SDC
STORAGE DEVELOPER CONFERENCE
SNIA SANTA CLARA, 2016

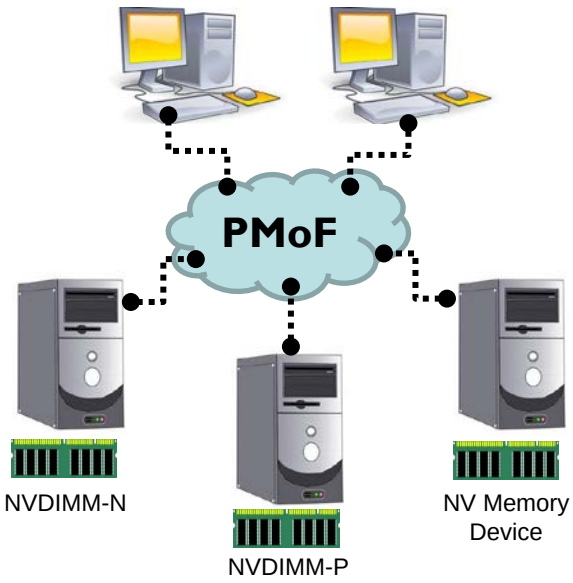
Persistent Memory over Fabric (PMoF)

SW Interface considerations – libfabric case study

Chet Douglas
Intel Corporation

PMoF

- Persistent Memory over Fabric
  - Generic term covering remote node access to persistent memory
  - Many possible non-volatile byte addressable devices are considered in scope for PMoF
  - NVDIMM, NVDIMM-N, NVDIMM-P, etc
- PMoF is fabric agnostic and includes IB, RoCE, iWARP, Omnipath, etc
- PMoF is transport agnostic and includes RDMA Verbs
- Does not include remote access to traditional storage



PMoF - OFA libfabric API Case Study

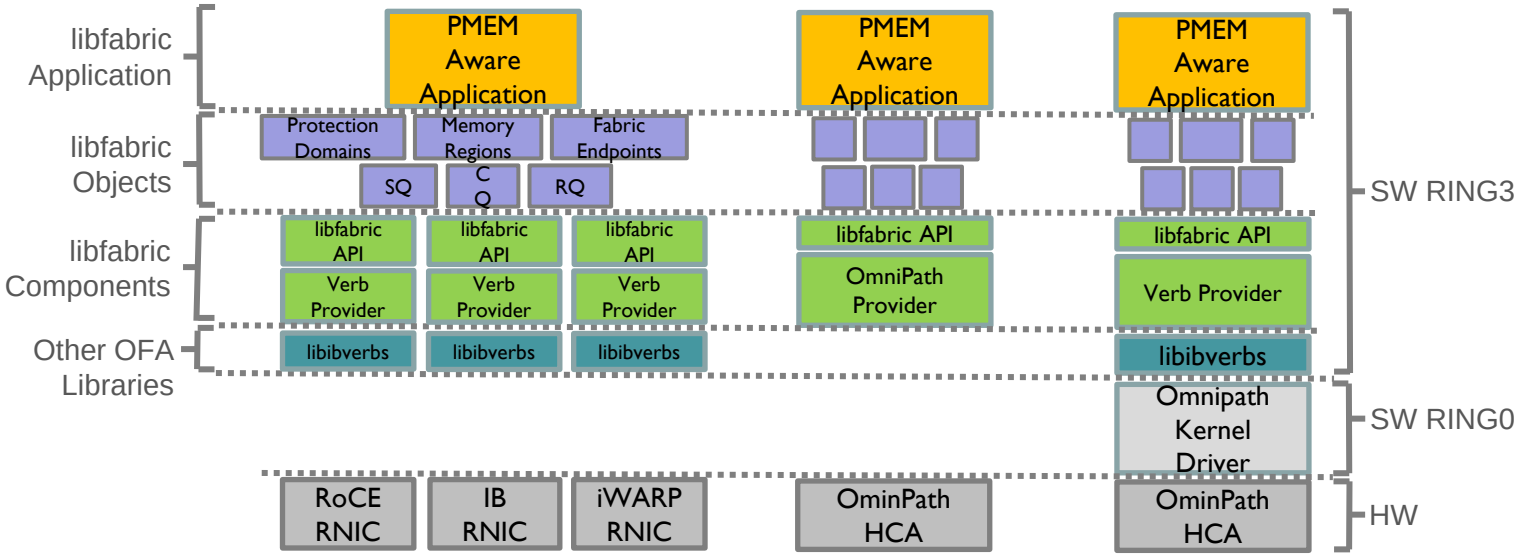
Introduction

- OpenFabrics Alliance (OFA)
  - Open source-based organization that develops, tests, licenses, supports and distributes OpenFabrics Software
- OpenFabrics Interfaces (OFI)
  - *Framework focused on exporting fabric communication services to applications*
  - Developed through OFIWG workgroup
- OFI libfabric library
  - Core component of OFI
  - Ring3 library that defines and exports the user-space API of OFI
  - Typically the only software that applications deal with directly
  - Provider libraries implement the libfabric API and provide fabric specific functionality

PMoF - OFA libfabric API Case Study

Basic Architecture

- libfabric providers implement the API and provide fabric specific functionality
- For RDMA Verbs, a thin Verb provider interfaces the libfabric API to the legacy libibverbs library
- For some fabrics, like Omnipath, which do not implement the RDMA Verbs HW interface:
  - A native Omnipath provider is utilized for lowest latency ring3 access
  - Applications wishing to utilize existing RDMA ordering and fencing semantics (perhaps for application compatibility) can utilize the Verbs provider and the underlying Omnipath kernel driver to provide basic RDMA functionality through SW



