12th ANNUAL WORKSHOP 2016

RDMA EXTENSIONS FOR REMOTE PERSISTENT MEMORY ACCESS

Tom Talpey

Microsoft

April 6, 2016

PROBLEM STATEMENT

- **Provide applications with remote access to Non-Volatile/Persistent Memory storage at ultra-low latency**

- **Examine storage protocol and RDMA protocol extensions in support of applications' workload**

- **Explore implications on RDMA implementations**

RDMA-AWARE STORAGE PROTOCOLS

- **Ecosystem – Enterprise / Private Cloud-capable storage protocols**
  - Scalable, manageable, broadly deployed
  - Provide authentication, security (integrity AND privacy)
  - Natively support parallel and highly available operation
- **SMB3 with SMB Direct**
- **NFS/RDMA**
- **iSER**
- **Others exist**

STORAGE LATENCIES DECREASING

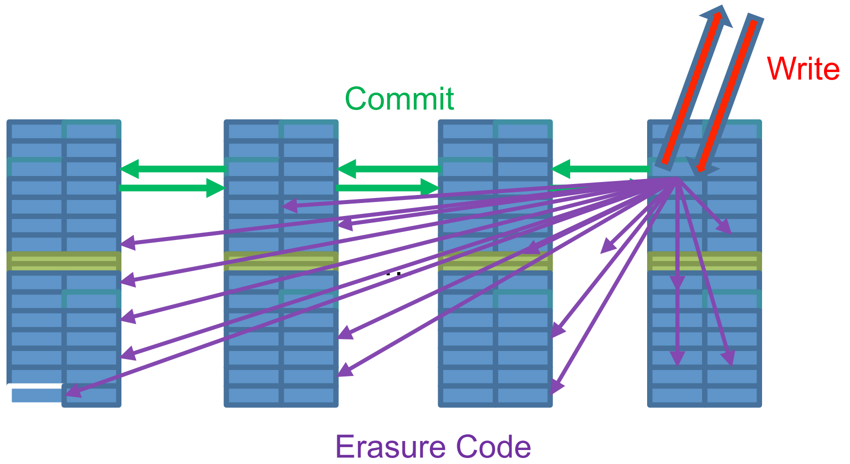
- **Write latencies of storage protocols (e.g. SMB3) today down to 30-50us on RDMA**
  - Good match to HDD/SSD
  - Stretch match to NVMe
  - PM, not so much ☺
- **Storage workloads are traditionally highly parallel**
  - Latencies are mitigated
- **But workloads are changing:**
  - Write replication adds a latency hop
  - Write latency critical to reduce

Technology	Latency (high)	Latency (low)	IOPS
HDD	10 msec	1 msec	100
SSD	1 msec	100 µsec	100K
NVMe	100 µsec	10 µsec (or better)	500K+
PM	< 1 µsec	(~ memory speed)	BW/size (>>1M/DIMM)

Orders of magnitude decreasing

WRITES, REPLICATION, NETWORK

- **Writes (with possible erasure coding) greatly multiplies network I/O demand**
  - Small, random
    - Virtualization, Enterprise applications
  - MUST be replicated and durable
    - A single write creates multiple network writes
    - And possible later erasure coding
  - The “2-hop” issue
- **All such copies must be made durable before responding**
  - Therefore, latency of writes is critical!



- **Reads**
  - Small, random are latency sensitive
  - Large, more forgiving
    - But recovery/rebuild are interesting/important

