SDC
STORAGE DEVELOPER CONFERENCE
SNIA SANTA CLARA, 2016

Low Latency Remote Storage: A Full-Stack View

Tom Talpey
Microsoft

Outline

- Problem Statement (review)
  - http://www.snia.org/sites/default/files/SDC15_presentations/persistant_mem/Talpey-Remote_Access_Storage.pdf
- Platform support
- RDMA support
- SMB3 support
- Windows Server 2016

Problem Statement

- "Storage Class Memory"
  - A new, disruptive class of storage
  - Nonvolatile medium with RAM-like performance
    - Low latency, high throughput, high capacity
  - Resides on memory bus
    - Byte addressable, byte or block semantics
  - Or also on PCIe bus
    - Block semantics predominate
- Remote access is necessary
  - Local-only storage lacks resiliency
  - New features needed throughout storage, network and platform

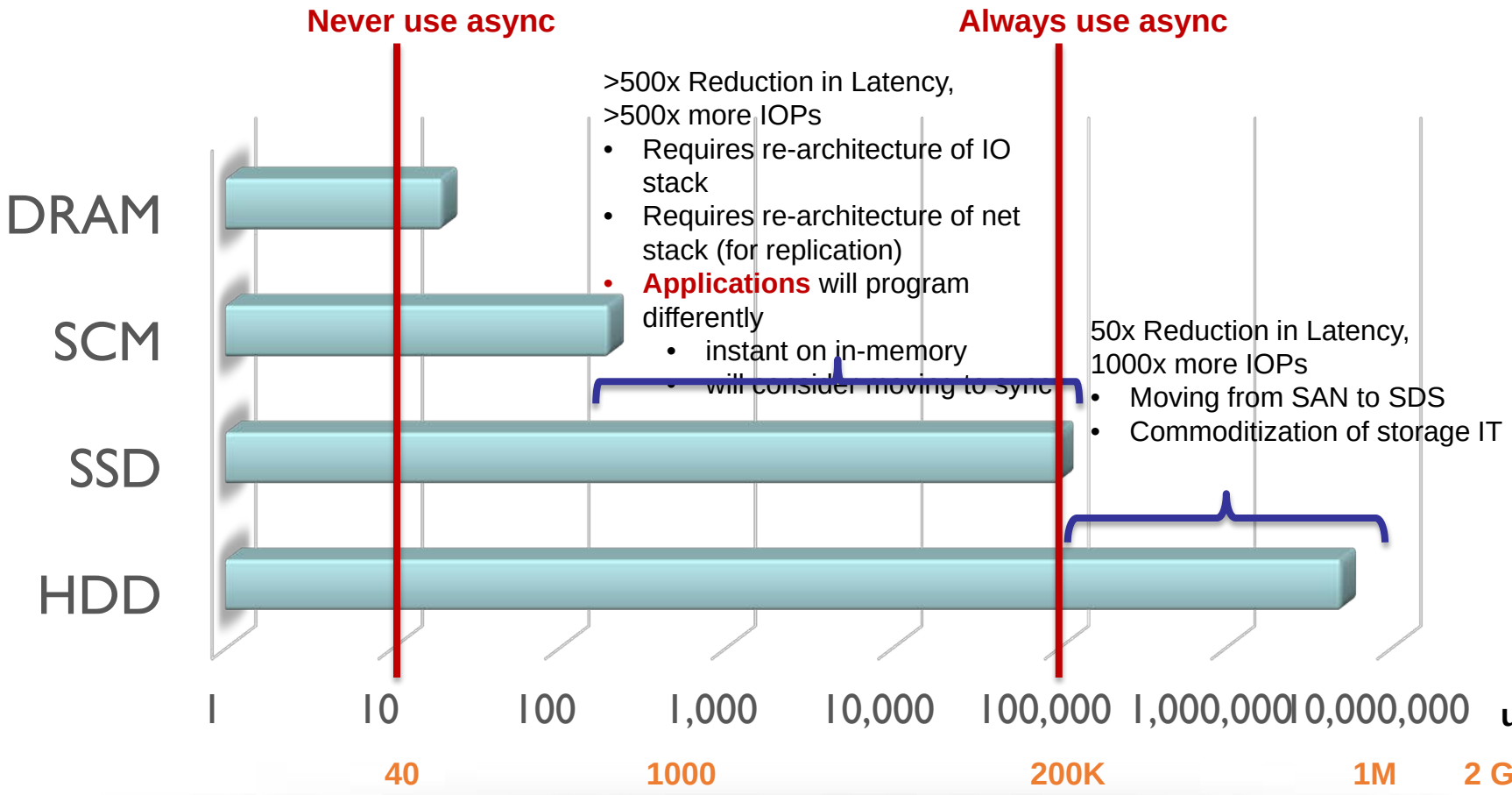
- Explore the "Full-stack" view in this presentation