PMoF - OFA libfabric API Case Study

Proposed Additions to Support PMoF

- Intel has proposed a set of extensions to the existing OFA libfabric API to add basic native PMoF support
  - **Configuration:** Ability to determine each end points PMoF capabilities
  - **Memory Registration:** Provide PMEM indication when registering memory regions with the end point
  - **Existing Writes:** Simply use the memory key registered with PMEM indication with existing write API
  - **Commit to Durability:** Ability to force specific write data in to the NVDIMMs durability domain
- The motivation for these SW API extensions is to increase bandwidth and obtain the lowest possible latency with the lowest HW design complexity
- These proposed API extensions are primarily driven by a detailed Intel HW assessment of the RDMA IO paths and an understanding of the Intel chipset architecture
- API extensions reflect Intel's current thinking on how we might implement native PMoF support

PMoF - OFA libfabric API Case Study

Proposed Additions to Support PMoF

fi_getinfo

- Retrieve capabilities of each fabric endpoint for a given connection
-
```
int fi_getinfo(int              version,
               const char       *node,
               const char       *service,
               uint64_t         flags,
               struct fi_info   *hints,
               struct fi_info   **info);
```
- Add the following capability bit to the ***info*** field to describe PMoF capabilities of the system:
- **FI_PMEM**
  - Initiator or target node is capable of supporting byte addressable persistent memory (PMoF). At a minimum this assumes the endpoint supports ALL of the following:
- The caller requests specific capabilities to be supported in the ***info*** struct and the endpoint SW will update the same ***info*** struct with the supported capabilities

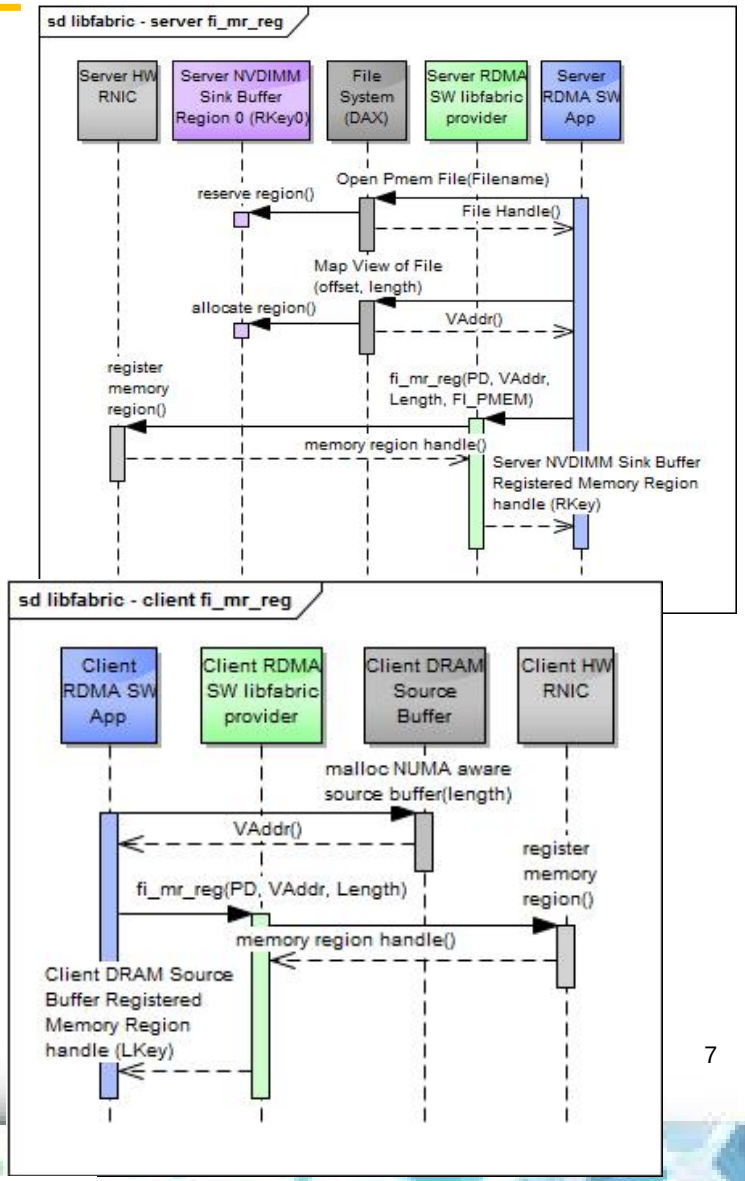
PMoF - OFA libfabric API Case Study

Proposed Additions to Support PMoF

fi_mr_reg

- Register persistent memory data buffer addresses with the fabric endpoint for a specific protection domain with requested accesses and return the opaque Registered Memory Region handle that describes the persistent memory region
-
```
int fi_mr_reg(      struct fid_domain * domain,
                    const void * buf,
                    size_t len,
                    uint64_t access,
                    uint64_t offset,
                    uint64_t requested_key,
                    uint64_t flags,
                    struct fid_mr ** mr,
                    void * context);
```

- Add the following bits to the ***flags*** field to further describe the attributes of the memory region being registered. The access requested is a combination of OR'ing these new access capabilities with existing flags:
- **FI_PMEM** – Memory region being registered is byte addressable persistent memory (PMoF)
- **FI_UNCACHED** - Hint that memory region should not be backed by cache.