APIS AND LATENCY

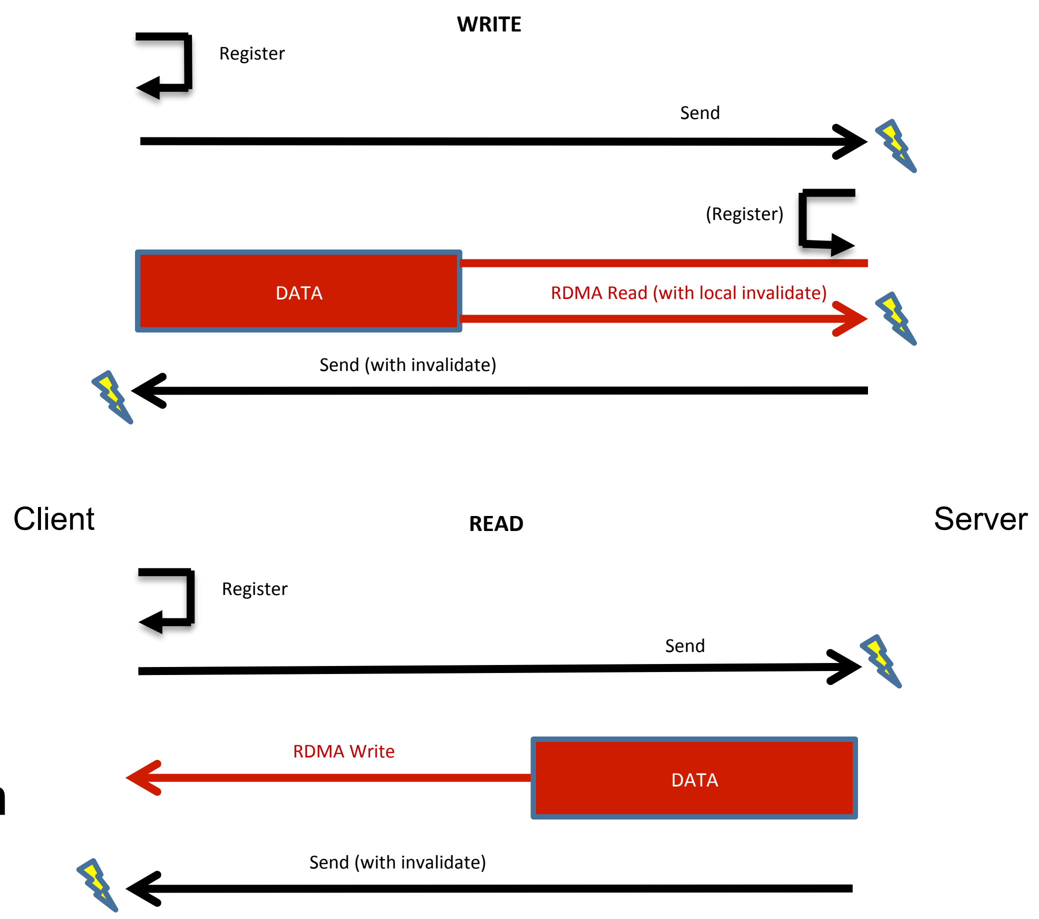
- **APIs also shift the latency requirement**
- **Traditional Block and File are often parallel**
- **Memory Mapped and PM-Aware APIs not so parallel**
  - Effectively a Load/Store expectation, nonblocking
  - Memory latency, with possibly expensive Commit
  - Local caches can improve Read (load) but not Write (store/remotely durable)

MANY LAYERS ARE INVOLVED

- **Storage layers**
  - SMB3 and SMB Direct
  - NFS, pNFS and NFS/RDMA
  - iSCSI and iSER
- **RDMA Layers**
  - iWARP
  - RoCE, RoCEv2
  - InfiniBand
- **I/O Busses**
  - Storage (Filesystem, Block e.g. SCSI, SATA, SAS, …)
  - PCIe
  - Memory

RDMA TRANSFERS – STORAGE PROTOCOLS TODAY

- **Direct placement model (simplified and optimized)**
  - Client advertises RDMA region in scatter/gather list
  - Server performs all RDMA
    - More secure: client does not access server's memory
    - More scalable: server does not preallocate to client
    - Faster: for parallel (typical) storage workloads
  - SMB3 uses for READ and WRITE
    - Server ensures durability
    - NFS/RDMA, iSER similar
- **Interrupts and CPU on both sides**