Low Latency Storage (Review)

- 2000 – HDD latency – SAN arrays accelerated using memory
  - ~5000 usec latency
- 2010 – SSD latency –mere mortals can configure high perf storage
  - ~100 usec latency (50x improvement)
- 2016 – beginning of Storage Class Memory (SCM) revolution
  - <1 usec latency (local), <10 usec latency (remote) – (~100x improvement)
  - Volume deployment imminent (NVDIMM today)

- 5000x change over 15 years!

Storage Latencies and Storage API



Local Filesystems and Local APIs

- DAX
  - Direct Access Filesystem, file or mapped-file API
  - Windows and Linux (very) similar
- NVML
  - NVM Programming Library
  - Open source, included in Linux, future included in Windows
- Specialized interfaces
  - Databases
  - Transactional libraries
  - Language extensions (!)
  - etc

Taking these APIs Remote

- Need a remote durability guarantee (“Commit”)
- At multiple layers:
  - Platform and PM device itself
  - Network transfer
  - Storage protocol
- Let’s walk the stack
  - Bottom-to-top

Consistency and Durability - Platform

RDMA Protocols

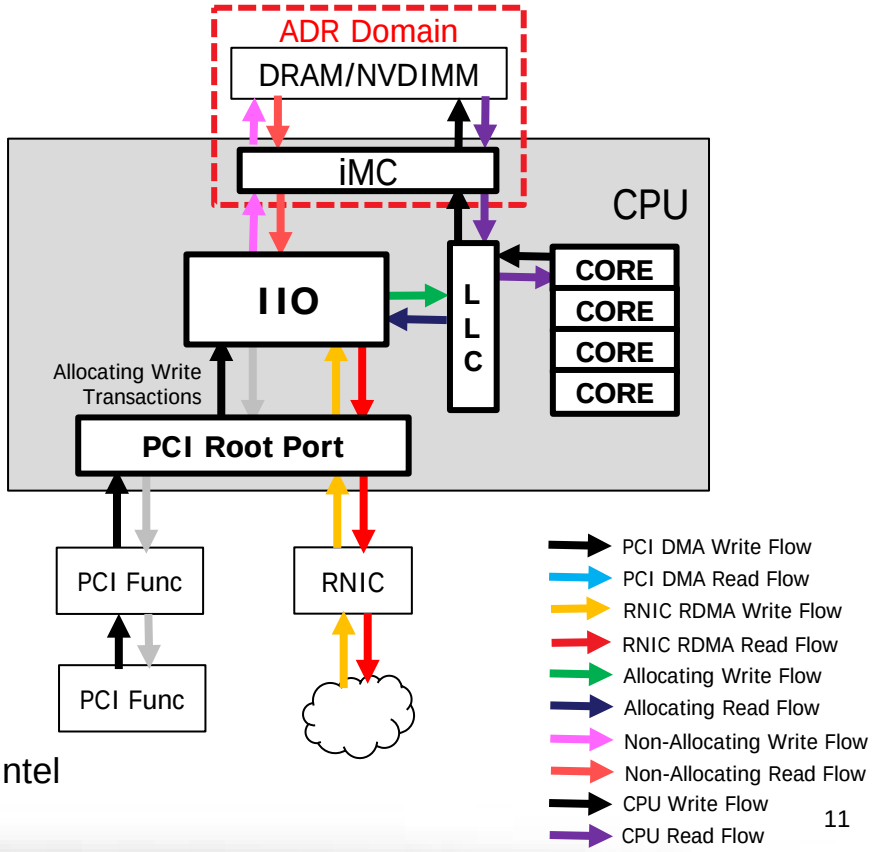
- Need a remote guarantee of Durability
- RDMA Write alone is not sufficient for this semantic
  - Completion at sender does not mean data was placed
    - NOT that it was even sent on the wire, much less received
    - Some RNICs give stronger guarantees, but never that data was stored remotely
  - 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
- An extension is required