PMoF - OFA libfabric API Case Study

Proposed Additions to Support PMoF

fi_writemsg

- The existing fi_writemsg API is extended to support writing to persistent memory
- The existing libfabric mechanism for setting up a CQ is utilized to set up and register an initiator SW completion queue and notification for writes with commit.
- ```
  static inline ssize_t fi_writemsg(struct fid_ep *ep,
                                    const struct fi_msg_rma *msg,
                                    uint64_t flags)
  ```

- ```
  struct fi_msg_rma {
          const struct iovec       *msg_iov;
          void                     **desc;
          size_t                   iov_count;
          fi_addr_t                addr;
          const struct fi_rma_iov  *rma_iov;
          size_t                   rma_iov_count;
          void                     *context;
          uint64_t                 data;
        };
  ```

- The following libfabric ***flags*** are added for handling writes to persistent memory.
- The flags are utilized by the target node endpoint device to precisely control steering of the write data and specifying the completion handling to utilize. Therefore these indicators should be visible in the wire protocol payload and available at the target end point through fabric protocol extensions:

  FI_COMMIT – Commit to pmem all data within scope of the command. Completion to the initiator occurs after all data has been committed to the durable memory domain. Previous fi_write* messages sent to the same Registered Memory Region on the same Connected Endpoint will also be committed to durability before the completion is signaled.

  -With a non-volatile memory region (memory registered with FI_PMEM), completion indicates all write data in scope has reached durability and is power fail safe. Once durability occurs the Initiator RNIC will insert a Completion WQE on the initiators CQ to notify SW.

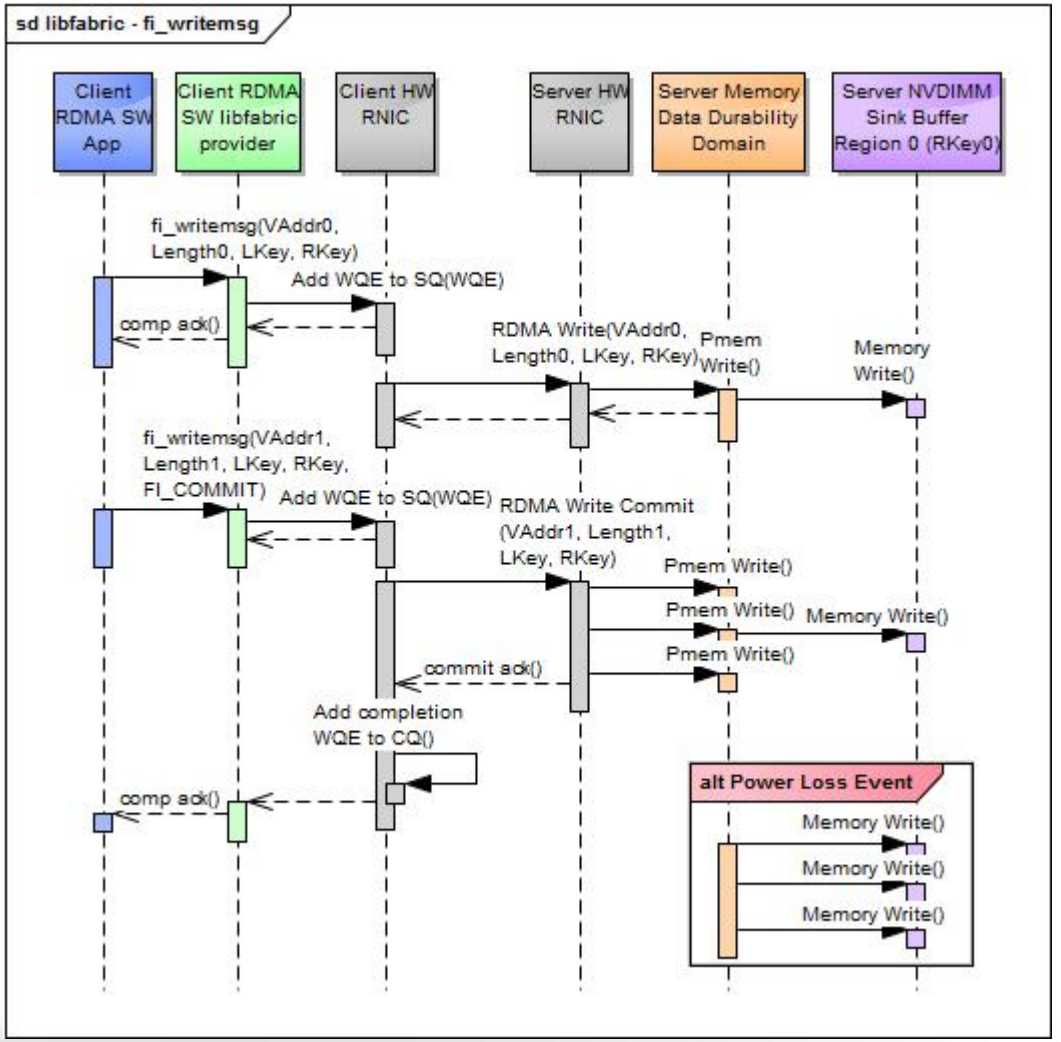
  -With a memory region that is volatile memory (memory registered without FI_PMEM). The completion indicates all write data in scope has reached the global visibility point

