SDC
STORAGE DEVELOPER CONFERENCE
SNIA SANTA CLARA, 2015

RDMA with PMEM

Software mechanisms for enabling access to remote persistent memory

Chet Douglas
Intel Corporation

RDMA with PMEM - Agenda

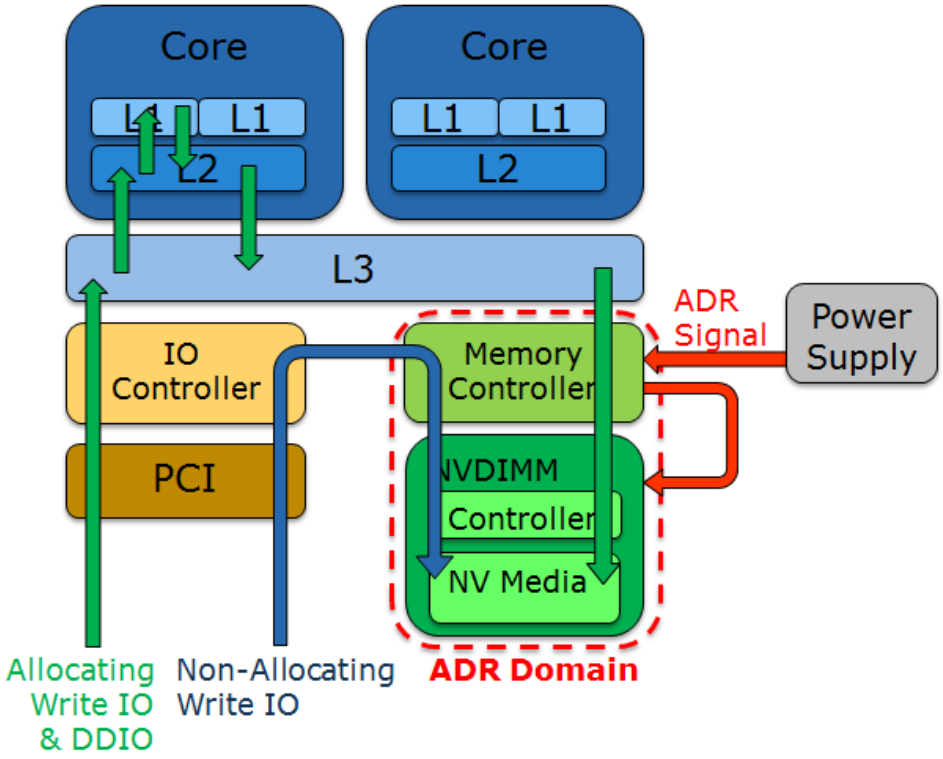
- Problem statement
- Platform HW Background
- Visibility vs Durability
- Local Node Data Durability Considerations
- Remote Node Data Durability Consideration
- “Appliance Method” for Remote Data Durability
- “General Purpose Server Method” for Remote Data Durability
- Modeling SW Latencies for Remote Data Durability
- Performance Tradeoffs & Durability Mechanism Pros/Cons
- Future Considerations
- Takeaways

RDMA with PMEM – Problem Statement

- Current RDMA HW and SW solutions do not take persistent memory (pmem) in to account and do not have an explicit mechanism for forcing RDMA Writes to remote pmem to be durable (power fail atomic)
- HA and HPC pmem Use Cases require writes to remote memory between compute nodes utilizing RDMA to support remote data base log updates, remote data replication, etc
- In the short term, SW only modifications are possible to make RDMA SW Applications persistent memory aware and to force written data to be persistent (this slide deck)
- In the longer term, industry wide enabling through standards organizations is required to have specific vendor agnostic mechanisms for utilizing persistent memory with RDMA, requiring HW and SW changes (See Tom Talpey's session)