PCI Protocol

- PCI support for Commit
  - To Memory, CPU, PCI Root, PM device, PCIe device, …
  - Avoids CPU interaction
  - Supports strong data consistency model
- Perform equivalent of Flush/Commit
  - Without CPU involvement (and latency)
- PCI Extensions take time
  - Workarounds exist (and can be used today)

RDMA with byte-addressable PM – Intel HW Architecture - Background

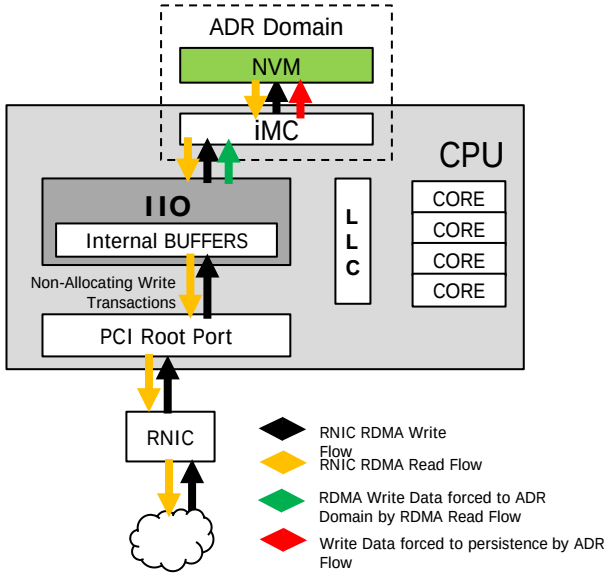
- ADR – Asynchronous DRAM Refresh
  - Allows DRAM contents to be saved to NVDIMM on power loss
  - Requires special hardware with PS or supercap support
  - ADR Domain – All data inside of the domain is protected by ADR and will make it to NVM before power dies. The integrated memory controller (iMC) is currently inside of the ADR Domain.
  - HW does not guarantee the order that cache lines are written to NVM during an ADR event
- IIO – Integrated IO Controller
  - Controls IO flow between PCIe devices and Main Memory
  - "Allocating write transactions"
    - PCI Root Port will utilize write buffers backed by LLC core cache when the target write buffer has WB attribute
    - Data buffers naturally aged out of cache to main memory
  - "Non-Allocating write transactions"
    - PCI Root Port Write transactions utilize buffers not backed by cache
    - Forces write data to move to the iMC without cache delay
  - Various Enable/Disable methods, non-default
- DDIO – Data Direct IO
  - Allows Bus Mastering PCI & RDMA IO to move data directly in/out of LLC Core Caches
  - Allocating Write transactions will utilize DDIO

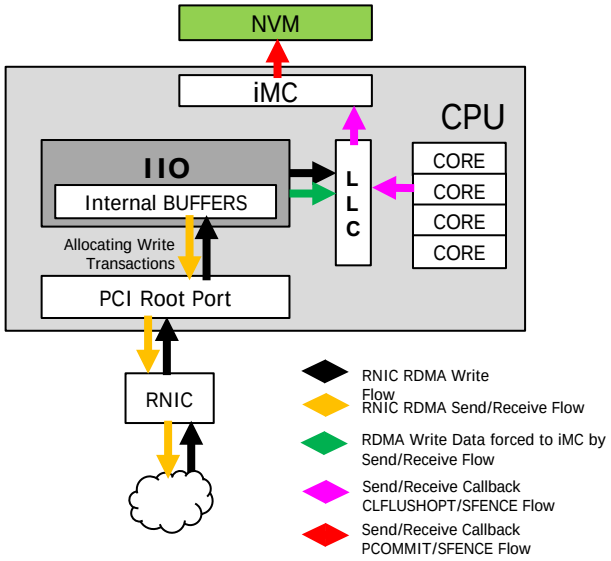


Credit: Intel

Platform Durability Workarounds

- Alternatives proposed – also see SDC 2015
- Significant performance (latency) implications, however!
- **Can we employ existing protocols, e.g. SMB3?**



