



SDC 18

September 24-27, 2018
Santa Clara, CA

www.storagedeveloper.org

Remote Persistent Memory

SNIA Nonvolatile Memory Programming TWG

Tom Talpey
Microsoft

Outline

- ❑ SNIA NVMP TWG activities
- ❑ Remote Access for High Availability work
- ❑ RDMA requirements and extensions
- ❑ Current and future TWG remote access focus



NVM Programming Model TWG - Mission

- ❑ Accelerate the availability of software that enables Persistent Memory (PM) hardware.
 - ❑ Hardware includes SSD's and PM
 - ❑ Software spans applications and OS's
- ❑ Create the NVM Programming Model
 - ❑ Describes application-visible behaviors
 - ❑ Allows API's to align with OS's
 - ❑ Describes opportunities in networks and processors

SNIA NVM Programming Model

- ❑ [Version 1.2](#) approved by SNIA in June 2017
- ❑ Expose new block and file features to applications
 - ❑ Atomicity capability and granularity
 - ❑ Thin provisioning management
- ❑ Use of memory mapped files for persistent memory
 - ❑ Existing abstraction that can act as a bridge
 - ❑ Limits the scope of application re-invention
 - ❑ Open source implementations available
- ❑ Programming Model, not API
 - ❑ Described in terms of attributes, actions and use cases
 - ❑ Implementations map actions and attributes to API's

NVM Programming Model modes

	Block Mode Innovation	Emerging PM Technologies
User View	NVM.FILE 	NVM.PM.FILE 
Kernel Protected	NVM.BLOCK	NVM.PM.VOLUME
Media Type	Disk Drive	Persistent Memory
NVDIMM	Disk-Like	Memory-Like

- ❑ 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 Load/Store/Move
 - ❑ Data is loaded into or stored from processor registers
 - ❑ Processor waits for data during instruction
 - ❑ No status returned – errors generate exceptions

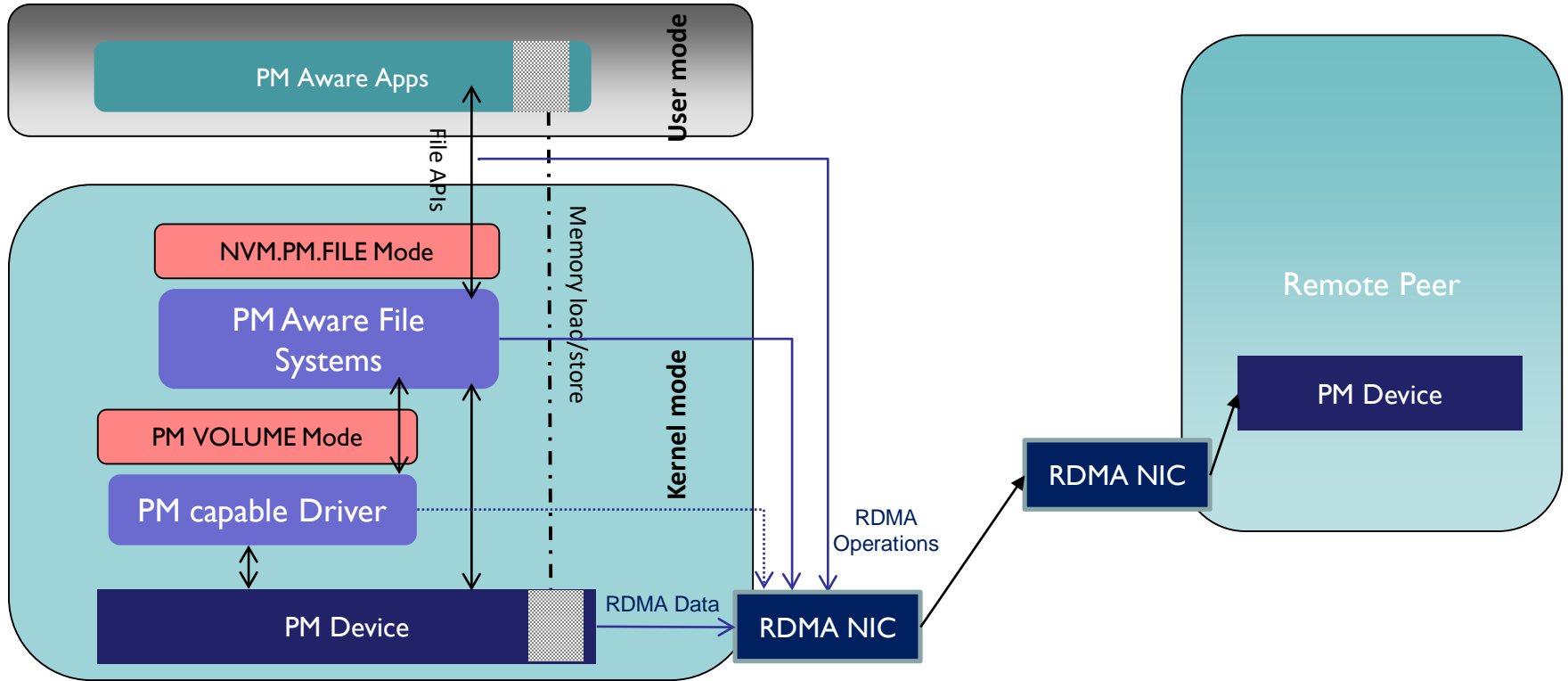
The current version (1.2) of the specification is available at