RDMA with PMEM – HW Background

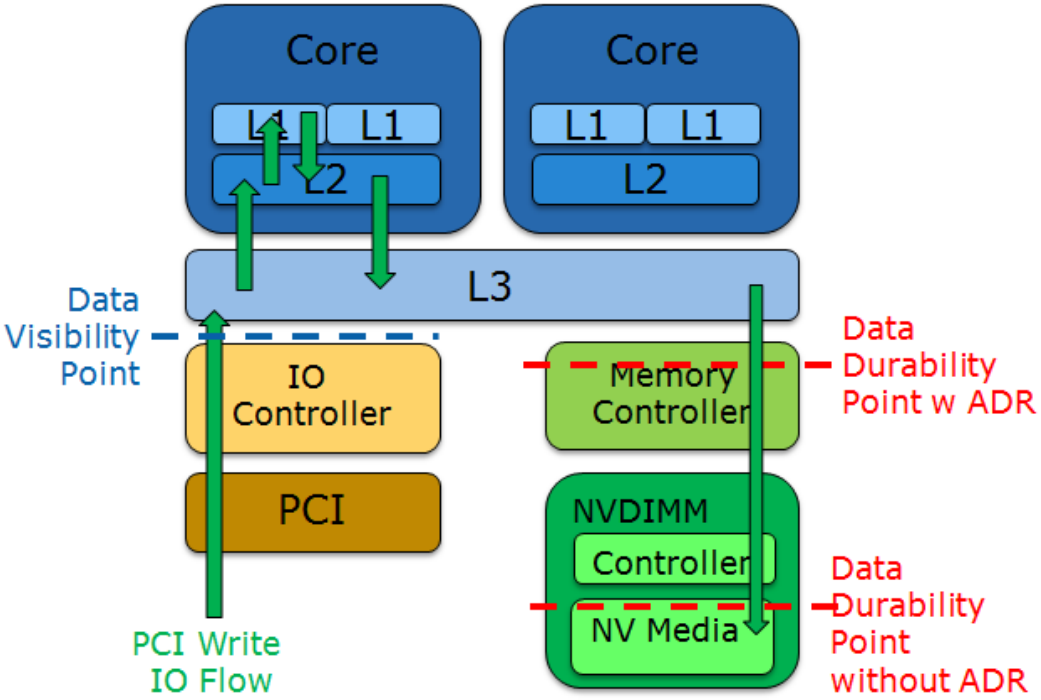
- **ADR - Asynchronous DRAM Refresh**
  - Allows NVDIMMs to save DRAM contents to flash on power loss
  - ADR Domain – All data inside of the domain is protected by ADR and will make it to pmem before supercap power dies. The integrated memory controller (iMC) is currently inside of the ADR Domain.
- **IO Controller**
  - Controls IO flow between PCIe devices and Main Memory
  - "Allocating write transactions"
    - PCI Root Port will utilize write buffers backed by cache
    - 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
  - Enable/Disable via BIOS setting