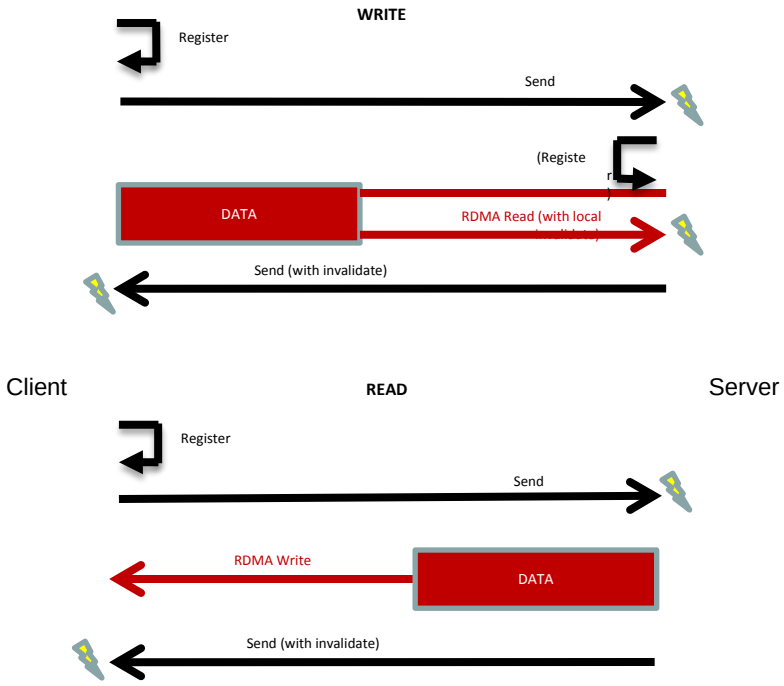


Credit: Intel

SMB3 Push Mode

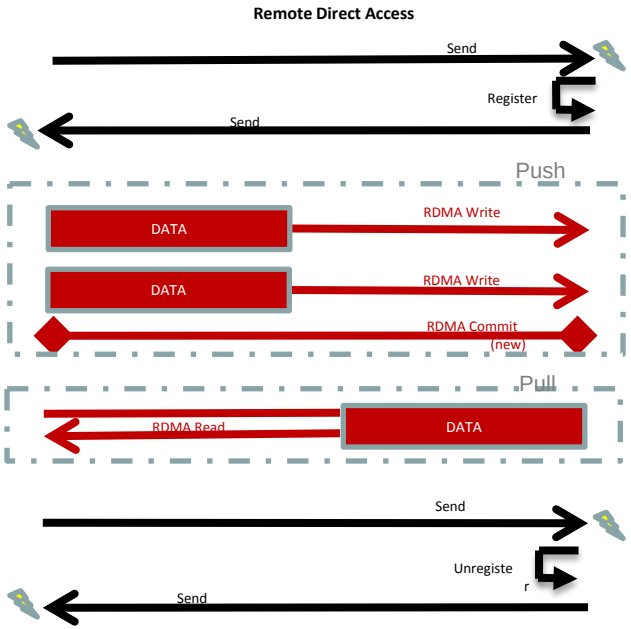
SMB3 Flush - Traditional Commit

- Basic steps:
  - Open file, obtain handle and lease
  - Write file with SMB2_WRITE (or read with SMB2_READ)
    - Buffered or writethrough
    - Server process performs load/store operations to PMEM file
  - Flush file with SMB2_FLUSH
    - Performs Commit when share is DAX-enabled
  - Handle possible recalls
  - Close file
- Advantages:
  - Traditional API, fully functional
- Disadvantages:
  - Existing latencies