PMoF - OFA libfabric API Case Study

Proposed Additions to Support PMoF

fi_writemsg

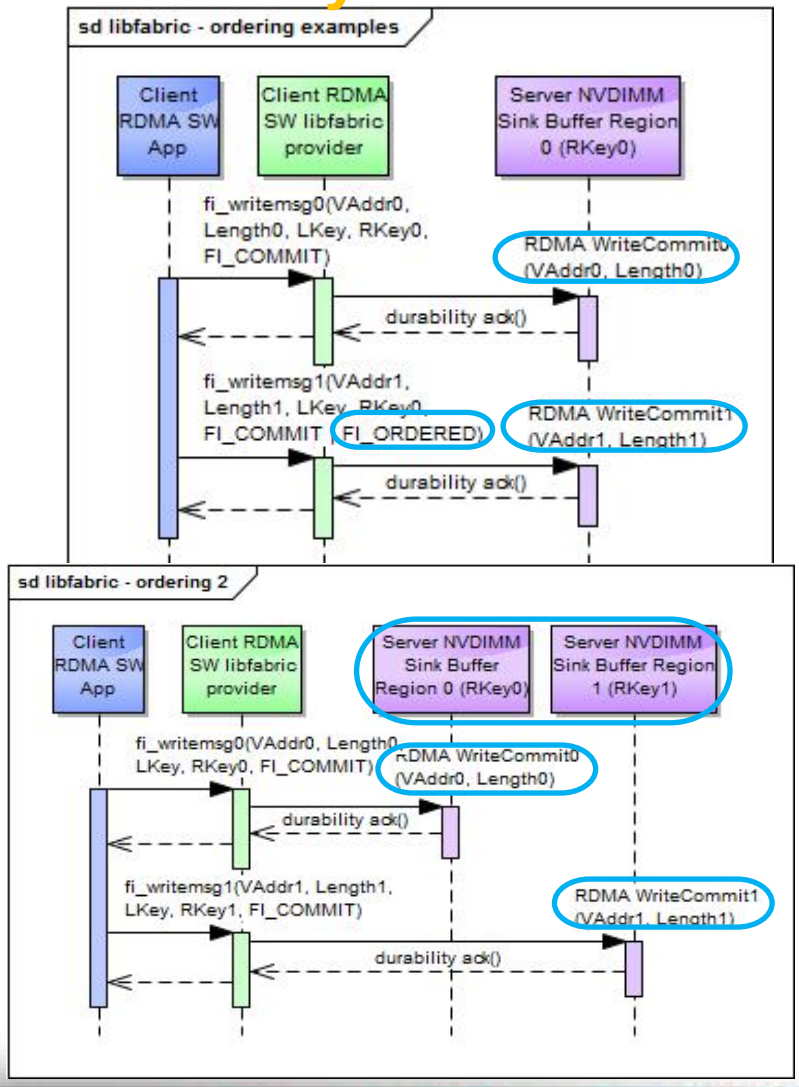
- The following libfabric ***flags*** are added for handling writes to persistent memory
  - **FI_IMMED** – Used in conjunction with FI_COMMIT – Once all write data in scope of the write has reached the pmem durability domain issue a Completion WQE to the target CQ.
  - **FI_ORDERED** – When set with FI_COMMIT, the target endpoint will guarantee that previous writes with the same Registered Memory Region will be made durable *before* executing this write to the same Registered Memory Region.



PMoF - OFA libfabric API Case Study

Enforcing Write Data Ordering to Durability Domain

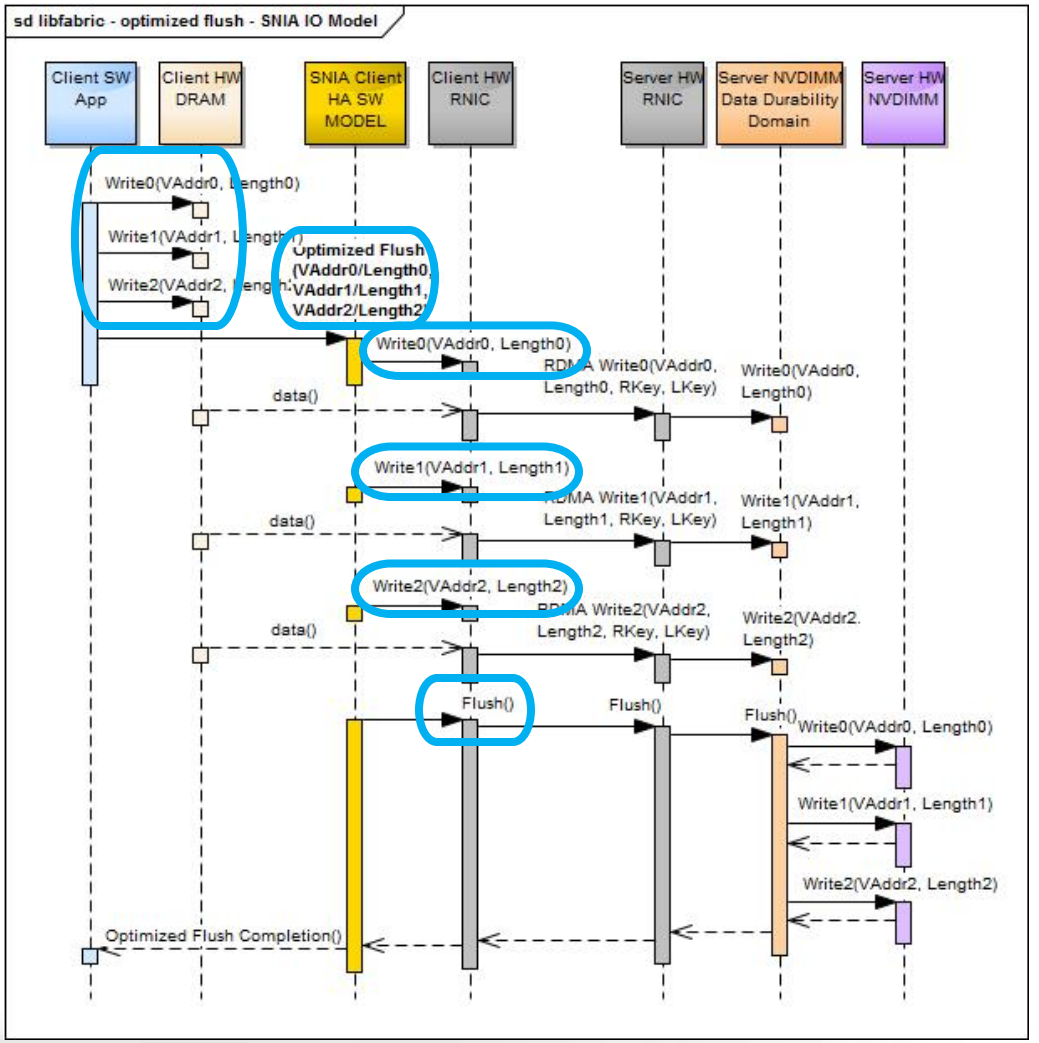
- *Use Case: Multiple writes occurring to the same Server NVDIMM Sink Buffer Region 0 (RKey0):*
  - FI_ORDERED set on fi_writemsg1 required to force fi_writemsg1's write data to be made durable after all previous writes (fi_writemsg0) are made durable
- *Use Case: Multiple writes occurring to different Server NVDIMM Sink Buffer Regions 0 & 1 (RKey0 & RKey1):*
  - Since the writes are to different regions, there is no ordering guarantee on the order in which the write data will be made durable, relative to each other
  - Setting FI_ORDERED on one fi_writemsg will have no affect on the order in which data is made durable relative to the other fi_writemsg