- **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

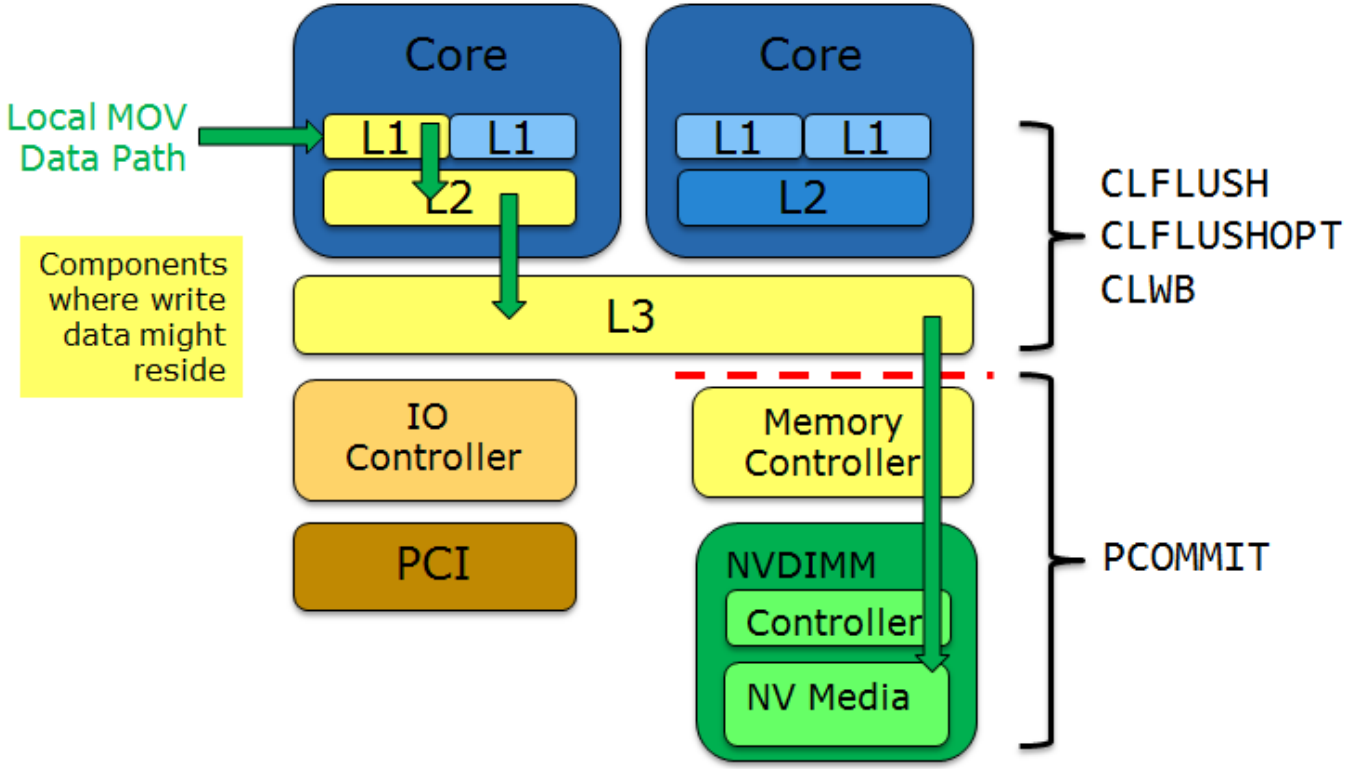
RDMA with PMEM – Visibility vs Durability

- Write Data Visibility Point – Once data makes it out of the IO controller, HW makes the data visible to all caches and cores, independent of DRAM or NVDIMM devices
- Write Data Durability Point – Typically data is not considered durable until it has been successfully written by the Memory Controller to persistent media



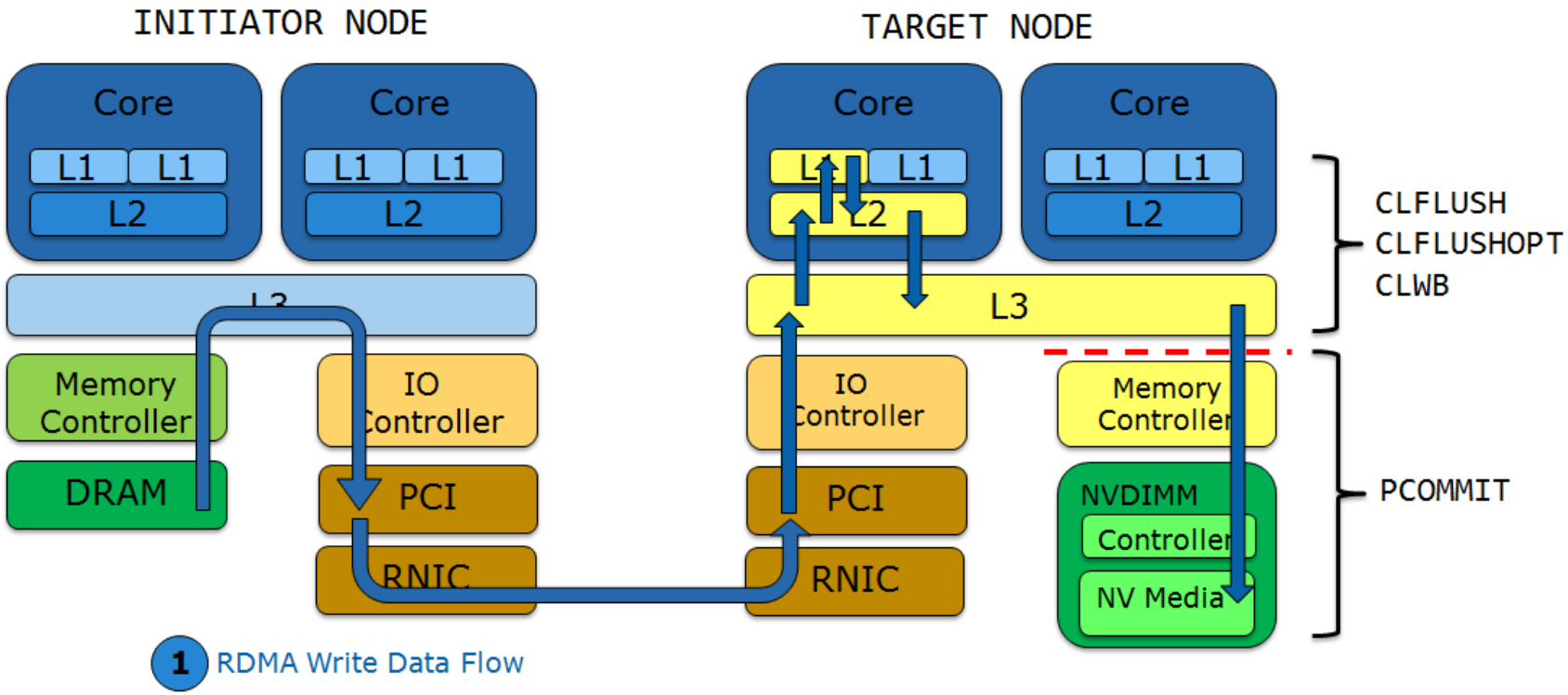
RDMA with PMEM – Local Node Data Durability Considerations