https://www.snia.org/sites/default/files/technical_work/final/NVMProgrammingModel_v1.2.pdf

Remote Access for HA

- ❑ History
 - ❑ Remote Access for High Availability white paper published 2016
 - ❑ NVM Programming Model 1.2 published June 2017
 - ❑ Major new installment on error handling
 - ❑ Optimized Flush Allowed
 - ❑ Deep Flush
- ❑ NVM Programming Model Specification 1.3 in development
 - ❑ Update specification to reflect learning from implementations
 - ❑ Incorporate learning from remote access white paper
 - ❑ Asynchronous Flush
 - ❑ Remote persistence ordering, error handling
- ❑ Remote Access Collaboration with Open Fabrics Alliance OFI WG
 - ❑ PM Remote Access for HA V1.1 in development
 - ❑ Expand remote access use case enumeration

Persistent Memory (PM) Modes, +Remote



PM Remote Access for HA

- ❑ NVMP TWG-developed interface for remote Pmem
- ❑ Maximize alignment with local PMEM interface
- ❑ Take remote environment into account
 - ❑ Including RDMA semantics and restrictions
- ❑ Analyze the error cases
 - ❑ As always, “the hard part”

Outline of “NVM PM Remote Access for HA”

- ❑ 1 Purpose
- ❑ 2 Scope
- ❑ 3 Memory Access Hardware Taxonomy
- ❑ 4 Recoverability Definitions
 - ❑ Durability vs Availability
 - ❑ Consistency
 - ❑ Recovery
 - ❑ Integrity
- ❑ 5 HA Extensions to NVM.PM.FILE
- ❑ 6 RDMA for HA
- ❑ 7 RDMA Security
- ❑ 8 Requirements Summary
- ❑ Appendices
 - ❑ Workload generation and measurement
 - ❑ HA protocol flow alternatives
 - ❑ Remote atomicity considerations

Map and Sync

- ❑ Map (local PMEM)
 - ❑ Associates memory addresses with open file
 - ❑ Caller may request specific address
- ❑ Sync (local PMEM)
 - ❑ Flush CPU cache for indicated range
 - ❑ Additional Sync types
 - ❑ Optimized Flush – multiple ranges from user space
 - ❑ Optimized Flush and Verify – Optimized flush with verification from media
- ❑ Warning! Sync does not guarantee order
 - ❑ Parts of CPU cache may be flushed out of order
 - ❑ This may occur before the sync action is taken by the application
 - ❑ Sync only guarantees that all data in the indicated range has been flushed some time before the sync completes
- ❑ All the above are true remotely, but
 - ❑ Remote addresses are not “mapped”
 - ❑ Stores do not magically become RDMA Writes
 - ❑ Flushing applies to RDMA, network and i/o pipeline (not simply CPU)
 - ❑ An asynchronous flush is needed for efficiency!