PMoF - OFA libfabric API Case Study

Relating libfabric "Write w Commit" IO Model to the SNIA Optimized Flush IO Model

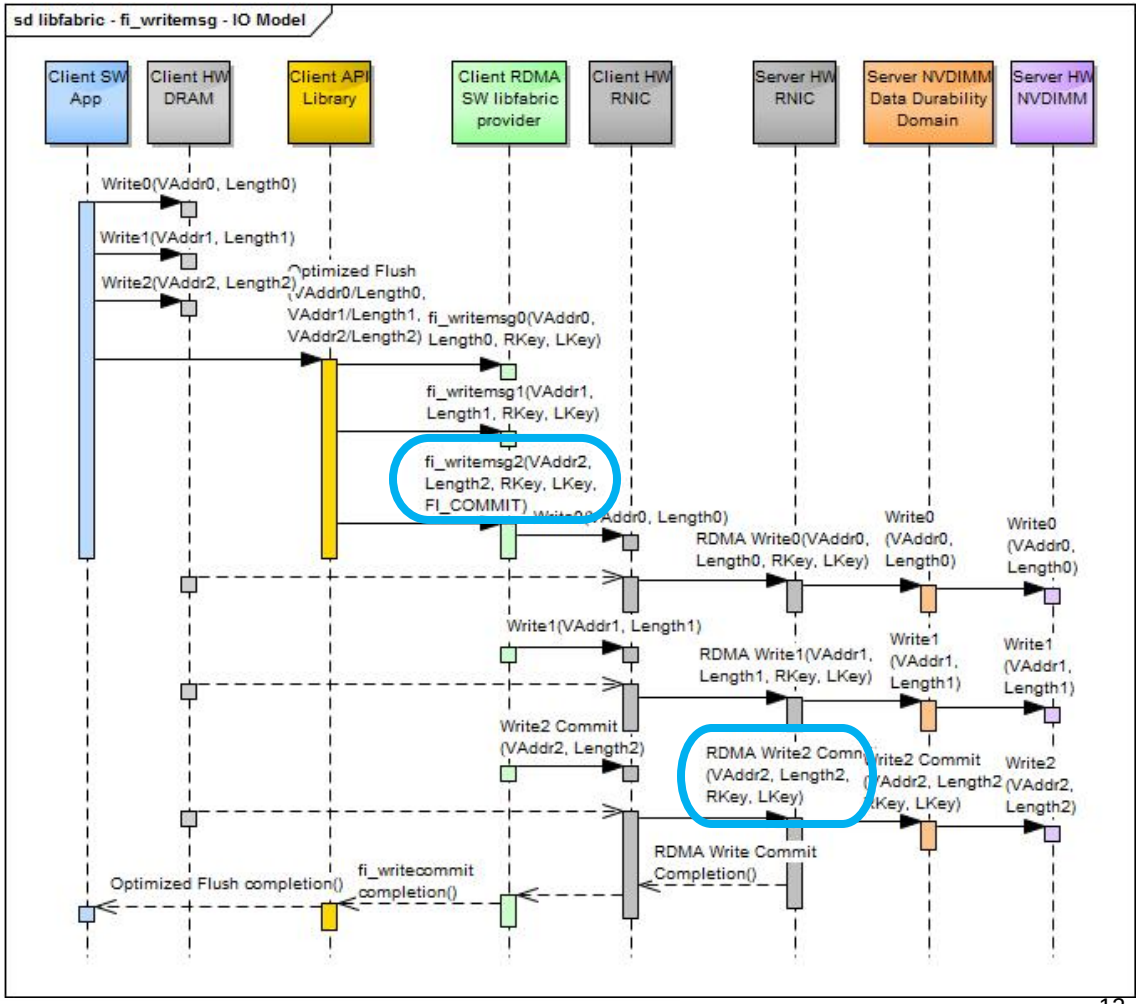
- SNIA HA Optimized Flush
  - Writes to local pmem file are executed prior to Optimized Flush
  - Optimized Flush suggested behavior is a synchronous request that takes multiple pmem regions
  - RDMA Writes to replicate the local writes are implemented internally by the Optimized Flush
  - RDMA Flushing mechanism implemented internally by Optimized Flush



PMoF - OFA libfabric API Case Study

Relating libfabric "Write w Commit" IO Model to the SNIA Optimized Flush IO Model