SMB3 Push Mode – With Explicit Commit

- 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
- Details follow



SMB3 Push Mode - Open

- Opening file performs “usual” processing
  - Authorization, handle setup, etc
  - New SMB2 Create Context (perhaps) signals desired Push Mode
  - Sent in conjunction with requesting a lease
    - If DAX filesystem and lease, establishes mapped-file r/w interface to server
    - Returns the create context to indicate push mode is available
- Obtaining a lease performs “usual” processing plus
  - The lease provides a means to manage mapping
  - If mapping changes, or if later registration revokes, lease recall is performed
- DAX mode may require certain “traditional” client steps
  - Issued via SMB3
  - E.g. extending file to allocate blocks / establish mapping
- Otherwise, nothing unusual

SMB3 Push Mode – Registration

- Need to register file region(s) for RDMA Write from client
- Client issues new FSCTL
  - MAP_REMOTE_REGION(offset, length, mode r/w)
  - Server pins mapping, RDMA registers pages, returns RDMA region handle
- Client can request multiple regions
  - And multiple r/w modes
- Client can remotely write or read

SMB3 Push Mode – Operation

- Client performs remote writes and reads directly to and from file
  - Entirely within RDMA layers, no server processing at all!
- If RDMA Commit operation available:
  - Client commits writes via RDMA
- Otherwise:
  - Client commits with SMB2_FLUSH
- Client may periodically update file metadata timestamps, etc
  - Using SMB2_IOCTL(FILE_BASIC_INFORMATION), etc
- Note – Push Mode cannot:
  - Add blocks / append file
  - Punch holes, etc
  - These must be done with traditional SMB3 operations

SMB3 Push Mode - Recall

- Server must manage sharing and registration
- Client ownership of a lease covers both
- Recalled upon:
  - Sharing violation (as usual)
  - Mapping change (new DAX callback)
  - Registration change (RDMA resource behavior)
    - Managed by Server itself
- When recalled, client performs the usual:
  - Flush any dirty buffers (and possibly commit)
  - Return the lease
  - Possibly re-obtain a new lease and re-register for push mode
    - If not, fall back to “normal” SMB3 operations

SMB3 Push Mode – Done

- When done, “usual” processing
  - Return lease / close handle
  - Returning the lease and/or closing handle clears push mode registration
    - Or also… an explicit fsctl? TBD
- Push Mode registration is reset during handle recovery
  - Must be re-obtained after handle recovered (like leases)
  - New mapping, new RDMA handle
    - If issued early, RDMA operations would fail and kill connection

RDMA Durability – Protocol Extension

“Doing it right” - RDMA protocols

- Need a remote guarantee of Durability
- Highly efficient (lowest latency)
- Standardized across all RDMA protocols

RDMA Commit (concept)

- RDMA Commit
  - New wire operation, and new verb
  - 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

- Under discussion:
  - 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 implemented in ordered operations
- These will be driven by workload (application) requirements, e.g.:
  - Small log-write scenario will always commit
  - Bulk data movement will prefer batching

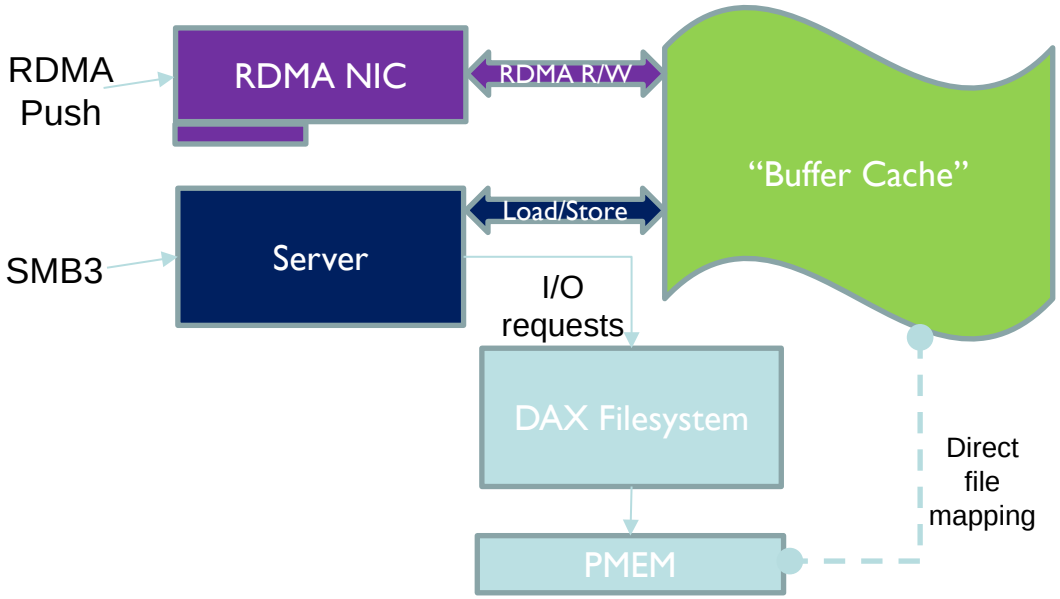
Latencies (expectations)

