SDC

STORAGE DEVELOPER CONFERENCE

SNIA ■ SANTA CLARA, 2017

Remote Persistent Memory With Nothing But Net

Tom Talpey

Microsoft

Outline

- Aspiration
  - RDMA NIC as a Persistent Memory storage adapter
- Steps to there:
  - Flush
  - Write-after-flush
  - Integrity
  - Privacy
  - QoS
- Some protocol, some not!

Discussed at past SDCs…

- Remote Flush
- SMB3 (etc) Push Mode
- Storage QoS

- Further development, and new thinking

RDMA Flush

- Need a remote guarantee of Durability
  - In support of SNIA NVMP interface OptimizedFlush()
- RDMA Write alone is not sufficient for this semantic
  - Completion at sender does not mean data was placed
    - NOT that it was even sent on the wire, much less received
    - Some RNICs give stronger guarantees, but never that data was stored remotely
  - Processing at receiver means only that data was accepted
    - NOT that it was sent on the bus
    - Segments can be reordered, by the wire or the bus
    - Only an RDMA completion at receiver guarantees placement
      - And placement != commit/durable
- An extension is required

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
  - Like non-posted RDMA Read or Atomic processing currently
  - Subject to flow control and ordering
- RNIC pushes pending writes to targeted region(s)
  - Alternatively, NIC may simply opt to push all writes
- RNIC performs any necessary PM commit
  - Possibly interrupting CPU in current architectures
  - Future (highly desirable to avoid latency) perform via PCIe
- RNIC responds when durability is assured

IBTA Discussion

- Broad agreement on RDMA Flush approach
- Discussion continues on semantic details
  - Per-region, per-segment, or per-connection?
  - Implications on the API (SNIA NVMP TWG)
    - Important to converge
- Ongoing discussion on “non-posted write”
  - Provides write-after-flush semantic
  - Useful for efficient log write, transactions, etc
    - While avoiding pipeline bubbles

PCI Protocol

- PCI support for Flush from RDMA adapter
  - To Memory, CPU, PCI Root, PM device, PCIe device, …
  - Avoids CPU interaction
  - Supports strong data persistency model
- Perform equivalent of Flush/Commit
  - Without CPU involvement (and latency)
- Possibly achievable without full-blown extensions
  - “Hints” etc
  - Accelerate adoption with platform-specific support

Other RDMA Semantics

- Goal to eliminate pipeline bubbles for:
  - Atomically-placed write-after-flush
    - E.g. “log pointer update”
  - Immediate data, or an ordered flush-following Send
    - E.g. to signal upper layer
  - These may be implemented in ordered operations
- Additional processing, e.g. integrity check (see later in this deck)
- Semantics will be driven by workload (*application*) requirements, e.g.:
  - Small log-write scenario will always commit
  - Bulk data movement will prefer batching

