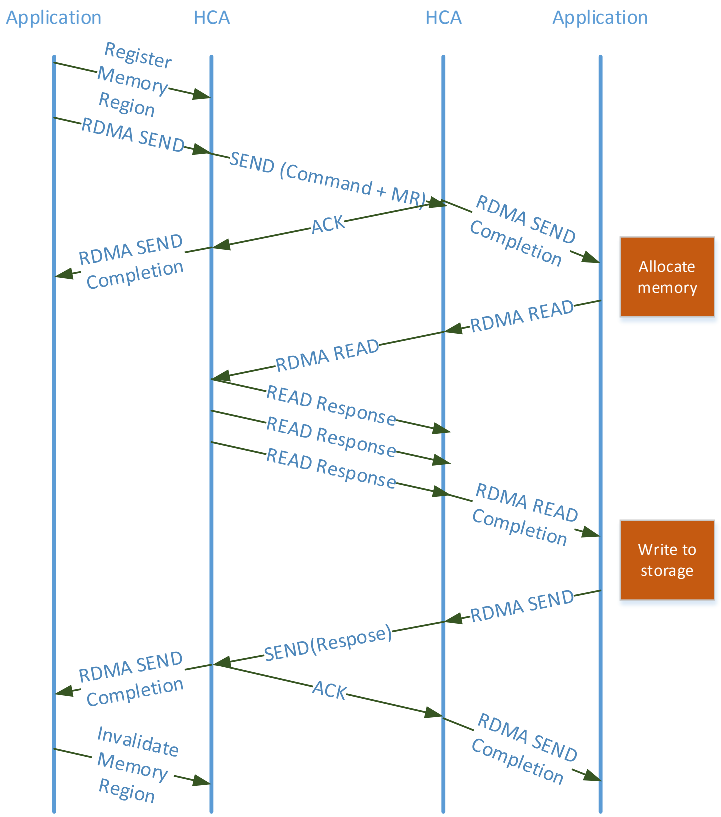
Summary

- Storage over RDMA Model Overview
  - Commands and Status
    - SEND
  - Data
    - RDMA initiated by Storage Target
- Protocol Design Considerations
  - Latency of storage device is orders of magnitude higher than network latency
  - Storage is usually not RDMA addressable
  - Scalability of intermediate memory

Latency Breakdown for Accessing Storage

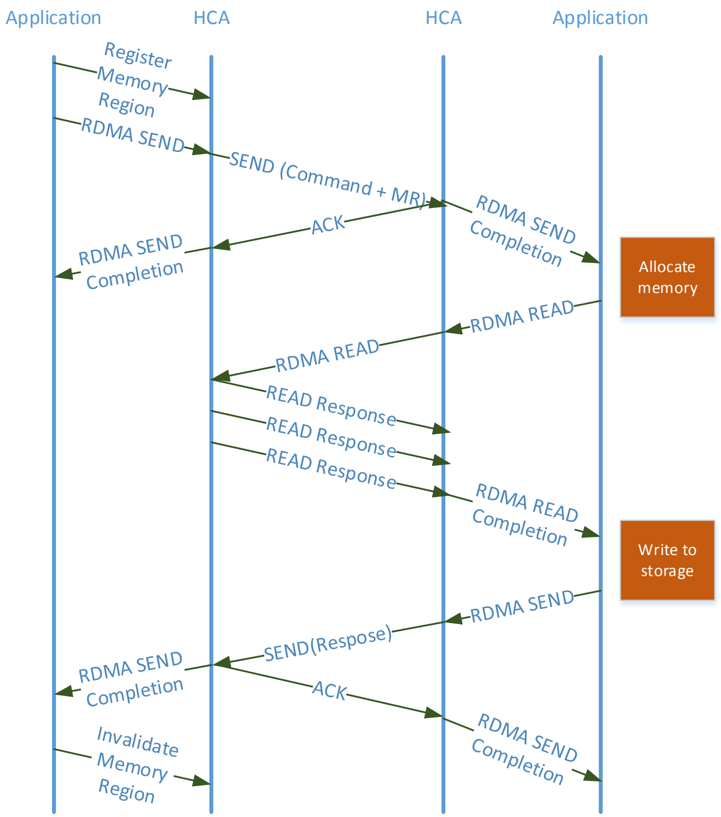
- Traditional storage latency are ~10usec added latency
  - Round trip time (RTT)
  - Interrupt overheads
  - Memory management and processing
  - Operating system call