Failure Atomicity

- ❑ Current processor + memory systems
 - ❑ Guarantee inter-process consistency (SMP)
 - ❑ But only provide limited atomicity with respect to failure
 - ❑ System reset/restart/crash
 - ❑ Power Failure
 - ❑ Memory Failure
- ❑ Failure atomicity is processor architecture specific
 - ❑ Processors provide failure atomicity of aligned fundamental data types
 - ❑ Fundamental data types include pointers and integers
 - ❑ PM programs use these to create larger atomic updates or transactions
 - ❑ Fallback is an additional checksum or CRC
- ❑ Failure atomicity is further impacted by network
 - ❑ Alignment restrictions are potentially different for bus-attached devices
 - ❑ Network and PCI Express packetization impact ordering and write boundaries
 - ❑ Network failures reflected in new scenarios

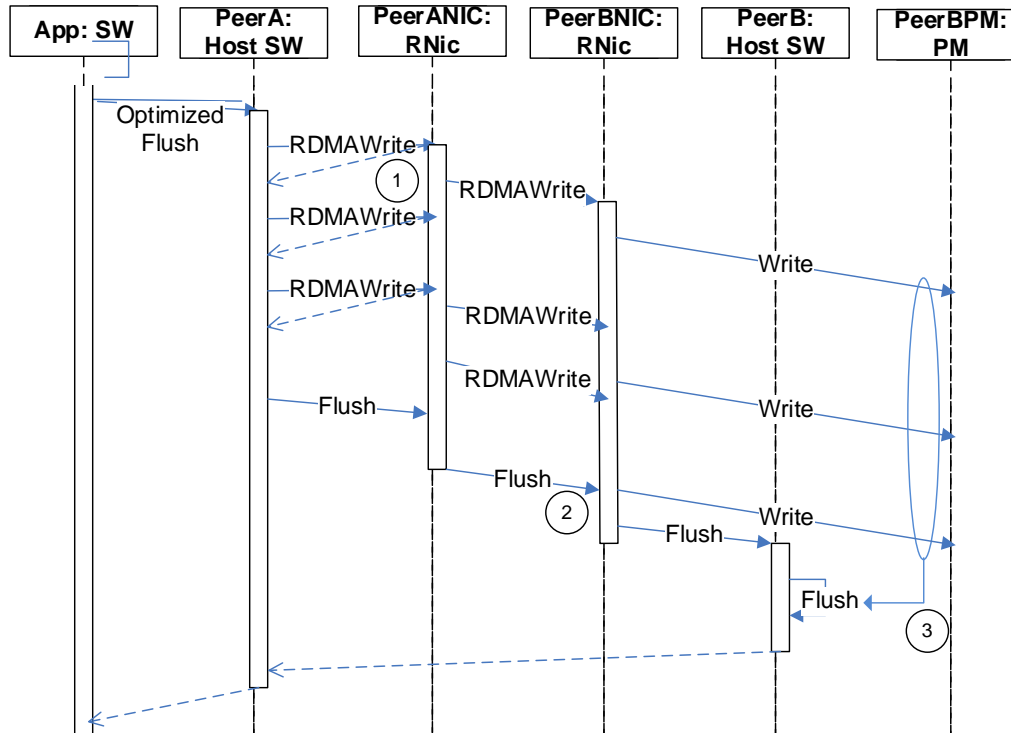
Consistency for recoverability

- ❑ Application level goal is recovery from failure
 - ❑ Requires robust local and remote error handling
 - ❑ High Availability (as opposed to High Durability) requires application involvement.
- ❑ Consistency is an application specific constraint
 - ❑ Uncertainty of data state after failure
 - ❑ Crash consistency
 - ❑ Higher order consistency points
- ❑ Atomicity of Aligned Fundamental Data Types
 - ❑ Required for consistency if additional protection is to be avoided
 - ❑ Failure atomicity as opposed to inter-process atomicity
 - ❑ Further implications when access is remote
- ❑ ▶ Application Involvement Is Required for High Availability