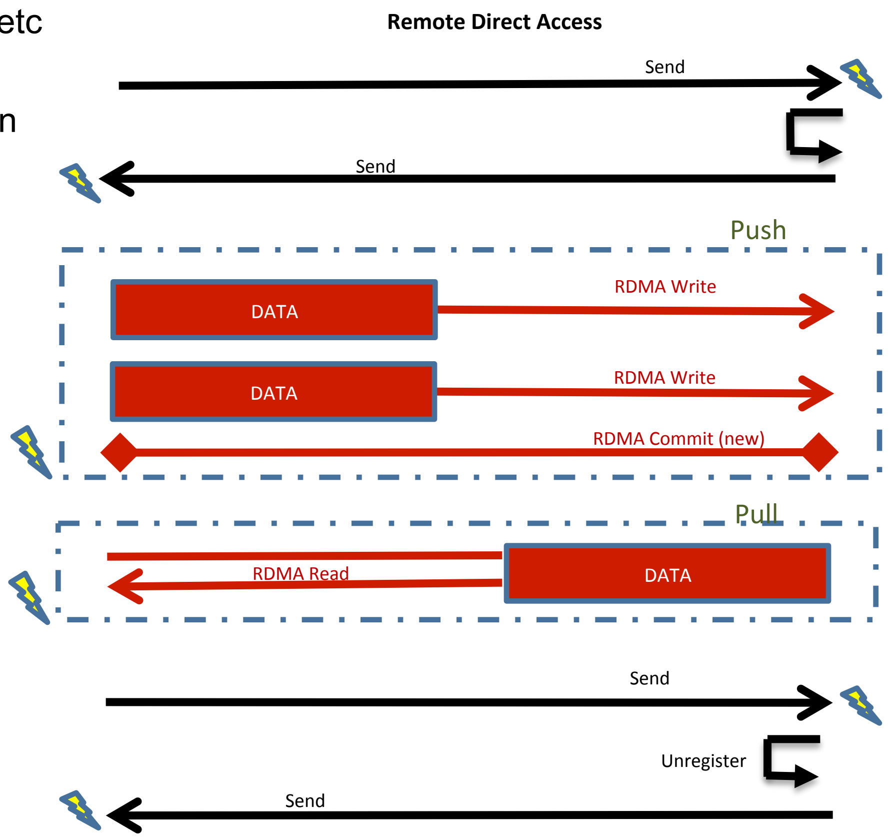


LATENCIES

- **Undesirable latency contributions**
  - Interrupts, work requests
    - Server request processing
    - Server-side RDMA handling
  - CPU processing time
    - Request processing
  - I/O stack processing and buffer management
    - To “traditional” storage subsystems
  - Data copies
- **Can we reduce or remove all of the above to PM?**

RDMA PUSH MODE (SCHEMATIC)

- **Enhanced direct placement model**
  - Client requests server resource of file, memory region, etc
    - MAP_REMOTE_REGION(offset, length, mode r/w)
  - Server pins/registers/advertises RDMA handle for region
  - Client performs all RDMA
    - RDMA Write to regionll
    - RDMA Read from region ("Pull mode")
    - No requests of server (no server CPU/interrupt)
      - Achieves near-wire latencies
  - Client remotely commits to PM (new RDMA operation!)
    - Ideally, no server CPU interaction
    - RDMA NIC optionally signals server CPU
    - Operation completes at client only when remote durability is guaranteed
- **Client periodically updates server via master protocol**
  - E.g. file change, timestamps, other metadata
- **Server can call back to client**
  - To recall, revoke, manage resources, etc
- **Client signals server (closes) when done**



STORAGE LAYERS PUSH MODE (HYPOTHETICAL)

- **SMB3 (hypothetical)**
  - Setup – a new create context or FSCTL, registers and takes a lease
  - Write, Read – direct RDMA access by client
  - Commit – Client requests durability, SMB2_FLUSH-like processing
  - Callback – Server manages client access, similar to oplock/lease break
  - Finish – Client access complete, close or lease return
- **NFSv4/RDMA (hypothetical)**
  - Setup – new NFSv4.x Operation, registers and offers delegation (or pNFS layout)
  - Write, Read – direct RDMA access by client
  - Commit – Client requests durability, NFS4_COMMIT-like processing
  - Callback – via backchannel, Similar to current delegation or layout recall
  - Finish – close or delegreturn/layoutreturn
- **iSER (very hypothetical)**
  - Setup – a new iSER operation – registers and advertises buffers
  - Write – a new Unsolicited SCSI-In operation
    - Implement RDMA Write within initiator to target buffer
    - No Target R2T processing
  - Read – a new Unsolicited SCSI-Out operation
    - Implement RDMA Read within initiator from target buffer
    - No Target R2T processing
  - Commit – a new iSER / modified iSCSI operation
    - Perform Commit, via RDMA with optional Target processing
    - Leverage FUA processing for signaling if needed/desired
  - Callback – a new SCSI Unit Attention
    - ???
  - Finish – a new iSER operation