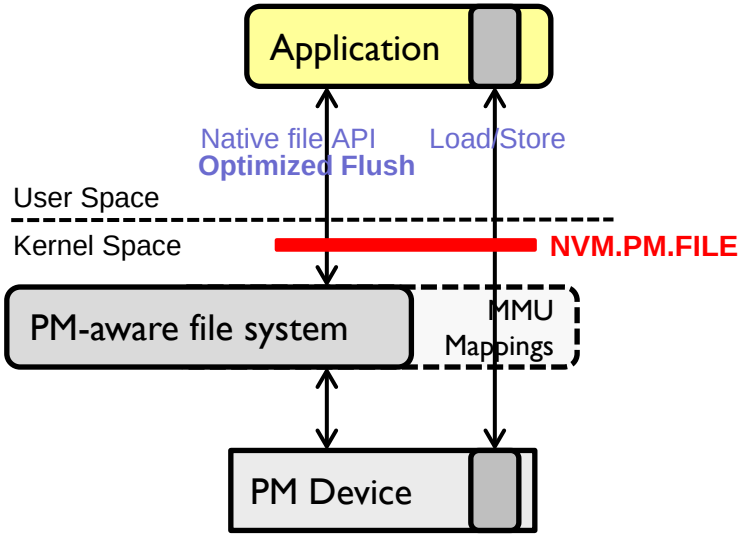
- Libfabric implementation of SNIA HA Optimized Flush
  - Create another API for libfabric or orthogonally another library to implement Optimized Flush
  - RDMA Writes to replicate the local writes are implemented internally by the Optimized Flush
  - RDMA Flushing mechanism implemented internally by Optimized Flush using fi_writemsg with FI_COMMIT set
  - **fi_writemsg with FI_COMMIT can easily be modelled using SNIA Optimized Flush with a change to scoping of the Optimized Flush API**



PMoF - OFA libfabric API Case Study

Relating libfabric "Write w Commit" scope to the SNIA Programming Model

- SNIA NVM.PM.FILE Model
  - Provides means for user space applications to directly access NVM
  - NVM exposed via persistent memory backed Files utilizing existing file system API
  - **NVM.PM.FILE.MAP** – Add a subset of a PM file to application's address space for load/store access.
  - **NVM.PM.FILE.SYNC & NVM.PM.FILE.OPTIMIZED_FLUSH**
    - Synchronize persistent memory content to assure durability and enable recovery by forcing data into the persistence domain

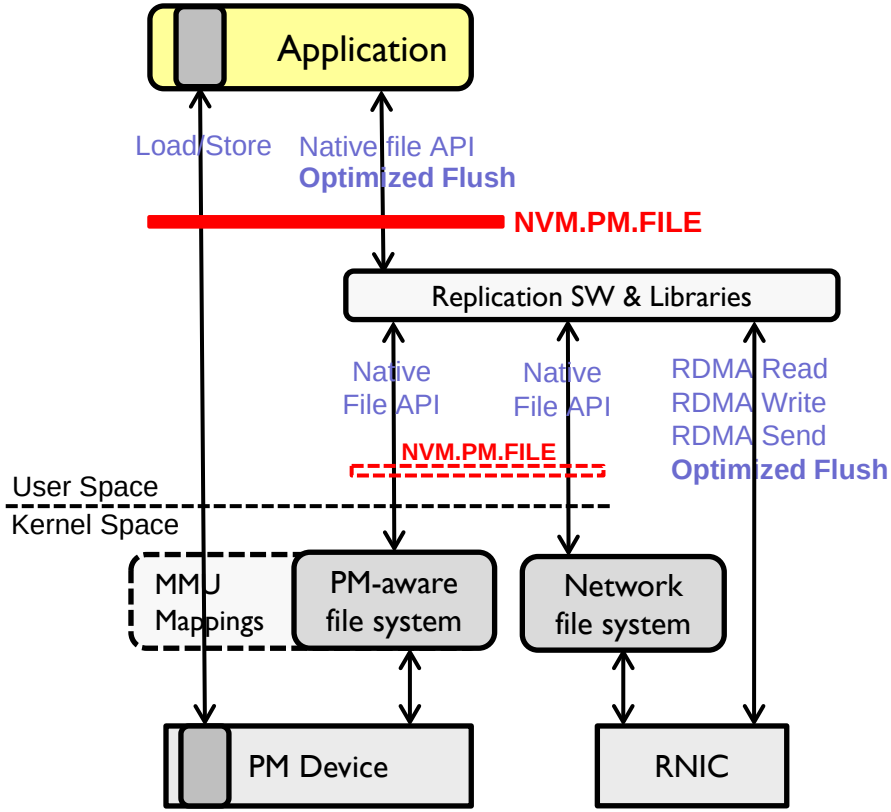


SNIA NVM.PM.FILE

PMoF - OFA libfabric API Case Study

Relating libfabric "Write w Commit" scope to the SNIA Programming Model

- Scope of memory ranges being flushed
  - NVM.PM.FILE.SYNC
    - **Flush Single memory range** within the file, with a single action
  - NVM.PM.FILE.OPTIMIZED_FLUSH
    - **Flush Multiple memory ranges** within the file, with a single action
    - API lifts page alignment constraints of the native file API, enables implementations to more optimally utilize persistent memory, relative to NVM.PM.FILE.SYNC
    - API implementations could reduce or eliminate additional context switches into kernel space