- Single-digit microsecond remote Write+Commit
  - Push mode minimal write latencies <<10us (2-3us plus data wire time)
  - Commit time is platform-managed and platform+payload dependent
  - Note, this is order-of-magnitude improvement over today's transfer (30-50us)
- Remote Read also possible
  - Roughly same latency as write, but without commit
- No server interrupt
  - Zero server CPU overhead or added latency
  - 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

Push Mode Considerations

SMB3 Push Mode – “Buffering”

- Buffered mode handles have a new meaning:
  - Buffer cache page is mapped from PMEM
    - Zero copy load/store
- All push mode handles opened in “buffered” mode
  - Or per-I/O buffered mode forced on
- Enables direct mapping and direct RDMA reading/writing of PMEM-resident file
  - In future, also allows RDMA Commit



SMB3 Push Mode – Recalls

- Push mode management requires new server plumbing
- Registration recalls may be needed for:
  - Sharing violations/file changes (as usual)
  - DAX filesystem events (file relocation, extension, etc)
  - RDMA resource constraints
  - Any other server matter
- Lease recall is mechanism for all
- ☞ Server “upcalls” may originate within and without filesystem

SMB3 Push Mode – Congestion

- Push Mode fan-in from many (1000's+) clients can congest interface
- And also congest the PMEM
- RDMA itself does not provide congestion control
- But, SMB3 does
  - With credits, and also Storage QoS
  - Existing client-side behavior can mitigate
- RNICs can also provide QoS control
- More thinking needed here

SMB3 Push Mode – Integrity

- Data-in-flight protection and data-at-rest integrity validation?
  - In SMB3, provided by signing and/or encryption, and the backend file store
  - Push mode transfers all data via RDMA, “below” SMB3
- RDMA protocols do not currently support signing or encryption
  - Specifications refer to IPsec or other lower-layer facility
  - Extensions to be discussed in RDMA standards areas
- Remote integrity also not available
  - Remote server CPU not at all involved in transfers
  - Extensions also being discussed for RDMA remote integrity validation

Windows Implementation

Windows Server 2016

- Tech Preview 5 available since Summer
- General Availability is “imminent”
- Supports DAX, and remote SMB access to DAX
  - With all the expected performance advantages
  - Does *not* implement Push Mode
  - No reliance on extending the above protocols
- We considered “Direct Mapping” mode (slide 27)
- With surprising results:

SCM Write Latencies – Traditional vs Direct
(write-only, no push mode)

diskspd.exe -b64K -c20M -d30 -L -o1 -si4k -t8 -w100 -h –fr <remote SCM file> [40Gb link]

RDMA traditional “unbuffered”
(RDMA to bounce pages, write to SCM file)

Write IO

thread	bytes	I/Os	MB/s	I/O per s	AvgLat	LatStdDev
total:	131612147712	2008242	4182.43	66918.86	0.119	0.025

%-ile	Read (ms)	Write (ms)	Total (ms)
min	N/A	0.068	0.068
25th	N/A	0.108	0.108
50th	N/A	0.117	0.117
75th	N/A	0.126	0.126
90th	N/A	0.137	0.137
95th	N/A	0.144	0.144
99th	N/A	0.165	0.165
3-nines	N/A	0.306	0.306
4-nines	N/A	0.548	0.548
5-nines	N/A	2.776	2.776
6-nines	N/A	4.030	4.030
7-nines	N/A	4.962	4.962
8-nines	N/A	4.962	4.962
max	N/A	4.962	4.962