RDMA PROTOCOLS

- **Need a remote guarantee of Durability**
- **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
  - No Commit operation
- **Certain platform-specific guarantees can be made**
  - But the remote client cannot know them
  - E.g. RDMA Read-after-RDMA Write (which won't generally work)

RDMA PROTOCOL EXTENSION

- **Two "obvious" possibilities**
  - RDMA Write with placement acknowledgement
    - Advantage: simple API – set a "push bit"
    - Disadvantage: significantly changes RDMA Write semantic, data path (flow control, buffering, completion). Requires creating a "Write Ack".
    - Requires significant changes to RDMA Write hardware design
      - And also to initiator work request model (flow controlled RDMA Writes would block the send work queue)
    - **Undesirable**
  - RDMA "Commit"
    - New operation, flow controlled/acknowledged like RDMA Read or Atomic
    - Disadvantage: new operation
    - Advantage: simple API – "flush", operates on one or more regions (allows batching), preserves existing RDMA Write semantic (minimizing RNIC implementation change)
    - **Desirable**

RDMA COMMIT (CONCEPT)

- **RDMA Commit**
  - New wire operation
  - Implementable in iWARP and IB/RoCE
- **Initiating RNIC provides region list, other commit parameters**
  - Under control of local API at client/initiator
- **Receiving RNIC queues operation to proceed in-order**
  - Like RDMA Read or Atomic processing currently
  - Subject to flow control and ordering
- **RNIC pushes pending writes to targeted regions**
  - Alternatively, NIC may simply opt to push all writes
- **RNIC performs PM commit**
  - Possibly interrupting CPU in current architectures
  - Future (highly desirable to avoid latency) perform via PCIe
- **RNIC responds when durability is assured**

OTHER RDMA COMMIT SEMANTICS

- **Desirable to include other semantics with Commit:**
  - Atomically-placed data-after-commit
    - E.g. “log pointer update”
  - Immediate data
    - E.g. to signal upper layer
  - An entire message
    - For more complex signaling
    - Can be ordinary send/receive, only with new specific ordering requirements
  - Additional processing, e.g. integrity check
  - These may be best implemented in ordered following operations
- **Decisions will be workload-dependent**
  - Small log-write scenario will always commit
  - Bulk data movement will permit batching

LOCAL RDMA API EXTENSIONS (CONCEPT)

- **New platform-specific attributes to RDMA registration**
  - To allow them to be processed at the server *only*
  - No client knowledge – ensures future interop
    - E.g. don't want clients performing RDMA Read with flush expectations
- **New local PM memory registration**
  - Register(region[], PMType, mode) -> Handle
    - PMType includes type of PM
      - i.e. plain RAM, or "commit required", or PCIe-resident, or any other local platform-specific processing
    - Mode includes disposition of data
      - Read and/or write
      - Cacheable after operation (needed by CPU on data sink)
    - Handle is processed in receiving NIC under control of original Mode

LOCAL RDMA API EXTENSIONS

- **Transparency is possible when upper layer provides Completions (e.g. messages or immediate data)**
  - Commit to durability can be piggybacked by data sink upon signaling
  - Upper layer may not need to explicitly Commit
  - Dependent on upper layer and workload
- **Can apply to RDMA Write with Immediate**
- **Or … ordinary receives**
  - Ordering of operations is critical:
    - Such RDMA Writes cannot be allowed to “pass” durability
  - Therefore, protocol implications exist
- **Completions imply latency, but transparency is good for rapid adoption**
  - Possible good first-phase approach