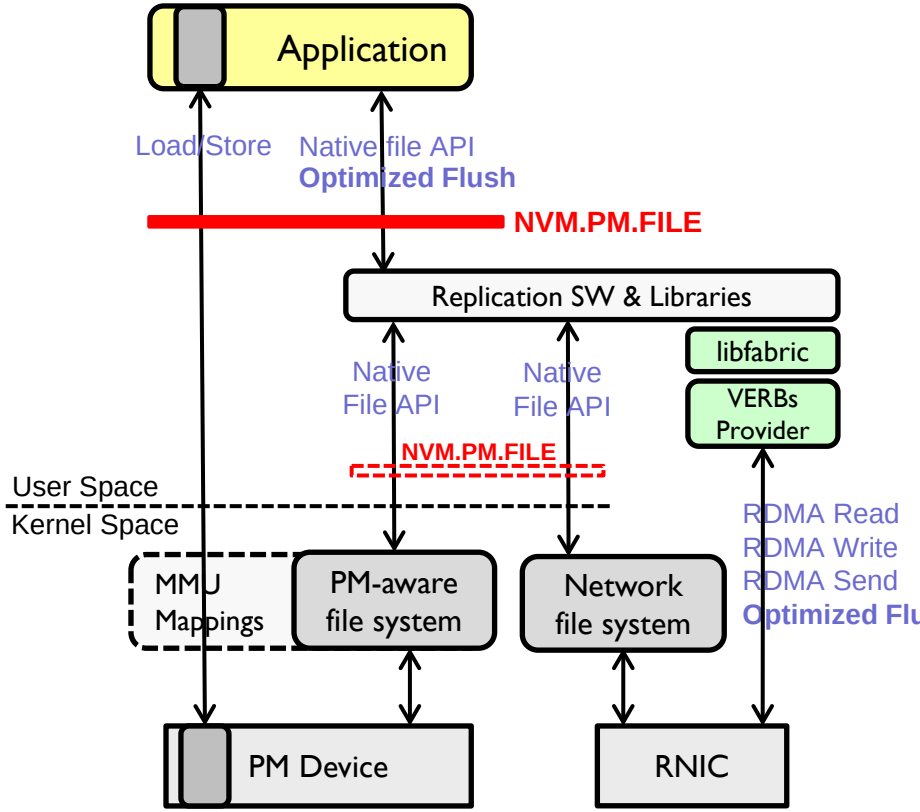


SNIA HA Remote Replication

PMoF - OFA libfabric API Case Study

Relating libfabric "Write w Commit" scope to the SNIA Programming Model

- Addition of libfabric API and provider library doesn't change the top edge of the NVM.PM.FILE API or the Replication SW shown in the SNIA HA model
- **Flexibility of the SNIA NVM Programming allows behavioral model to be applied to different implementations**
- SNIA NVM TWG has discussed this library architecture in some detail and we should revisit these arguments once LWG has made further progress