Key Remotable NVMP interfaces

- ❑ Directly mappable to RDMA (with extensions):
 - ❑ In NVMP 1.2:
 - ❑ OPTIMIZED_FLUSH
 - ❑ OPTIMIZED_FLUSH_AND_VERIFY
 - ❑ Under discussion:
 - ❑ ASYNC_FLUSH (initiates flushing)
 - ❑ ASYNC_DRAIN (waits for flush completion, persist fence)
 - ❑ Ordering (write-after-flush)
- ❑ Other NVM PM methods remotable via upper layer(s)

Remote access for HA ladder diagram



Asynchronous Flush

- ❑ Separates “Flush” and “Drain” stages
- ❑ Allows early scheduling of Writes without blocking
 - ❑ “Giddy-up”
 - ❑ Important for efficient concurrent processing
 - ❑ For both applications and middleware libs
- ❑ Drain allows application to ensure persistence
 - ❑ Less data remaining to flush: less wait latency
- ❑ Error conditions require careful analysis
 - ❑ Subject of NVMP TWG current work

Under Development

- ❑ Visibility vs Persistence
 - ❑ E.g.: Compare and Swap in PMEM does not necessarily yield a persistent lock!
- ❑ “Consumers of visibility” vs “Consumers of persistence”
 - ❑ Failure semantic for consumers of persistence
- ❑ Assurance of persistence integrity
 - ❑ Explicit integrity semantic, as opposed to current Best-effort
- ❑ Scope of flush
 - ❑ Conceptual “store barrier” or “order nexus”
 - ❑ Streams of stores, which are later flushed to ensure persistence
 - ❑ Flush hints (ASYNC_FLUSH)
 - ❑ Modeling these in programming interface, and protocol
 - ❑ Understanding, and guiding, platform implementation

Remote Persistent Memory - RDMA

RDMA Protocols

- ❑ RDMA adapters provide connection-oriented ordered, reliable operations
- ❑ Memory registration provides “handle” to enable remote access
- ❑ Triplet { handle, offset, length } describes remote buffer(s)
- ❑ Send, RDMA Read, RDMA Write (and others) act on buffers
- ❑ Example RDMA protocols:
 - ❑ iWARP
 - ❑ RoCE, RoCEv2
 - ❑ InfiniBand

RDMA Protocols

- ❑ Need a remote guarantee of Persistence
 - ❑ In support of OptimizedFlush()
- ❑ RDMA Write alone is not sufficient for this semantic
 - ❑ Guarantees remote visibility, but not final PM residency (durability)
 - ❑ Does not guarantee ordered delivery w.r.t. other operations (“post” only)
- ❑ An extension is desired
 - ❑ Proposed “RDMA Commit”, a.k.a. “RDMA Flush”
- ❑ Executes like RDMA Read
 - ❑ Ordered, Flow controlled, acknowledged
 - ❑ Initiator requests specific byte range to be made durable
 - ❑ Responder acknowledges only when durability complete
 - ❑ Strong consensus on these basics

RDMA Flush (concept)