PLATFORM-SPECIFIC EXTENSIONS

- **PCI extension to support Commit**
  - Allow NIC to provide durability directly and efficiently
  - To Memory, CPU, PCI Root, PM device, PCIe device, …
  - Avoids CPU interaction
  - Supports strong data consistency model
- **Performs equivalent of:**
  - CLFLUSHOPT (region list)
  - PCOMMIT
- **Or if NIC is on memory bus or within CPU complex…**
  - Other possibilities exist
  - Platform-specific implementations, on platform-local basis
- **Standard extensions are most desirable**

LATENCIES (EXPECTATIONS)

- **Single-digit microsecond remote Write+Commit**
  - Push mode minimal write latencies (2-3us + data wire time)
  - Commit time NIC-managed and platform+payload dependent
  - Note, this is order-of-magnitude improvement over today's transfer mode
    - 30-50us as mentioned
- **Remote Read also possible**
  - Roughly same latency as write, but without commit
- **No server interrupt**
  - Zero server CPU overhead
  - Once RDMA and PCIe extensions in place
- **Single client interrupt**
  - Moderation and batching can reduce further when pipelining
- **Deep parallelism with Multichannel and flow control management**

RDMA IMPLICATIONS

- **Remote PM access is a new upper layer protocol**
  - It will have RNIC and Verbs implications
- **Possible implications on:**
  - RDMA operation “workload”
  - Resources
  - Memory registration semantics
- **Speculation/exploration slides follow**

RDMA PM WORKLOAD

- **“Push mode” workload very simple:**
  - One-time setup to connect, authenticate, request registered memory
  - Following: pure RDMA and commit operation stream
- **One-sided workload – all programmatic activity on the initiator**
- **Target-side CPU nearly silent**
  - Except for metadata updates, recalls, revocation, etc.
- **Implication: target-side efficiency completely in the RNIC**
- **A return to the “storage adapter” model?**

RDMA RESOURCE IMPLICATIONS #1

- **Large regions registered**
  - Regions will span large segments of PM (e.g. entire DIMM)
  - Or perhaps long scatter lists mapping single file (e.g. virtual hard disk)
- **Note – 1 TB is 40 bits of addressing**
  - Expect 6 TB in first-gen Intel 3DXP! (43 bits, 2^{31} pages)
  - This will favor region-based TPT's
- **The DMA MR / Privileged MR / Stag0 cannot span "all physical"**
  - It would cross domains from DRAM to PMEM to IO space
  - These domains require different durability properties and commit methods
- **IO size changes**
  - Very small (bytes, cachelines)
  - Very large (entire regions)

RDMA RESOURCE IMPLICATIONS #2

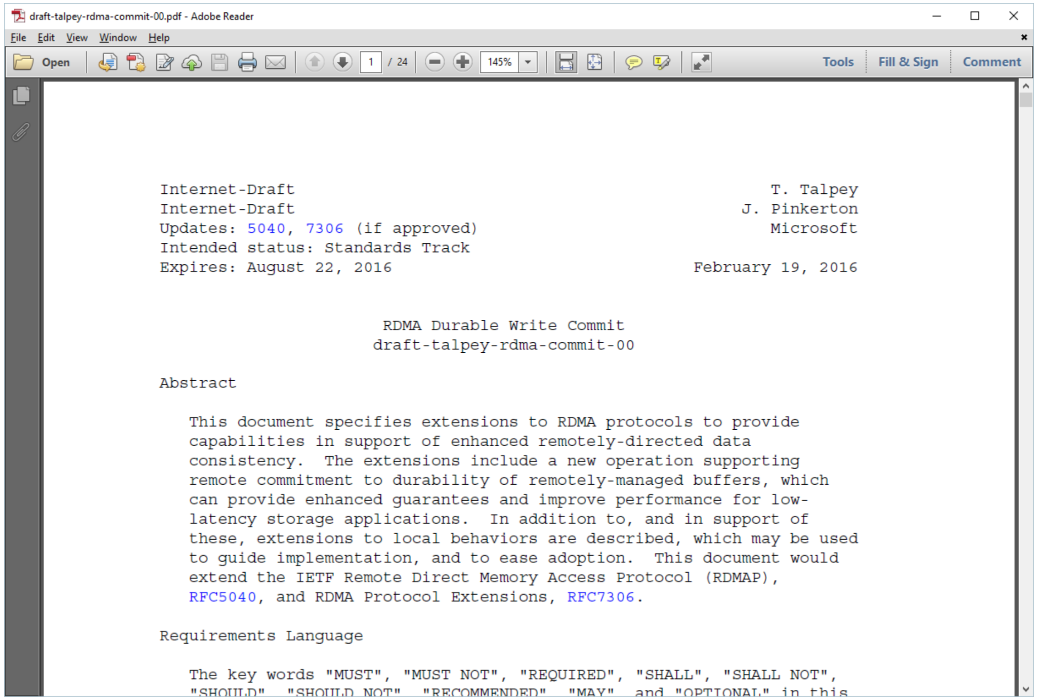
- **Long-lived target regions**
  - Regions will not be registered/invalidated per-IO
  - Smaller number of regions needed – 1/DIMM, 1/object, etc?
  - More Protection Domains? (for better isolation: 1/object?)
- **Greatly reduced initiator (client) regions**
  - No RDMA from server to client means no need for remote access to client
- **QP count changes?**
  - Not sure about this yet
- **IRD/ORD or similar flow control limits**
  - Commit operation may require its own queue
  - Commit latency may require higher than existing RDMA Read or Atomic queueing limit to fill pipeline
- **Summary: something of a seismic shift in RNIC resource requirements**