- RDMA latency may go down to 0.7 usec



Latency Breakdown for Accessing Storage

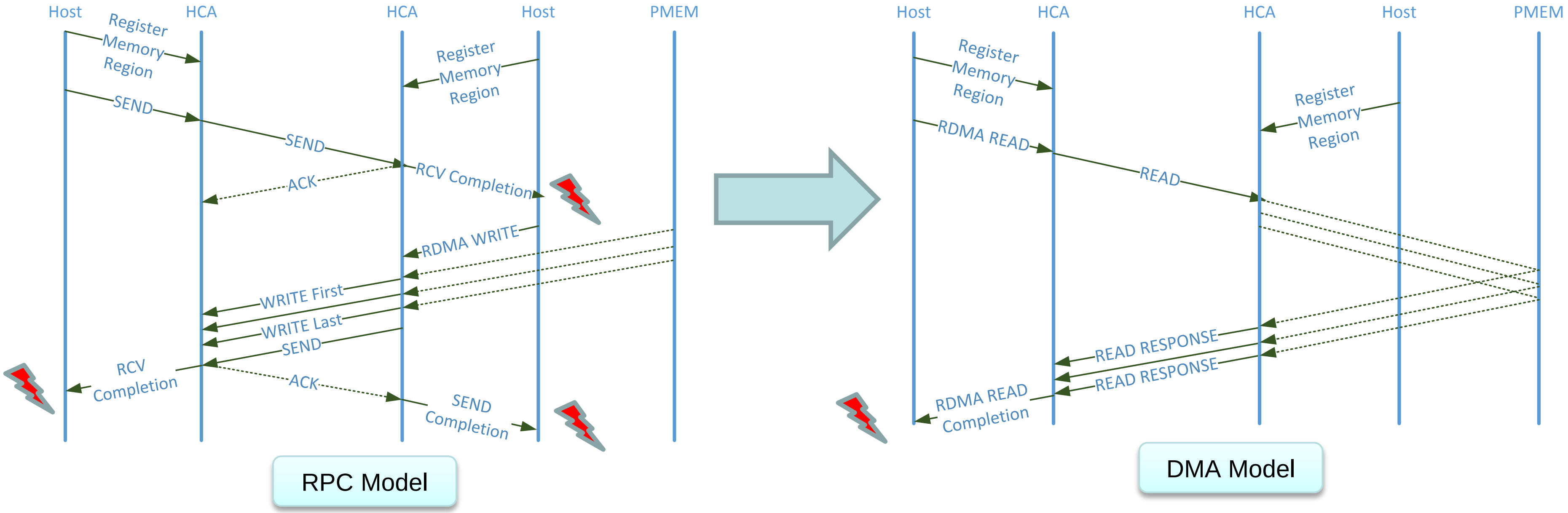
- Persistent Memory Non-Volatile memory technologies are reducing the latency to <1usec
- Spending time on RTT and interrupts becomes unacceptable and unnessacary
  - Memory mapped for DMA is not a staging buffer, it is the storage media
- PMEM storage should be exposed directly to RDMA access to meet its performance goals