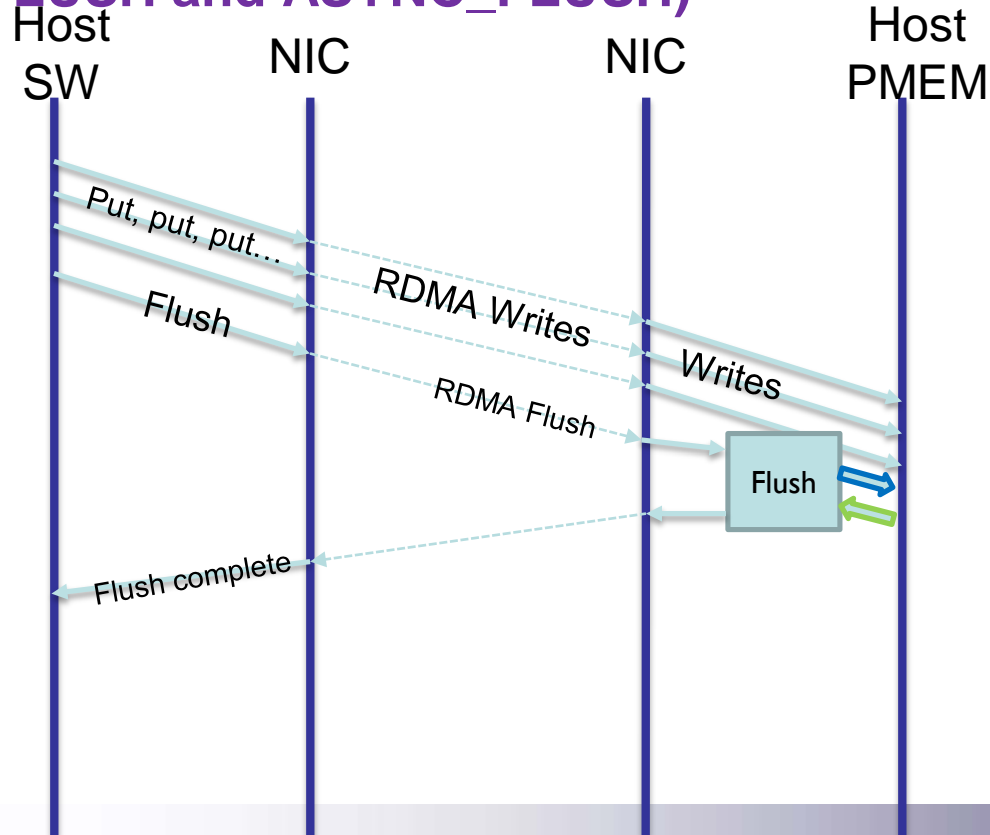
- ❑ New wire operation, and new verb
- ❑ Implementable in iWARP and IB/RoCE
- ❑ Initiating RNIC provides region, other commit parameters
 - ❑ Under control of local API at client/initiator
- ❑ Receiving RNIC queues operation to proceed in-order
 - ❑ Similar to current RDMA Read or Atomic processing
 - ❑ Subject to flow control and ordering
- ❑ RNIC pushes pending writes to targeted region
 - ❑ Alternatively, NIC may simply opt to push all writes, or all writes on this queue pair
- ❑ RNIC performs any necessary PM commit
 - ❑ Possibly interrupting CPU in current architectures
 - ❑ Possibly implementing platform “workarounds” (cache disable, etc)
 - ❑ Future (highly desirable to avoid latency) perform via PCIe operation(s)
- ❑ RNIC responds when durability of specified region is assured

Workload: Basic Replication

- ❑ Simple replication (mirroring) with writes and flushes
- ❑ Write, optionally more Writes, OPTIMIZED_FLUSH
 - ❑ No overwrite
 - ❑ No ordering dependency (but see following workloads)
 - ❑ No completions at data sink
 - ❑ Pipelined (no “bubbles”)

Writes and Flush

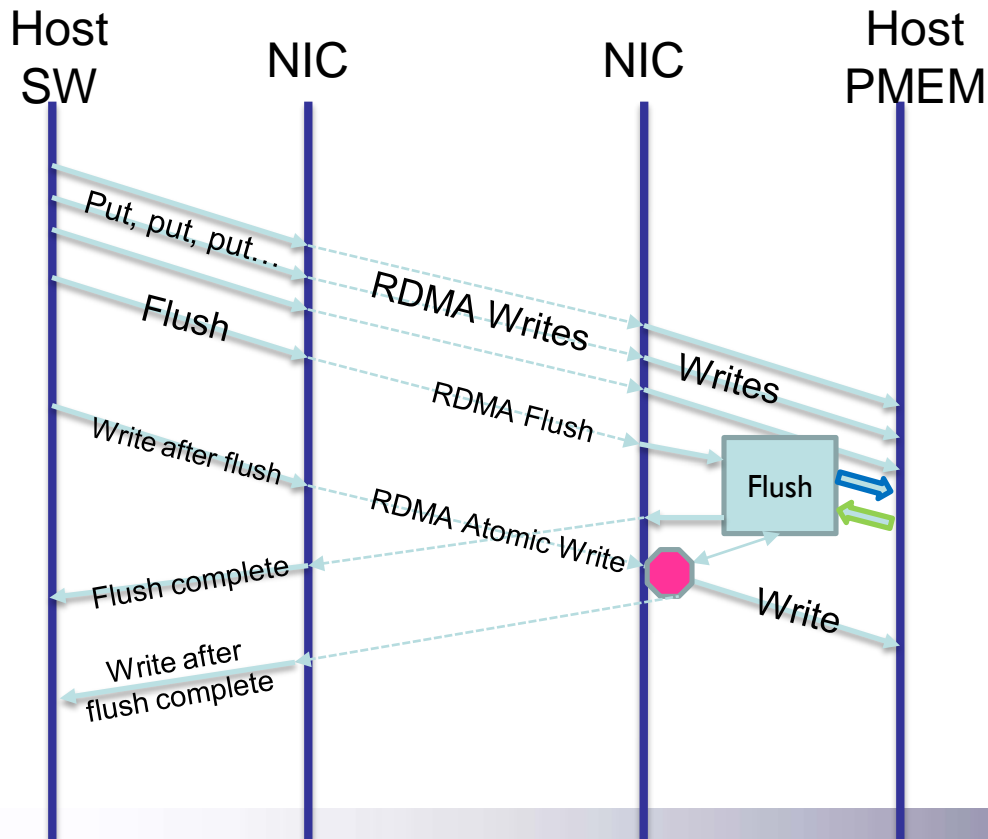
(OPTIMIZED_FLUSH and ASYNC_FLUSH)