- First, Write Data must be flushed from CPU caches to the Memory Controller and optionally invalidated
- Second, Write Data in the Memory Controller must be flushed to pmem



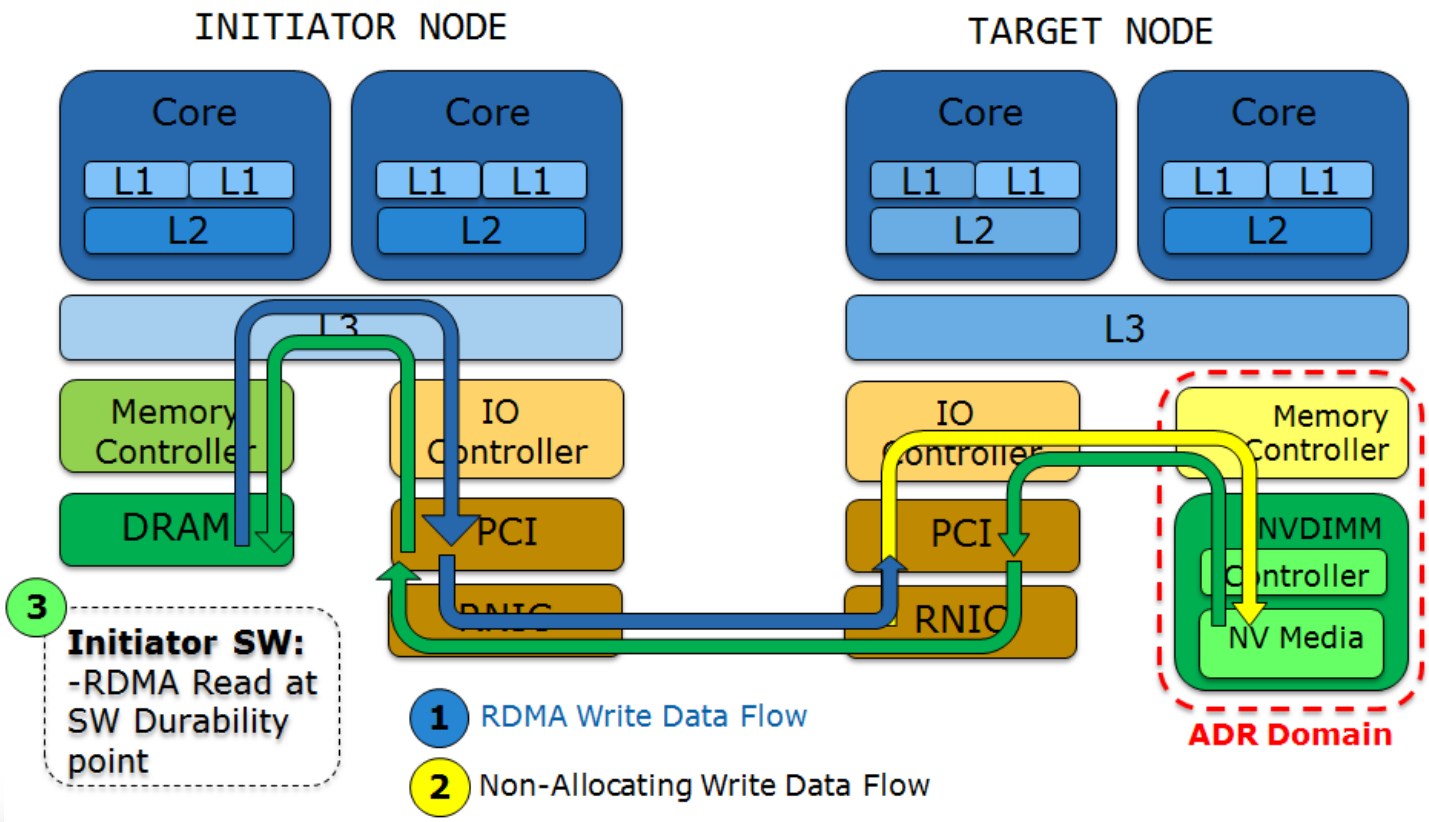
RDMA with PMEM – Remote Node Data Durability Considerations