Remote Access to Persistent Memory Exchange Model

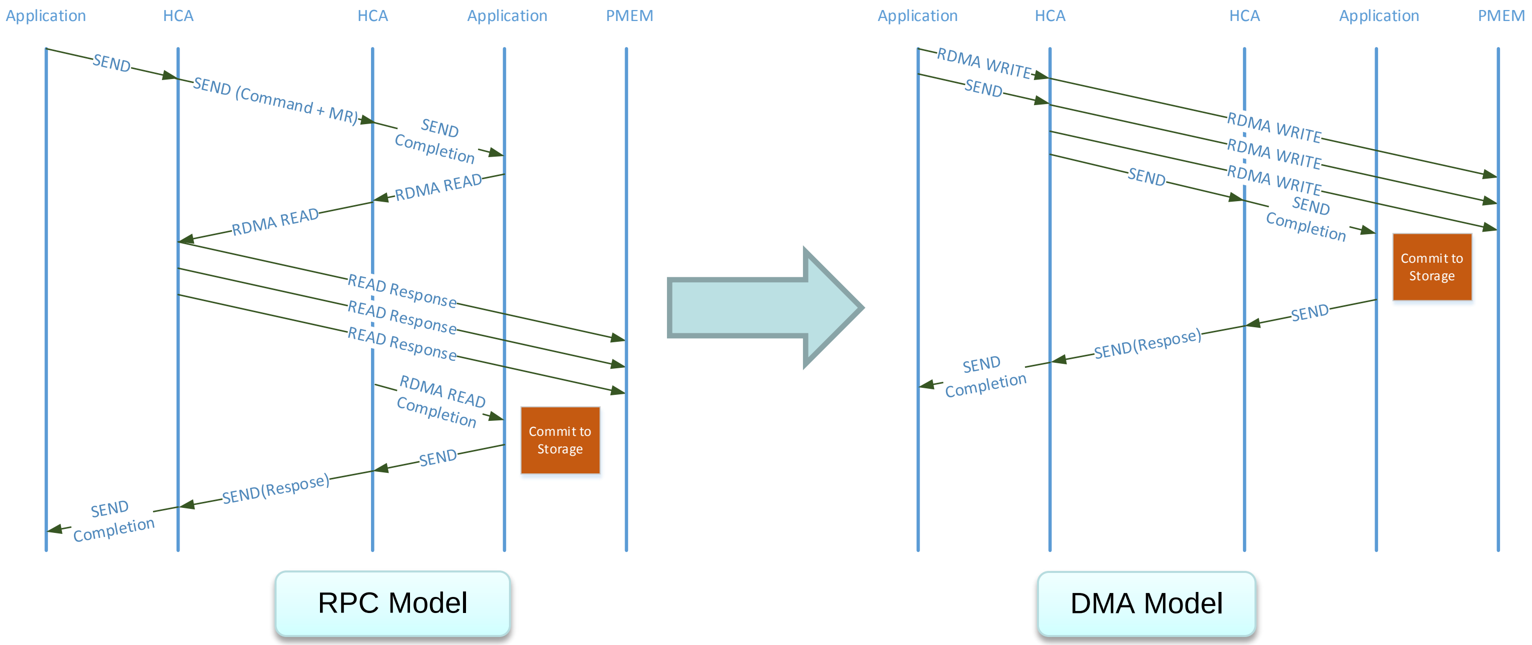
Proposal – RDMA READ Directly From PMEM

- Utilized the RDMA capabilities to directly with storage



Proposal – RDMA Write Directly to PMEM

- Utilized the RDMA capabilities to directly communicate with storage

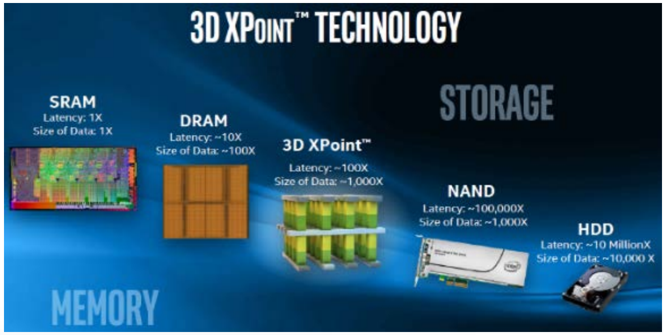


Resolves the latency, memory scalability and processing overhead while communicating with PMEM

DMA Model is Mostly Relevant When…

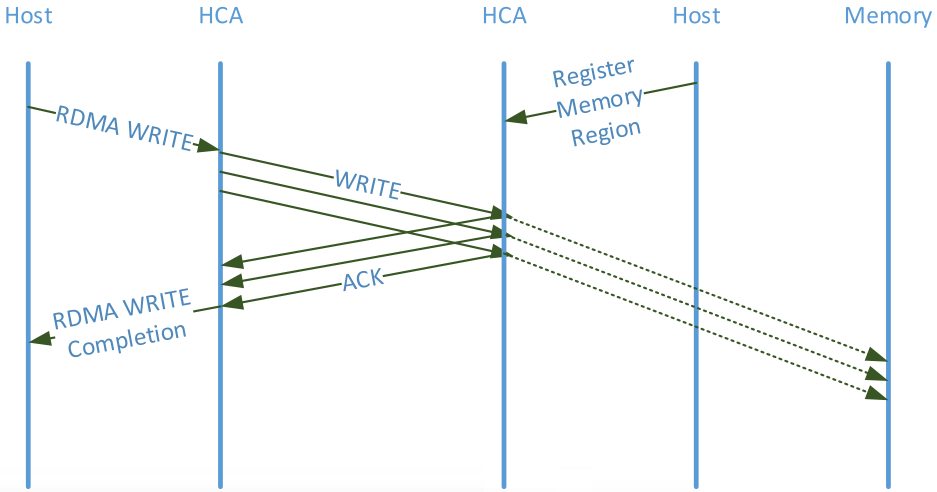
- Storage Access Time becomes comparable to Network Roundtrip Time
- Storage Device can be Memory Mapped Directly as RDMA WRITE Target Buffer