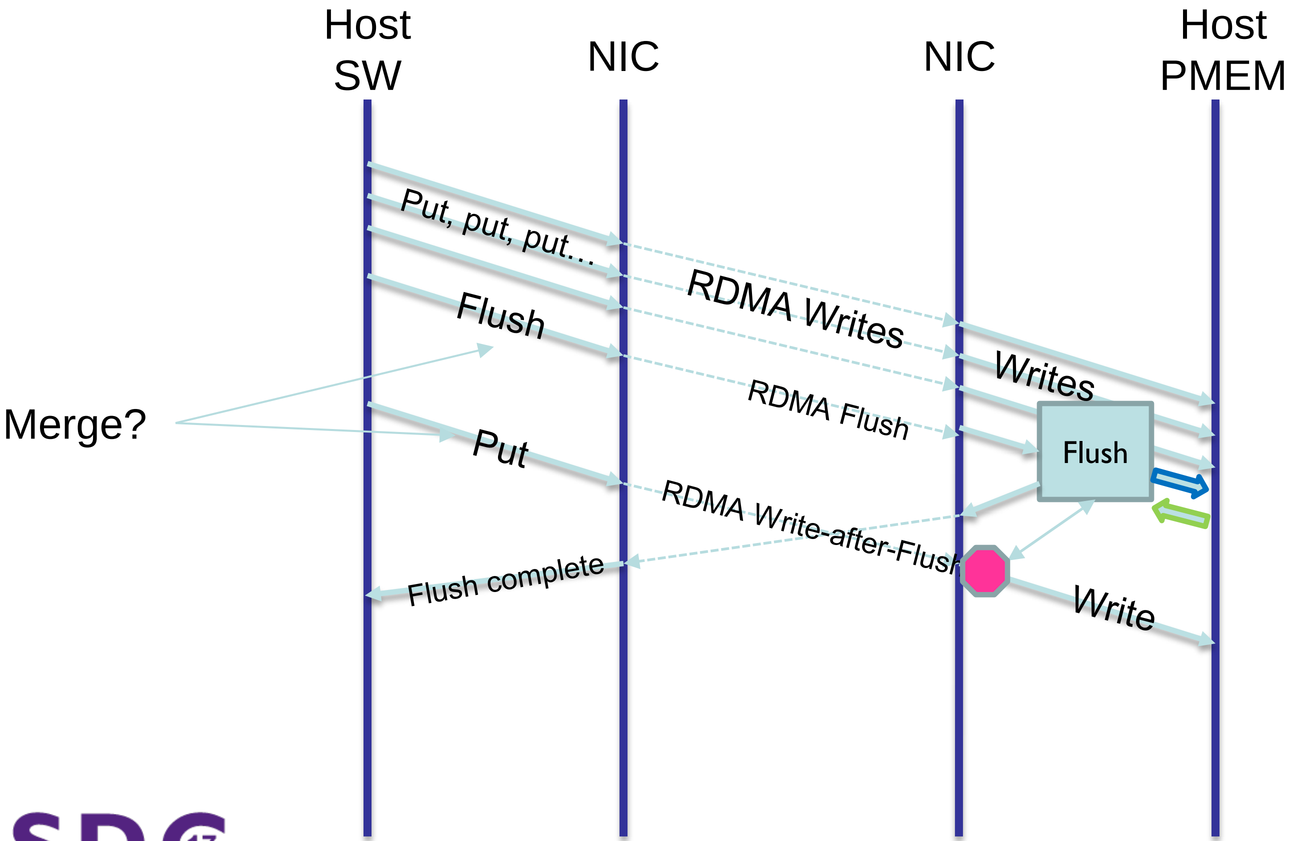
Example: Log Writer (Filesystem)

- For (ever)

  { Write log record, Commit }, { Write log pointer (, Commit) }

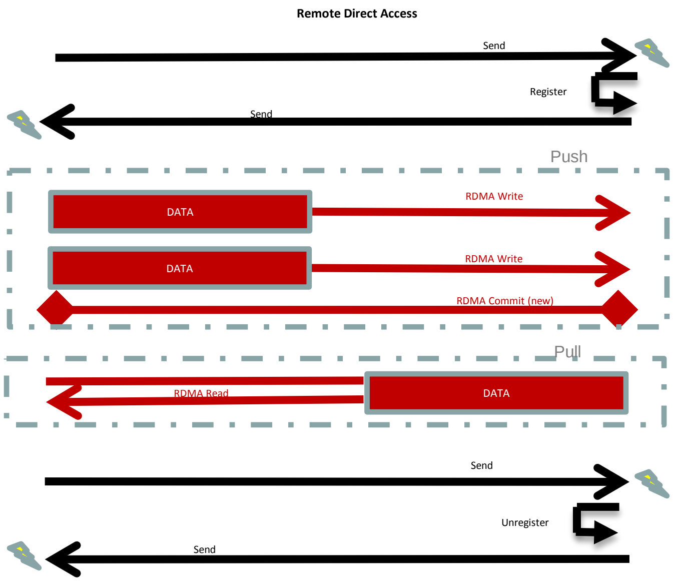
  - Latency is critical
  - Log pointer cannot be placed until log record is successfully made durable
    - Log pointer is the validity indicator for the log record
    - Transaction model
  - Log records are eventually retired, buffer is circular
- Protocol implications:
  - Must wait for first commit (and possibly the second)
  - Wait introduces a pipeline bubble – very bad for throughput and overall latency
  - Desire an ordering between Commit and second Write to avoid this
- Possible solution: "**Non-posted write**"
  - Special Write which executes "like a Read" – ordered with other non-posted operations (Flush, Read, etc) to enable streaming
    - Specific size and alignment restrictions
  - Being discussed in IBTA