- Same steps to make write data durable on a local node, must be followed on a remote Target node
- Signal to make data durable does not need to come from the Initiator node



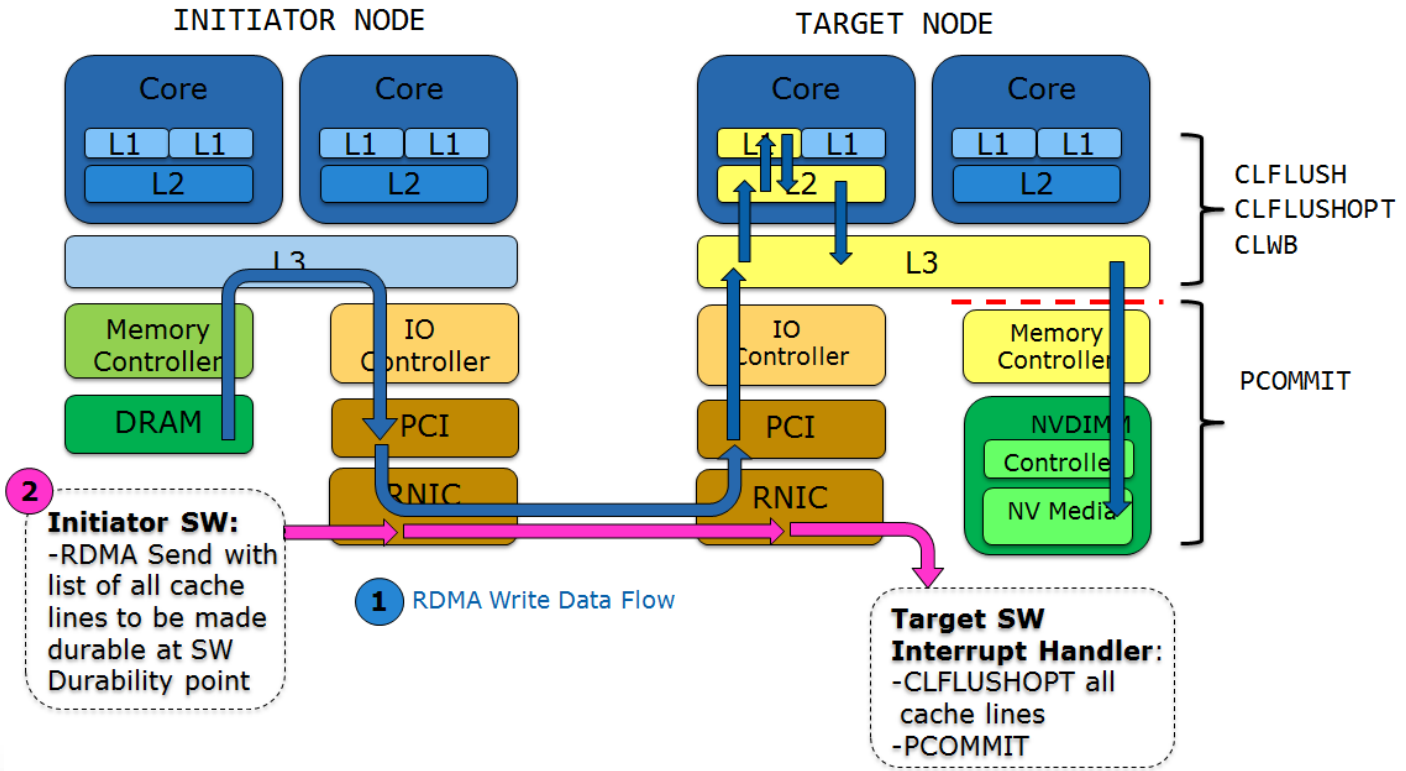
RDMA with PMEM – "Appliance" Durability Mechanism