Workload: Log Writer (Filesystem)

- ❑ For (ever)
 - { Write log record, Commit }, { Write log pointer }
 - ❑ Latency is critical
 - ❑ Log pointer cannot be placed until log record is successfully made durable
 - ❑ Log pointer is the validity indicator for the log record
 - ❑ i.e., Transaction model
 - ❑ Log records are eventually retired, buffer is circular
- ❑ Protocol implications:
 - ❑ Must ensure successful commit (e.g. ASYNC_DRAIN)
 - ❑ Potentially introduces a pipeline bubble – very bad for throughput and overall latency
 - ❑ Desire an ordering between Commit and second Write to avoid this
- ❑ Proposed solution: RDMA “Atomic Write”
 - ❑ Special Write which executes “like a Read” – ordered with other non-posted operations
 - ❑ Specific size and alignment restrictions
 - ❑ Being discussed in Standards orgs (IETF, IBTA)

Write, Flush and Write-after-Flush (ordered)



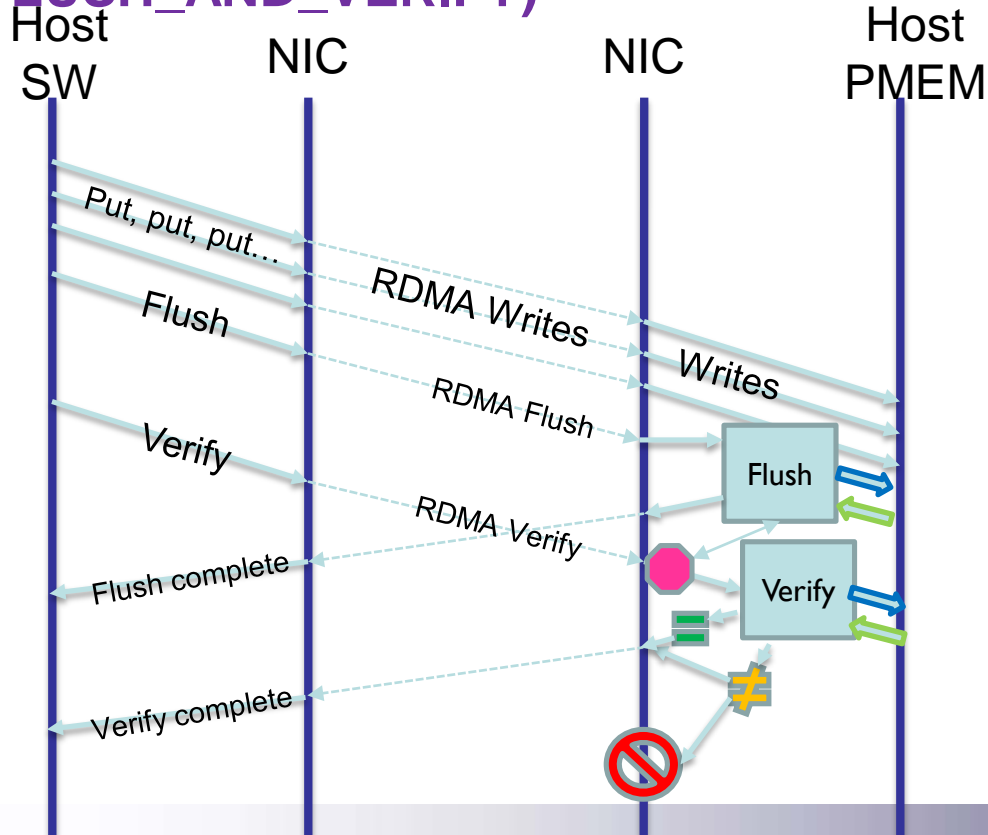
Workload: Paranoid Log Writer (Remote Data Integrity)