Writes, Flush and Write-after-Flush



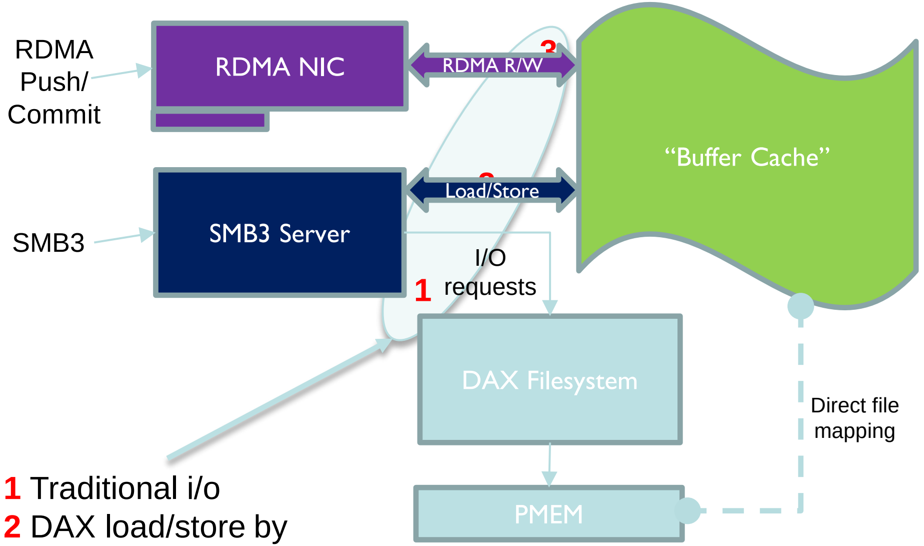
Example: SMB3 Push Mode

- Basic steps:
  - Open DAX-enabled file
  - Obtain a lease
  - Request a push-mode registration
  - While (TRUE)
    - Push (or pull) data
    - Commit data to durability
  - Release registration
  - Drop lease
  - Close handle



Example: Going Remote – SMB3

- SMB3 RDMA and "Push Mode" discussed at previous SNIA Storage Developers Conferences
- Enables zero-copy remote read/write to DAX file
  - Ultra-low latency and overhead
- Phases 2, 3 can be enabled even before RDMA extensions become available, with only slight incremental cost



1 Traditional i/o
2 DAX load/store by SMB3 Server
3 Push Mode direct from RDMA NIC

12

So Far, So Good

- Only two RDMA protocol extensions
- Or maybe one!

But we still need a CPU

- For all that other storage processing
  - Integrity
  - Privacy
  - QoS (fairness, management, congestion)

CPU-less

- Can we do these with only a NIC?
  - And, without more protocol extensions?
- Yes.
  - Mostly yes.