- Main challenge:
  - RDMA WRITE/Atomics reliability guarantees



RDMA WRITE Semantics

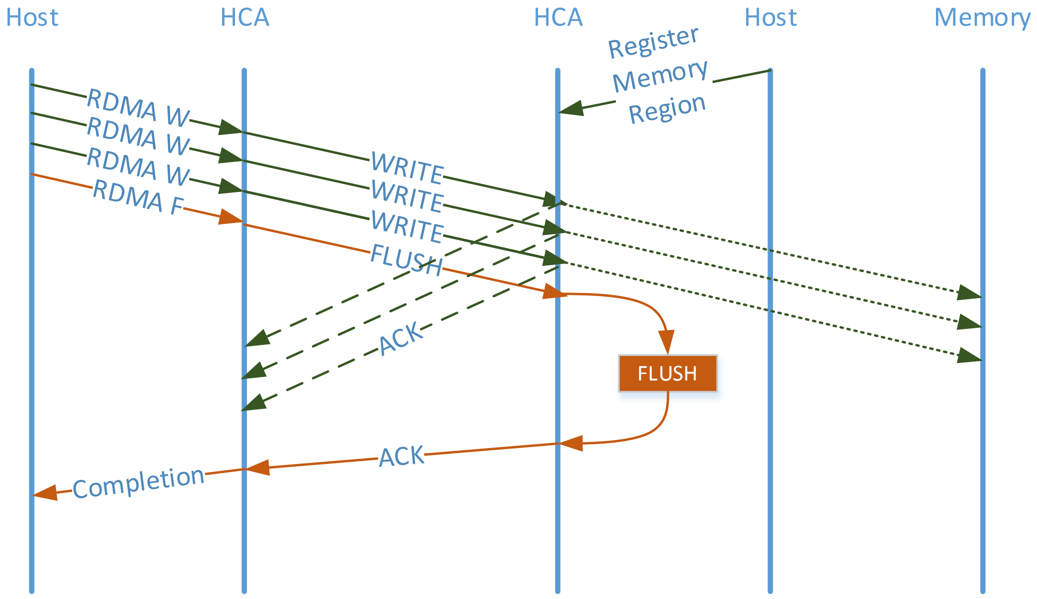
- RDMA Acknowledge (and Completion)
  - Guarantee that Data has been successfully received and accepted for execution by the remote HCA
  - Doesn’t guarantee data has reached remote host memory
- Further Guarantees implemented by ULP therefore requires interrupt and add latency to the transaction
- Reliability extensions are required



RDMA Extensions for Persistency and Consistency

RDMA Extensions for Memory Persistency and Consistency

- RDMA FLUSH Extension
  - New RDMA operation
  - Higher Performance and Standard
  - Straightforward Evolutionary fit to RDMA Transport
- Target guarantees
  - Consistency
  - Persistency



RDMA FLUSH Operation System Implication

- System level implication may be:
  - Caching efficiency
  - Persistent memory bandwidth / durability
  - Performance implications for the flush operation

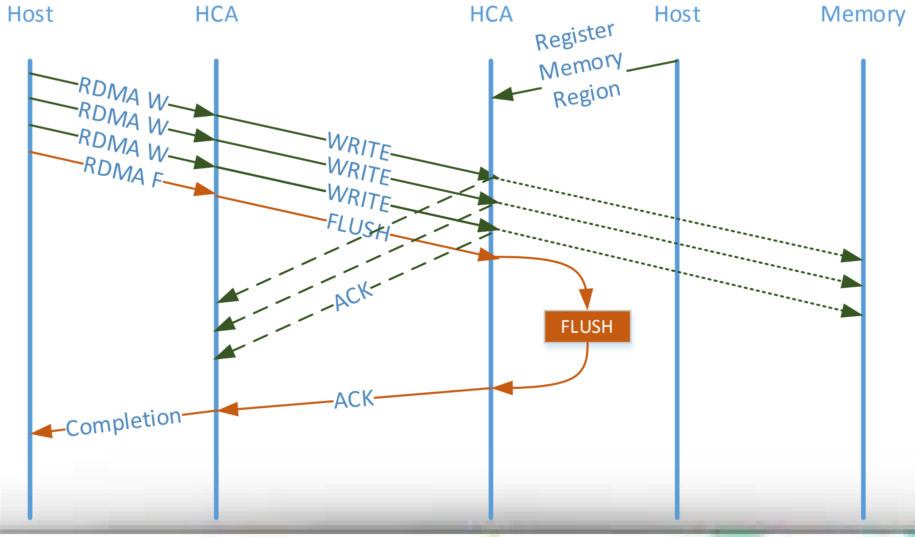
- The new reliability semantics design should consider these implications during the design of the protocol