- ❑ Assuming we have an RDMA Write + RDMA Commit
- ❑ And the Writes + Commit all complete (with success or failure)
- ❑ How does the initiator know the data is intact?
 - ❑ Or in case of failure, which data is not intact?
 - ❑ BEFORE completing the transaction (e.g. writing the log pointer)
- ❑ Possibilities:
 - ❑ Reading back
 - ❑ extremely undesirable (and possibly not actually reading media!)
 - ❑ Signaling upper layer
 - ❑ high overhead
 - ❑ Upper layer possibly unavailable (the “Memory-Only Appliance”!)
 - ❑ Other?
- ❑ Same question applies also to:
 - ❑ Array “scrub” (verifying existing data)
 - ❑ Storage management and recovery (rebuild)
 - ❑ etc

RDMA “VERIFY”

- ❑ Concept: add integrity hashes to a new, queued, operation
- ❑ Hash algorithms to be negotiated by upper layers
- ❑ Hashing implemented in RNIC or Library “implementation”
 - ❑ Which could be in
 - ❑ Platform, e.g. storage device itself
 - ❑ RNIC hardware/firmware, e.g. RNIC performs readback/integrity computation
 - ❑ Other hardware on target platform, e.g. chipset, memory controller
 - ❑ Software, e.g. target CPU
 - ❑ Ideally, as efficiently as possible
- ❑ Semantic:
 - ❑ Source computes expected hash of region, sends to target
 - ❑ Target computes and compares
 - ❑ If *matches*, returns computed hash value as “success”
 - ❑ If *mismatches*, optionally:
 - ❑ Return computed hash value (scrub/rebuild scenario)
 - ❑ OR
 - ❑ Break the connection with a hash-verify-mismatch code
 - ❑ Which fences any subsequent operations, and forces recovery

Write, Flush and Verify (OPTIMIZED_FLUSH_AND_VERIFY)



RDMA Extension Implications on Programming Model

- ❑ Strengthens need for `ASYNC_FLUSH`
- ❑ Increased imprecision of errors
 - ❑ RDMA connection is simply broken
- ❑ Need for bubbling up `AtomicWrite` completion?
- ❑ Asynchronous `Verify` indication?
- ❑ `Verify Fail` imprecision in connection-break mode

Ongoing NVMP TWG work

- ❑ Core NVM PM v1.3 update
 - ❑ Current work in progress (in the NVMP TWG)
- ❑ Asynchronous Flush
- ❑ Incorporate implementation learnings
 - ❑ Optimized Flush, Deep Flush, Flush on Fail
 - ❑ Interaction between NVMP and C memory model (!)
 - ❑ Visibility versus Persistence
- ❑ Continue Remote Access for HA work
 - ❑ Greater RDMA mapping detail
 - ❑ Efficient remote programming interface models
 - ❑ RDMA and platform requirements clarified
 - ❑ Errors, error handling, error recovery in remote scenario
 - ❑ Collaboration with Open Fabrics Alliance OFI WG
- ❑ Scope of flush, flush barriers, analysis
- ❑ “Flush on Fail Fail” (failure of persistence) analysis

Thank you!

Questions?

SNIA NVM Programming TWG:

<https://www.snia.org/forums/sssi/nvmp>

PM Programming Model:

https://www.snia.org/tech_activities/standards/curr_standards/npm

PM Remote Access for HA (and other papers):

https://www.snia.org/tech_activities/standards/whitepapers