- Use Non-Allocating Writes for the remote Target node RNIC to bypass CPU caches – Target node does not need to issue CLFLUSHOPTs
- Follow a number of RDMA Writes with an RDMA Read at the point SW wishes to make data durable – Flushes PCI and Memory Controller pipelines
- Utilize ADR so Target node does not need to issue PCOMMIT



RDMA with PMEM – "General Purpose Server" Durability Mechanism

- Use default platform configuration & allocating write flows
- Initiator SW issues RDMA Send at SW Durability Point and passes list of cache lines to make durable
- Target SW Interrupt Handler is interrupted and issues CLFLUSHOPT for each cache line in the list to push data in to the Memory Controller followed by PCOMMIT to force write data to pmem media



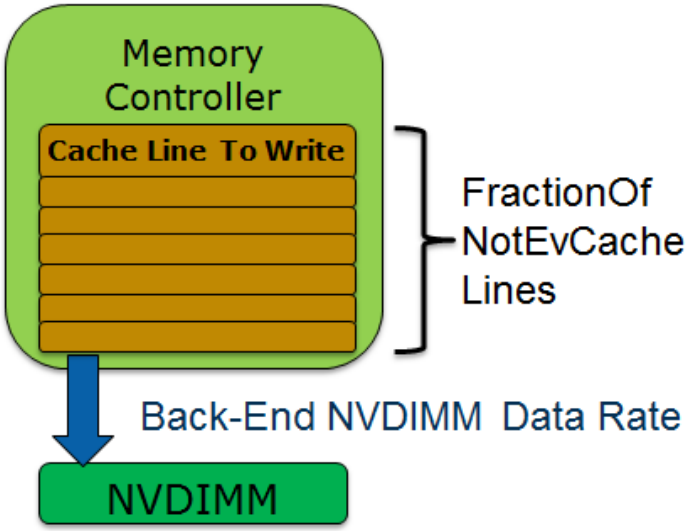
RDMA with PMEM – Modeling SW Latencies for Remote Durability Mechanisms

- CLFLUSHOPT/SFENCE Model
- Basic SW latency model for determining the order of magnitude for flushing write data from CPU caches to the Memory Controller and waiting for flushes to complete
- Not trying to be cycle accurate for a non-blocking "CLFLUSHOPT" instruction which is not that interesting at this level
- For a single cache line flush, use CLFLUSH/SFENCE as a proxy
- For multiple cache lines to flush, use full line non-temporal (NT) stores as a proxy
  - NT stores flush each cache line but don't have the serializing behavior of CLFLUSH
  - _mm_stream_si128() streaming execution time can vary depending on CPU preparation and cache preloading
  - Measure time moving the entire data transfer amount so that variance in first cache line move are averaged out