- These implications are the base for our requirement

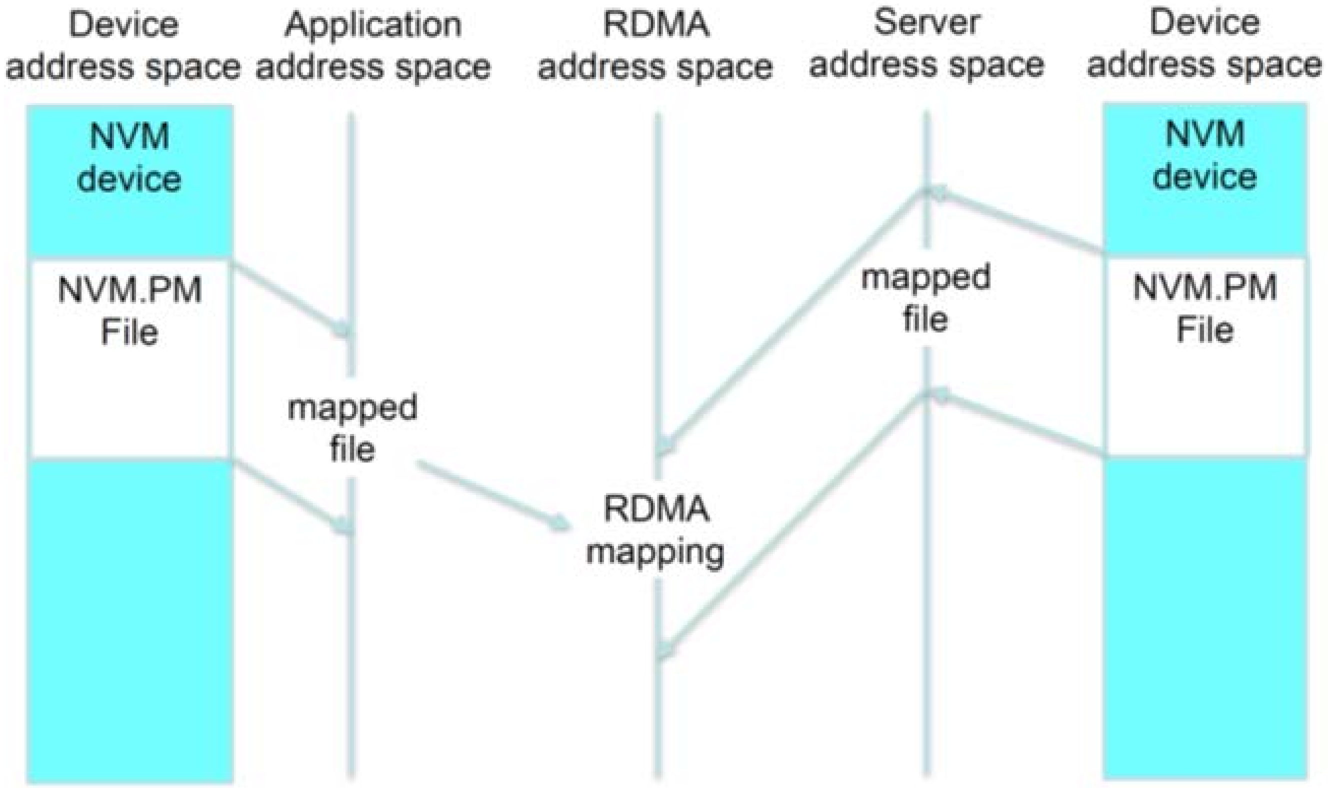
Baseline Requirement

- Enable FLUSH the data selectively to persistency
- FLUSH the data only once consistently
- Responder should be capable of provisioning which memory regions could execute FLUSH and are targeted to PMEM
- Enable amortization of the FLUSH operation over multiple WRITE
  - Utilize RDMA ordering rules for ordering guarantees