RDMA direct “buffered”
(Map SCM file pages, RDMA directly to pages, release mappings)

Write IO

thread	bytes	I/Os	MB/s	I/O per s	AvgLat	LatStdDev
total:	132543741952	2022457	4212.76	67404.19	0.118	0.027

%-ile	Read (ms)	Write (ms)	Total (ms)
min	N/A	0.068	0.068
25th	N/A	0.107	0.107
50th	N/A	0.116	0.116
75th	N/A	0.126	0.126
90th	N/A	0.136	0.136
95th	N/A	0.143	0.143
99th	N/A	0.163	0.163
3-nines	N/A	0.306	0.306
4-nines	N/A	0.676	0.676
5-nines	N/A	2.724	2.724
6-nines	N/A	6.639	6.639
7-nines	N/A	6.868	6.868
8-nines	N/A	6.868	6.868
max	N/A	6.868	6.868

- **Latencies differ less than 1%**
- Reasons are understood, and go deep
- DAX delivers, but
- ☞ No advantage for “buffered” DAX mapping (yet)

No, We Are Not Disappointed

- Results indicate Windows SMB stack already efficient
- Many new insights into Memory Manager and Cache Manager
- Stability and performance are maintained
- Improve in future update
  - Without protocol impact or dependency
  - After thorough testing

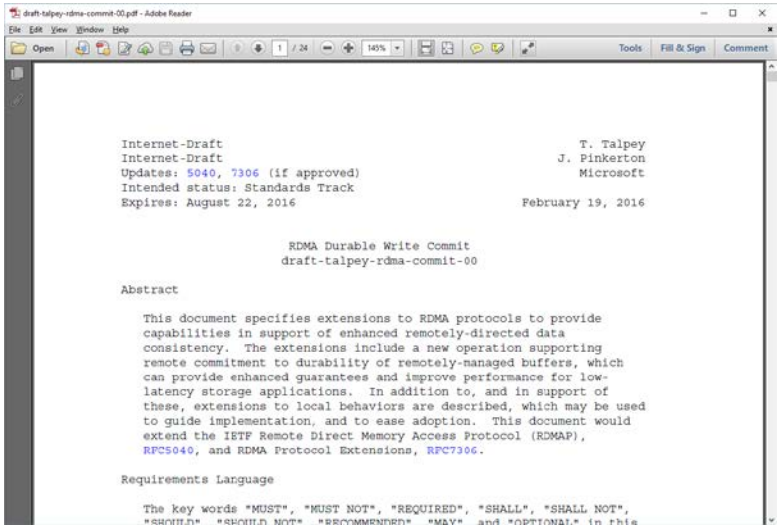
With Windows Server 2016

- Install your favorite SCM solution
  - E.g. NVDIMM
- Install your favorite RDMA NIC
  - All types available to choose from
- Configure a DAX filesystem (*)
- Create an SMB3 share
- Use and Enjoy

* “**format M: /dax**” https://technet.microsoft.com/en-us/windows-server-docs/storage/commands/format

External RDMA Efforts

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



Internet-Draft
Internet-Draft
Updates: 5040, 7306 (if approved)
Intended status: Standards Track
Expires: August 22, 2016

T. Talpey
J. Pinkerton
Microsoft

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/

Resources

- SNIA NVM Programming TWG:
  - http://www.snia.org/forums/sssi/nvmp
- NVM Summit 2016:
  - http://www.snia.org/nvmsummit2016
  - DAX:
    - http://www.snia.org/sites/default/files/NVM/2016/presentations/Neal%20Christiansen_SCM_in_Windows_NVM_Summit.pdf
    - http://www.snia.org/sites/default/files/NVM/2016/presentations/JeffMoyer_Persistent-Memory-in-Linux.pdf
- Open Fabrics Workshop:
  - https://openfabrics.org/index.php/2016-ofa-workshop-presentations.html
- SDC 2015:
  - http://www.snia.org/events/storage-developer/presentations15