Bytes to flush	NTMOV Time (uSec)
64 B	0.11
128 B	0.13
256 B	0.19
512 B	0.29
1024 B	0.53
2048 B	0.97
4096 B	1.86
8192 B	3.88
16384 B	7.91
32768 B	15.21
65536 B	30.80
131072 B	61.25
262144 B	122.26

RDMA with PMEM – Modeling SW Latencies for Remote Durability Mechanisms

- **PCOMMIT/SFENCE Model**
- Basic SW latency model for determining the order of magnitude for committing write data from the iMC to pmem
- Not trying to be cycle accurate for a non-blocking “PCOMMIT” instruction which is not that interesting at this level
- To bound the latency calculation the calculation can be bounded with a fraction of cache lines that have not been evicted by the time SW gets to committing the write data
  - Determine the amount of write data to be committed based on FractionOfNotEvCacheLines
  - Use the backend NVDIMM BW to calculate a basic latency based on write data not already evicted



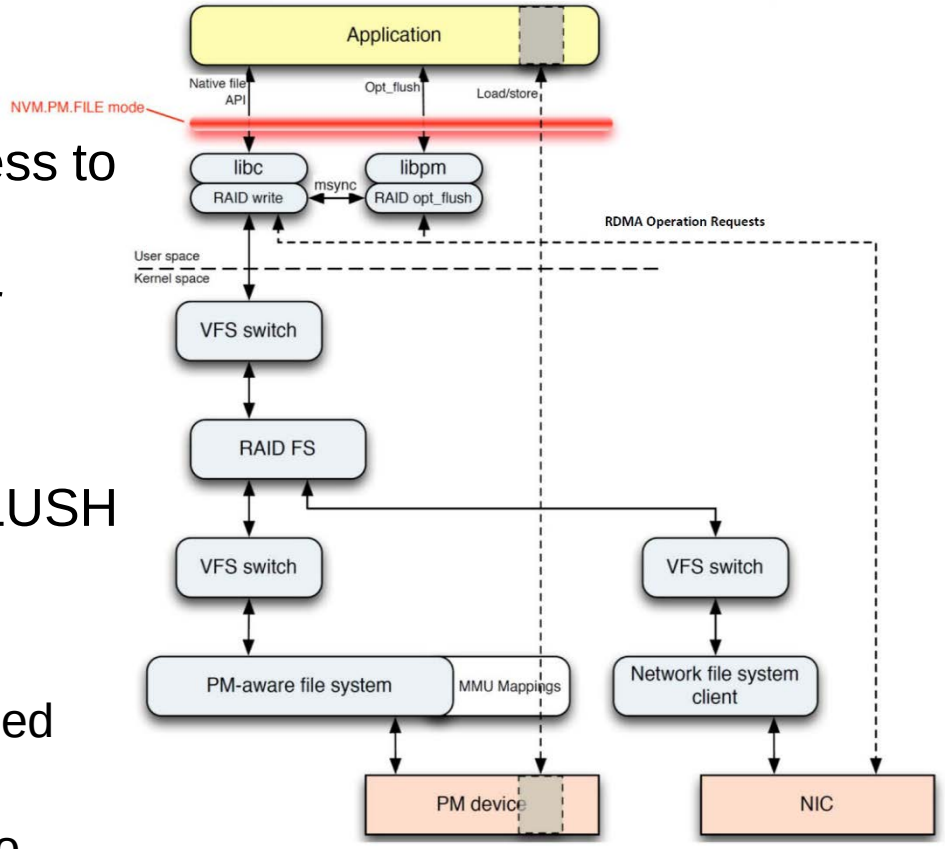
Modelling true Memory Controller queue depth is difficult since write data is naturally aging out of cache & actively being evicted from cache from many different threads of execution