Figure: Flush is a new transport operation



Use Case: RDMA to PMEM for High Availability



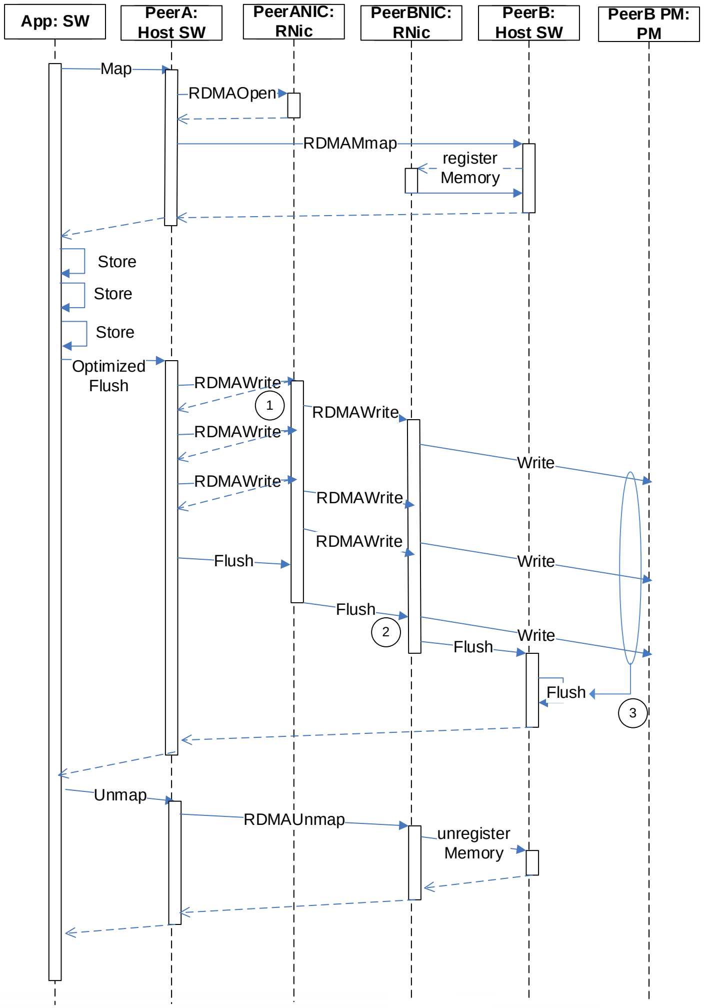
Mapping controlled by RNIC

Mapping controlled by client peer OS

Mapping controlled by server peer OS

Use Case: RDMA to PMEM for High Availability

- MAP
  - Memory address for the file
  - Memory address + Registration of the replication
- SYNC
  - Write all the “dirty” pages to remote replication
  - FLUSH the writes to persistency
- UNMAP
  - Invalidate the registered pages for replication