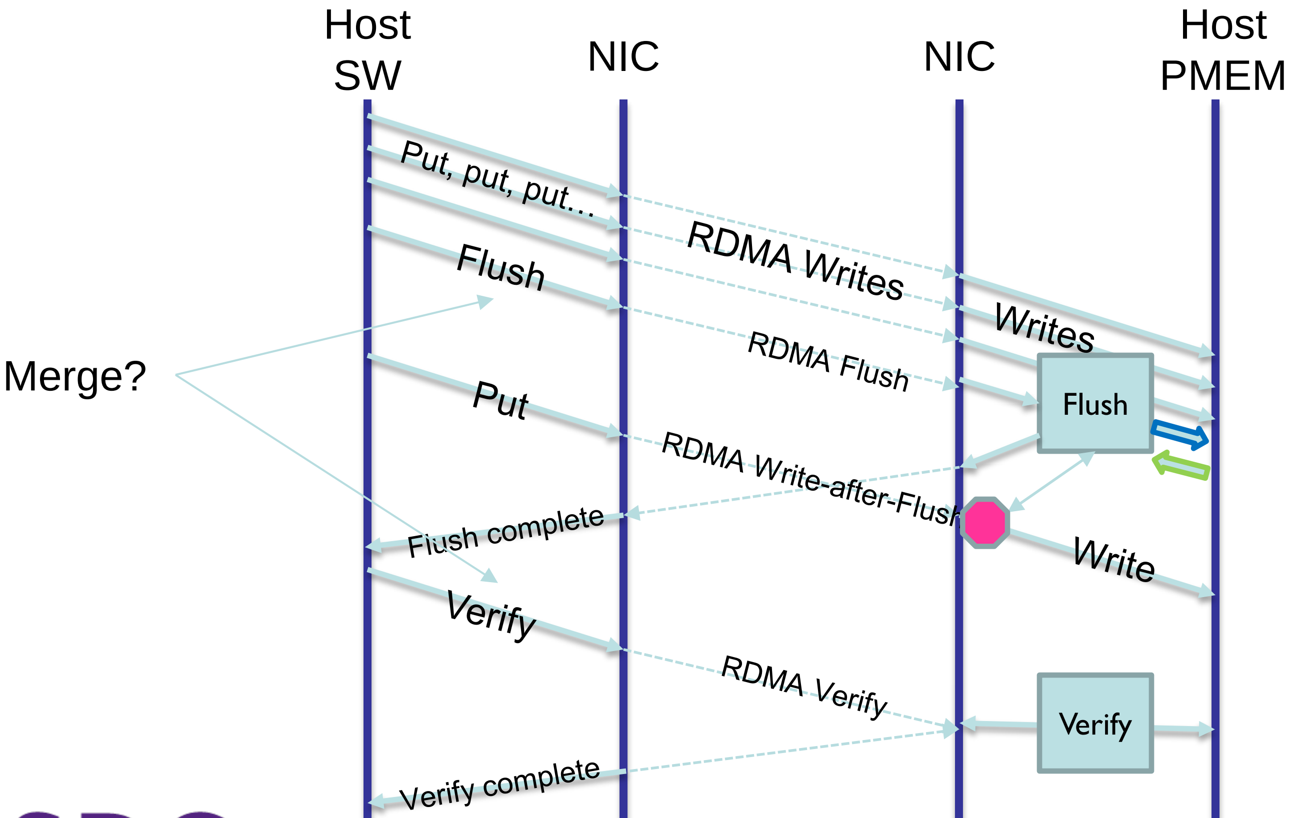
Remote Data Integrity

- Assuming we have an RDMA Write + RDMA Flush
- And the Flush + 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?
- 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”
  - Storage management and recovery
  - etc

RDMA "VERIFY"

- Concept: add integrity hashes to a new operation
  - Or, possibly, piggybacked on Flush (which would mean only one protocol change)
  - Note, not unlike SCSI T10 DIF
- 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
- Options:
  A. Source requests hash computation, receives hash as result, performs own comparison
  B. Source sends hash to target, target computes and compares, returns success/failure
  C. ???
- Roughly mapped to SNIA NVMP TWG OptimizedFlushAndVerify()

Write, Flush and Verify



Privacy

- Upper layers protect their send/receive messages today
- But RDMA direct transfers are not protected
  - No RDMA encryption standard
- Desire to protect User Data with User Key
  - Not global, machine or connection key!
  - Rules out IPsec, TLS, DTLS
- Why not just use the on-disk crypto?
  - Typically a block cipher, requiring block not byte access
  - No integrity – requires double computation