VERBS IMPLICATIONS

- **Commit operation**
  - With potentially complex scatter list
  - New “commit fail” semantics which return status and do not break connection
- **Memory registration**
  - Region properties as mentioned earlier:
    - Type of PM
    - Commit disposition
    - Other properties (integrity, …)
  - Memreg verb must support large offsets (40 bits and up)
    - Each individual RDMA wire operation probably still ok at 32 bits.
  - Memreg verb may need to “split” regions or return short results
    - Because a single memory handle cannot span devices or device types
  - DMA MR possibly obsolete
- **However – no Verb-layer negotiation**
  - This is best left to the upper layers, as is done now

EXTERNAL EFFORTS

- **Requirements and Protocol**
  - For RDMA Commit operation
  - Also local PM behaviors
    - Memory registration
  - Independent of transport
    - Applies to iWARP, IB, RoCE
- **IETF Working Group**
  - STORM: RDMA (iWARP) and Storage (iSCSI)
  - Recently closed, but active for discussion
  - Another WG, or individual process TBD
- **Also discussing in**
  - IBTA (IB/RoCE) expected
  - SNIA NVM TWG
  - Open Fabrics DS/DA? etc.



Internet-Draft T. Talpey
Internet-Draft J. Pinkerton
Updates: 5040, 7306 (if approved) Microsoft
Intended status: Standards Track
Expires: August 22, 2016 February 19, 2016

RDMA Durable Write Commit
draft-talpey-rdma-commit-00

Abstract

This document specifies extensions to RDMA protocols to provide capabilities in support of enhanced remotely-directed data consistency. The extensions include a new operation supporting remote commitment to durability of remotely-managed buffers, which can provide enhanced guarantees and improve performance for low-latency storage applications. In addition to, and in support of these, extensions to local behaviors are described, which may be used to guide implementation, and to ease adoption. This document would extend the IETF Remote Direct Memory Access Protocol (RDMAP), RFC5040, and RDMA Protocol Extensions, RFC7306.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT",

https://datatracker.ietf.org/doc/draft-talpey-rdma-commit/

CLOSING QUESTIONS

- **Getting to the right semantic?**
  - Discussion in multiple standards groups (PCI, RDMA, Storage, …)
  - How to coordinate these discussions?
  - Implementation in hardware ecosystem
  - Drive consensus from upper layers down to lower layers!
- **What about new API semantics?**
  - Does NVML add new requirements?
  - What about PM-aware filesystems (DAX/DAS)?
- **Other semantics – or are they Upper Layer issues?**
  - Authentication?
  - Integrity/Encryption?
  - Virtualization?

OPENFABRICS ALLIANCE

12th ANNUAL WORKSHOP 2016

THANK YOU

Tom Talpey

Microsoft

Microsoft