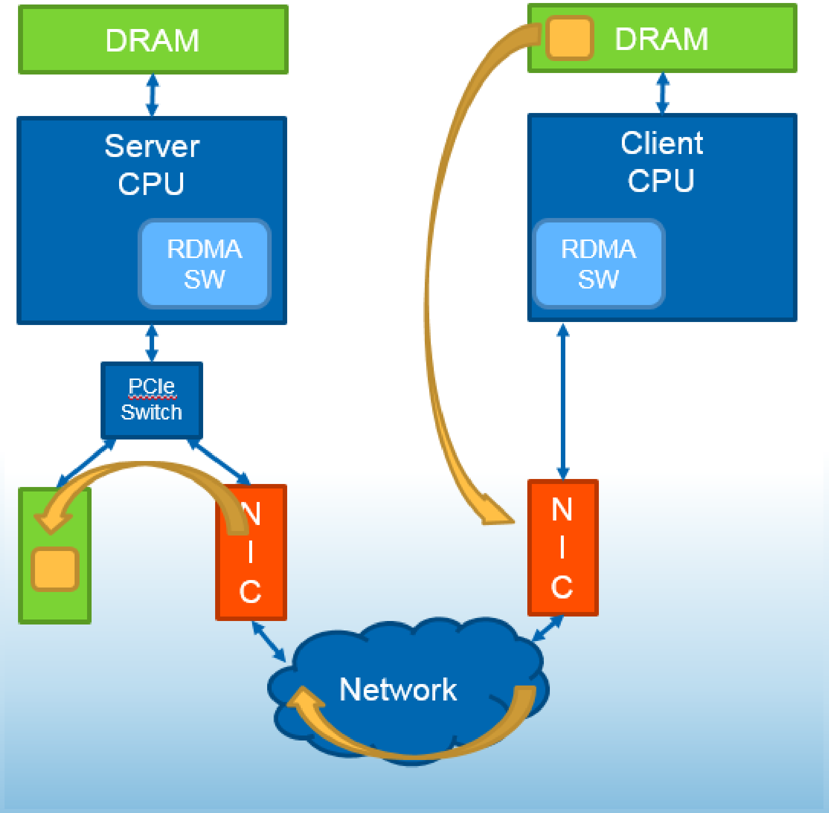


Proof of Concepts

Peer-Direct NVRAM over RDMA Fabrics Proof of Concept

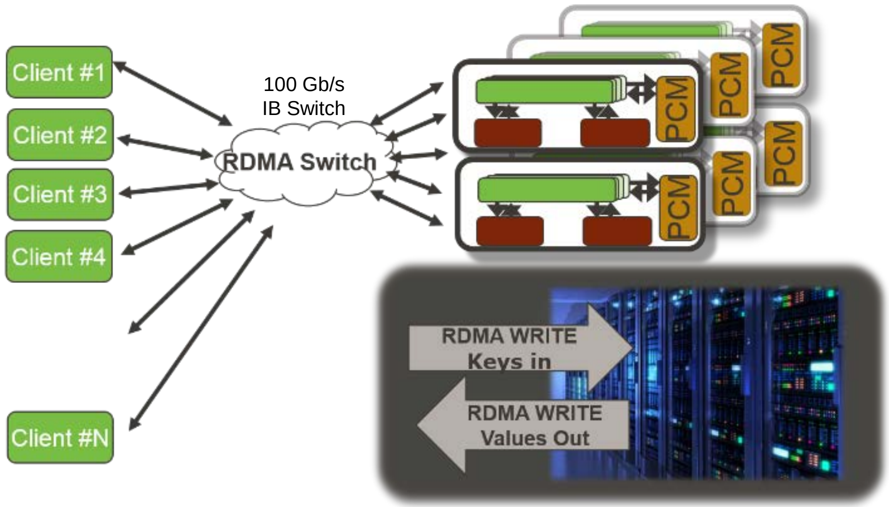
- Development platform to enable testing of remote memory transactions over RDMA fabrics to non-volatile storage
  - Mellanox RDMA HCA
  - Microsemi NVRAM Card
  - Microsemi PCIe Switch
- IO transactions bypass host CPU on server using Peer-Direct
  - Reduced server load and DRAM bandwidth
- 4us latency for 4KB IO from client to server non-volatile memory over RDMA connection
  - Network latency no longer a don't-care for remote block IO transactions

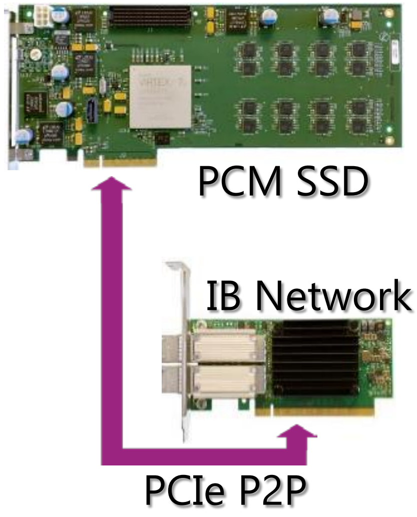
Flash Memory Summit 2015



HGST FMS Demo

Key-Value Store fetch from Non-Volatile Storage in ~ 2 µs, comparable to cutting-edge DRAM systems





Conclusions

- Persistent memory performance and functionality brings a paradigm shift to storage technology
- Effective remote access to such media will require revision of the current RDMA storage protocols
- IBTA extends the transport capabilities of RDMA to support placement guarantees to align these into a single efficient standard
  - Join us to IBTA if you want to contribute

Thanks!

idanb@mellanox.com