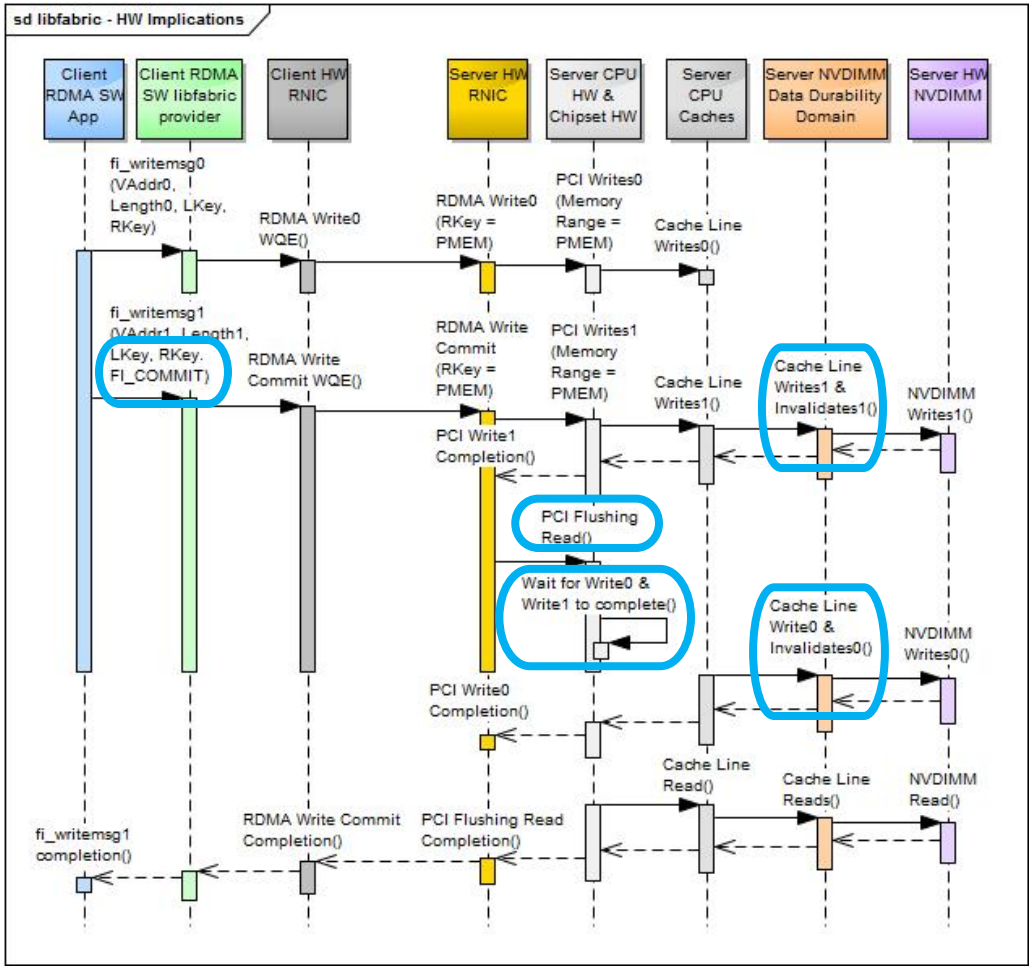


SNIA HA Remote Replication + libfabric

PMoF - OFA libfabric API Case Study

HW Implications for the proposed libfabric API

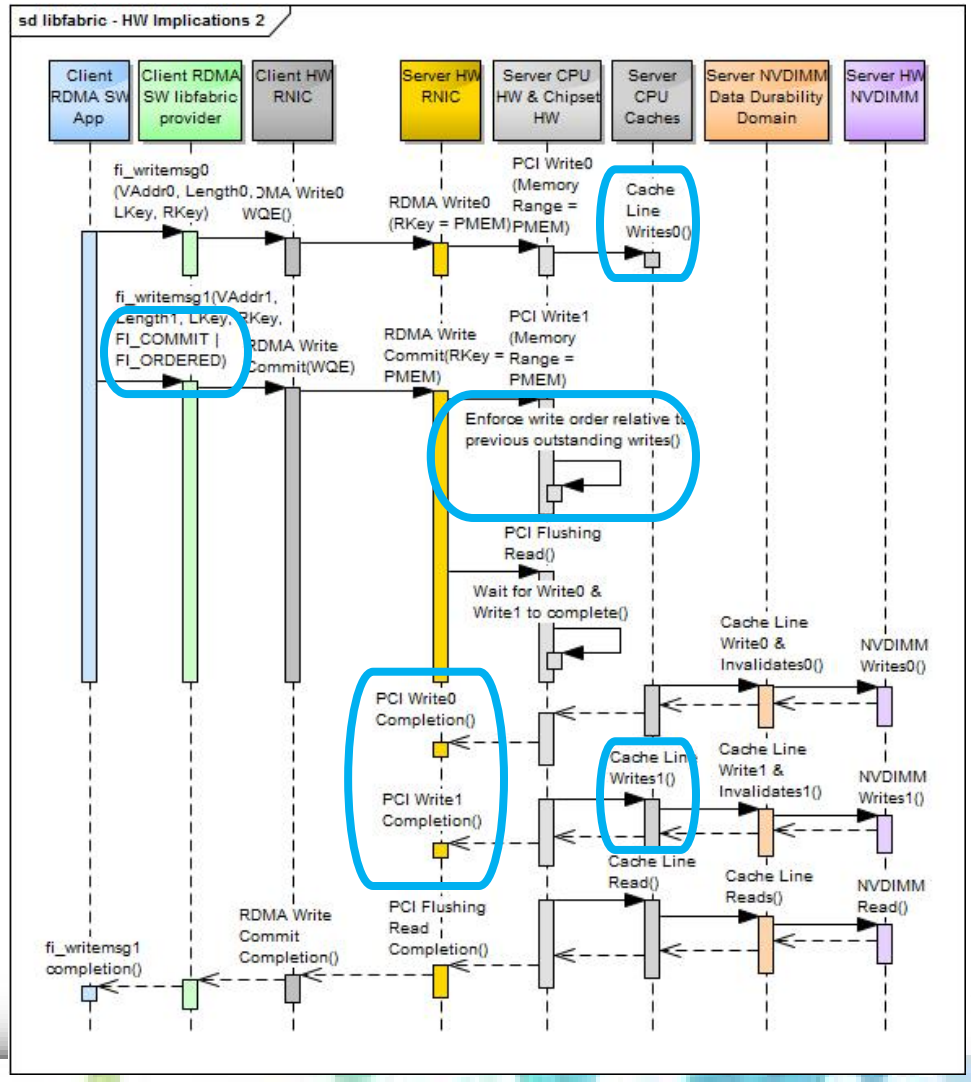
- RNIC HW and CPU/Chipset HW considerations to support fi_writemsg FI_COMMIT
  - RNIC
    - Fabric Write Commit opcode
    - Commit flag needs to force previous writes thru the pipeline
  - Chipset HW
    - Control submission and completion ordering for in-flight writes to PMEM
    - Write data to DRAM versus directly to PMEM
    - Make writes to the NVDIMMs durability domain synchronous



PMoF - OFA libfabric API Case Study

HW Implications for the proposed libfabric API

- RNIC HW and CPU/Chipset HW considerations to support fi_writemsg FI_COMMIT w FI_ORDERED
  - RNIC
    - Fabric Write Commit opcode w Ordered indicator
    - Control ordering of all outstanding writes that have not been started yet when Order Commit received
  - Chipset HW
    - Control ordering of all outstanding writes already in-flight when the Ordered Commit is received
    - Write data to cached memory or directly to PMEM - DDIO
    - Make writes to the NVDIMMs durability domain synchronous



PMoF - OFA libfabric API Case Study

Tying it together

- Native PMEM RDMA support is being discussed through-out the HW and SW industry
- It is helpful to develop common industry wide Use Cases and workloads, develop the network protocol extensions, and HW & SW interfaces, in parallel, and work through the architecture issues together
- libfabric API case study illustrates some of the areas of the existing RDMA architecture that require focused industry alignment and direction for native PMEM support
  - Write Commit execution ordering (initiator & target) relative to other fabric commands
  - Write Commit write data placement ordering at target relative to other write data
  - Controlling ordering with fencing or barriers
  - Completion semantics for data written to the durability domain

PMoF - OFA libfabric API Case Study - Getting Involved & Call to Action

- Native PMEM support with RDMA being discussed in many standards bodies
- Many places to get involved in the technical discussions and help influence the drive towards common industry mechanisms
- Networking Extensions
  - IBTA - Link Working Group (LWG)
    - Designated home for current technical discussions covering the network extensions, ordering, fencing semantics, error handling, etc.
  - IETF
    - Microsoft's Tom Talpey – Basic RDMA Extensions for PMEM – Extension of Tom's SCD2015 and other SNIA discussions for iWARP
- VERB & libibverb Extensions
  - OFA Verbs
    - Pmem support is not being discussed at a significant level.
    - Lets see what happens in the LWG and go from there
- libfabric Extensions
  - OFA DSDA
    - Reviewed this libfabrics proposal in detail in DSDA to start discussing the technical issues/opens, gather directional feedback
    - WG agrees that this is important work and the libraries and API need to implement in application space are very important
    - WG agrees that some form of the API extensions should also be implemented in kernel space as well
  - OFA Main WG
    - Reviewed this libfabrics proposal in detail in DSDA to start discussing the technical issues/opens, gather directional feedback