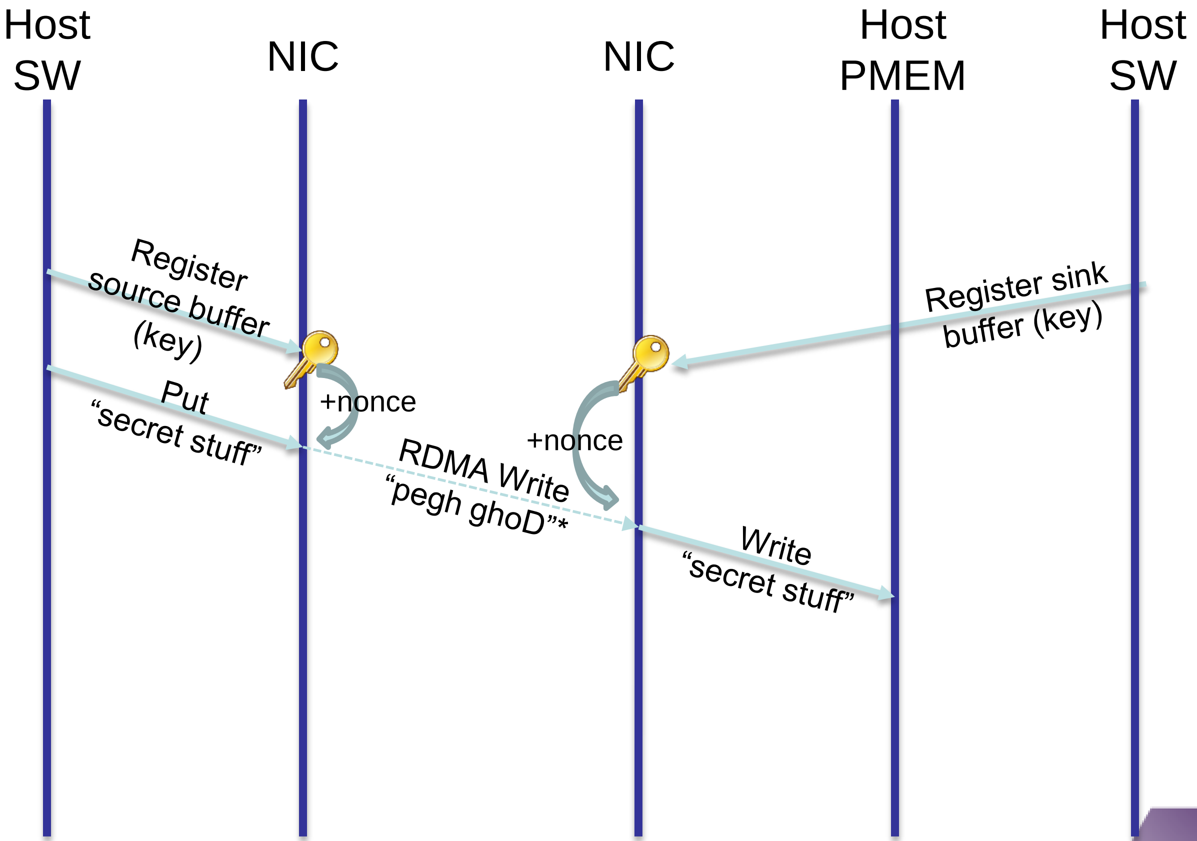
RDMA Privacy

- Authenticated stream cipher (e.g. AES-CCM/GCM as used by SMB3)
  - Provides wire privacy *and* integrity, efficiently
    - (not at-rest – still need RDMA Verify)
  - Arbitrary number of bytes per transfer
  - Shares cipher and keying with upper layer
  - But, how to plumb key into RDMA NIC message processing?
- Enhance RDMA Memory Regions
  - Which does not require RDMA protocol change!

Memory Region Keys

- Extend MR verb and NIC TPT to include key
  - Handle = Register(PD, buffer, mode, **key**)
- Keys held by upper layer, **user** policy, and passed down to NIC
- NIC uses key when reading or writing each region

RDMA Write encryption



* Klingon ☺

Cipher Housekeeping

- Authenticated ciphers typically employ nonces (e.g. AES-GCM)
  - Same {key,nonce} pair used at each end to encrypt/decrypt each message
  - Never reuse nonce for different payload!
    - Upper layer must coordinate nonce usage with RDMA layer
    - RDMA layer must consider when retrying
  - NIC may derive nonce sequence from RDMA connection
    - E.g. from RDMA msn. Not from the MR!
  - Alternatively, prepend/append to data buffer
  - Upper layer consumes nonce space, too
- Re-keying necessary when nonce space exhausted!
  - Nonces are large (SMB3 employs 11 bytes for GCM), but require careful management and sharing with ULP
  - Key management is upper layer responsibility, as it should be

Protecting the Network

- Upper layers have no trouble saturating 40, 56, even 100Gb networks with RDMA today
- Memory can sink writes at least this fast
- ► Networks will rapidly congest
- Rate control, fairness and QoS are required in the RDMA NIC

RDMA QoS

- Must implement QoS rate control in NIC
- Simplistically, bandwidth limiting
- More sophisticated approach desirable
  - Classification/end-to-end QoS techniques
    - SDC2014 presentation (“resources” slide)
  - Software-Defined Network techniques
    - Generic Flow Table-based policy
- Existing support in many Enterprise-class NICs

Putting It All Together

- Ok, assuming we have
  - Durability, ordered atomicity, privacy/integrity and rate control/QoS
- Then we support the “hot path” completely in the RDMA NIC
- But, is that all?
- Of course, no…

We Still Need an Upper Layer

- Connection management
- Authentication
  - Key derivation and provisioning
  - Nonce management
- Authorization
  - Granting and revoking of remote “push handles”
- Assigning QoS policy
- And all the other things Upper Layers already do

- Think of this as an “offload” for the PM data handling

Summary

- With Persistent Memory as the storage media,
- And the above extensions…
- We enable an RDMA-only remote storage access method
  - Avoiding CPU and upper layer processing for most operations
- And rock-bottom latencies (single-digit µS)

Resources

- SNIA NVM Programming TWG:
  - http://www.snia.org/forums/sssi/nvmp
- Open Fabrics Workshop 2017:
  - Remote Persistent Memory Access - Workload Scenarios and RDMA Semantics
  - https://www.openfabrics.org/images/eventpresos/2017presentations/405_RemotePM_TTalpey.pdf
- SDC 2016:
  - Low Latency Remote Storage – A Full-stack View
  - https://www.snia.org/sites/default/files/SDC/2016/presentations/persistent_memory/Tom_Talpey_Low_Latency_Remote_Storage_A_Full-stack_View.pdf
- SDC 2014
  - Storage Quality of Service for Enterprise Workloads
  - https://www.snia.org/sites/default/files/TomTalpey_Storage_Quality_Service.pdf

SDC

STORAGE DEVELOPER CONFERENCE

SNIA ■ SANTA CLARA, 2017

Thank you!