RDMA with PMEM – Performance & Durability Mechanism Pros/Cons

Config & Durability Mechanism	BIOS Changes	Initiator RDMA SW Changes	Target RDMA SW Changes	PROs	CONs	Measured & Modelled Durability Latency 4096B Write Data
-ADR Domain -No DDIO **"Appliance Method"**	Enable Non-Allocating Writes on platform	-RDMA Read after RDMA Writes to force client data in to ADR Domain	None	-Minimal SW changes -No callback on server CPU	-Requires ability to enable Non-Allocating Writes on platform -Platform performance implications when using Non-Allocating Writes -Additional latency of RDMA Read	RDMA Read: **5→8 uSec**
-No ADR Domain -DDIO **"General Purpose Server Method"**	None	-RDMA Send after RDMA Writes to pass list of cache lines to make persistent	-RDMA Send Interrupt Handler issues CLFLUSHOPT for each cache line followed by SFENCE & PCOMMIT, SFENCE	-DDIO can be utilized -No special platform configuration -Most RDMA Apps use Send/Receive for internal book keeping already	-Requires server side callback -Additional latency of RDMA Send/Receive -Additional latency of explicitly flushing all write data	RDMA Send: **9→11 uSec** CLFLUSHOPT/ SFENCE: **2 uSec** PCOMMIT/SFENCE: **0.34→1.6 uSec** **~50% higher latency than "Appliance Method"**

RDMA with PMEM – Future Considerations

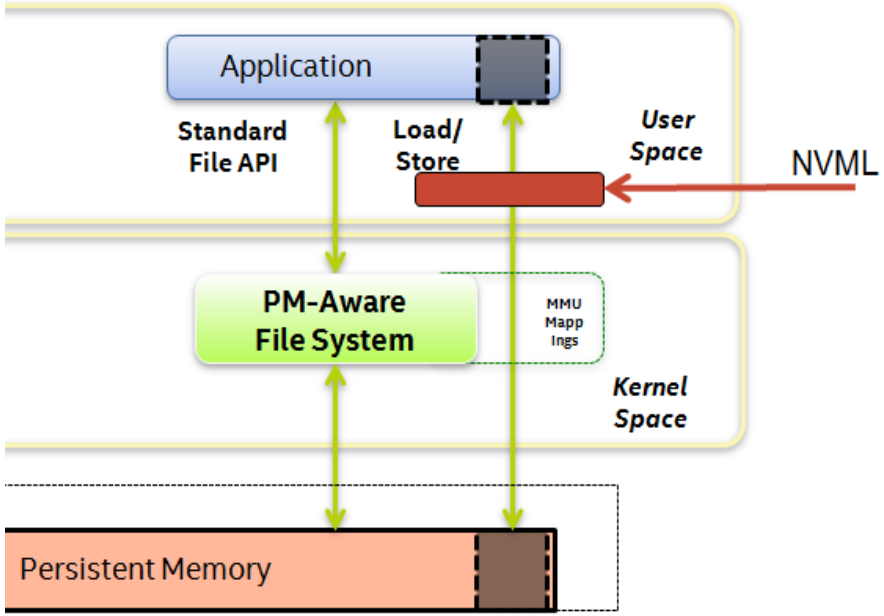
- SNIA *NVM Programming Model*
  - Extend to address remote access to pmem
- SNIA *NVM PM Remote Access for High-Availability* white paper
  - Already contains basic NVM.PM.FILE.OPTIMIZED_FLUSH concepts
  - Add Appendix or Extension
    - Durability mechanisms described here
    - Platform and HW capabilities to detect supported mechanisms



Proposed *NVM Programming Model* updates for NVM.PM.FILE.OPTIMIZED_FLUSH from *NVM PM Remote Access for High-Availability*

RDMA with PMEM – Key Takeaways

- Inserting SW durability points in to existing RDMA based SW is not difficult
- Prepare for future fabric protocol, HW (RNIC, CPU, chipset) & SW extensions to make remote persistent memory programming even easier and improve latency
- Consider abstracting durability mechanisms behind a SW library such as Intel's NVML, OFA libfabrics and libibverbs API
- Get involved in SNIA NVM Programming TWG!
  Bigger changes coming for future extensions to RDMA, PMEM, and data durability



http://www.snia.org/forums/sssi/nvmp