OPENFABRICS ALLIANCE

SNIA NVM Programming Model Workgroup Update

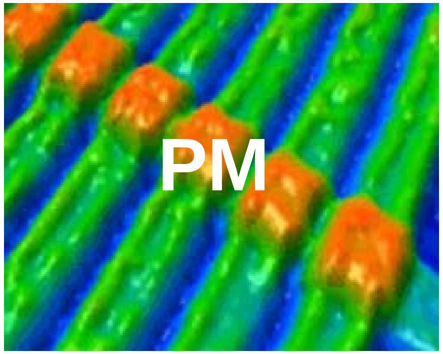
11TH ANNUAL INTERNATIONAL OPENFABRICS SOFTWARE DEVELOPERS' WORKSHOP

#OFADevWorkshop

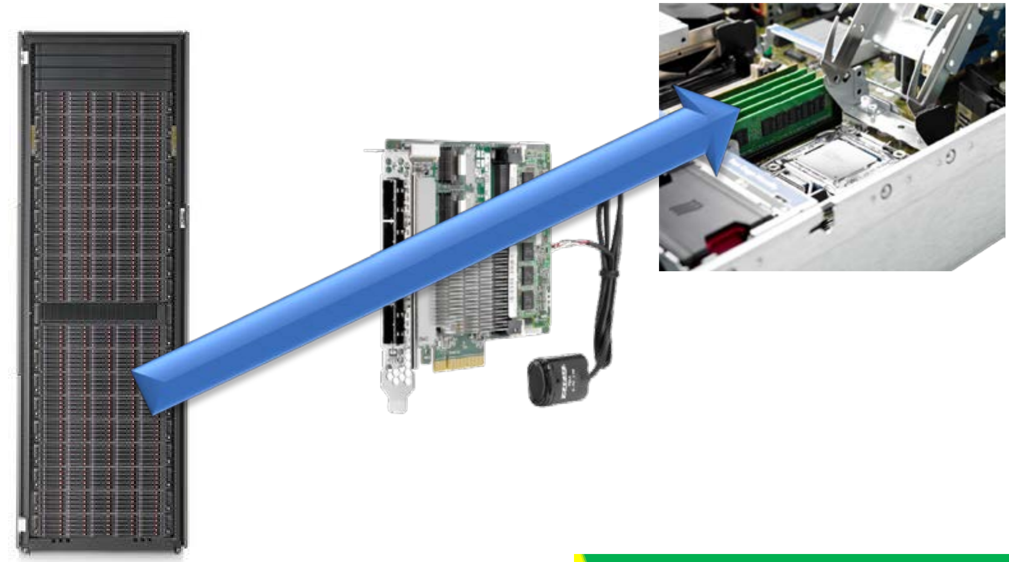
Persistent Memory (PM) Vision

PM Brings Storage

Fast Like Memory



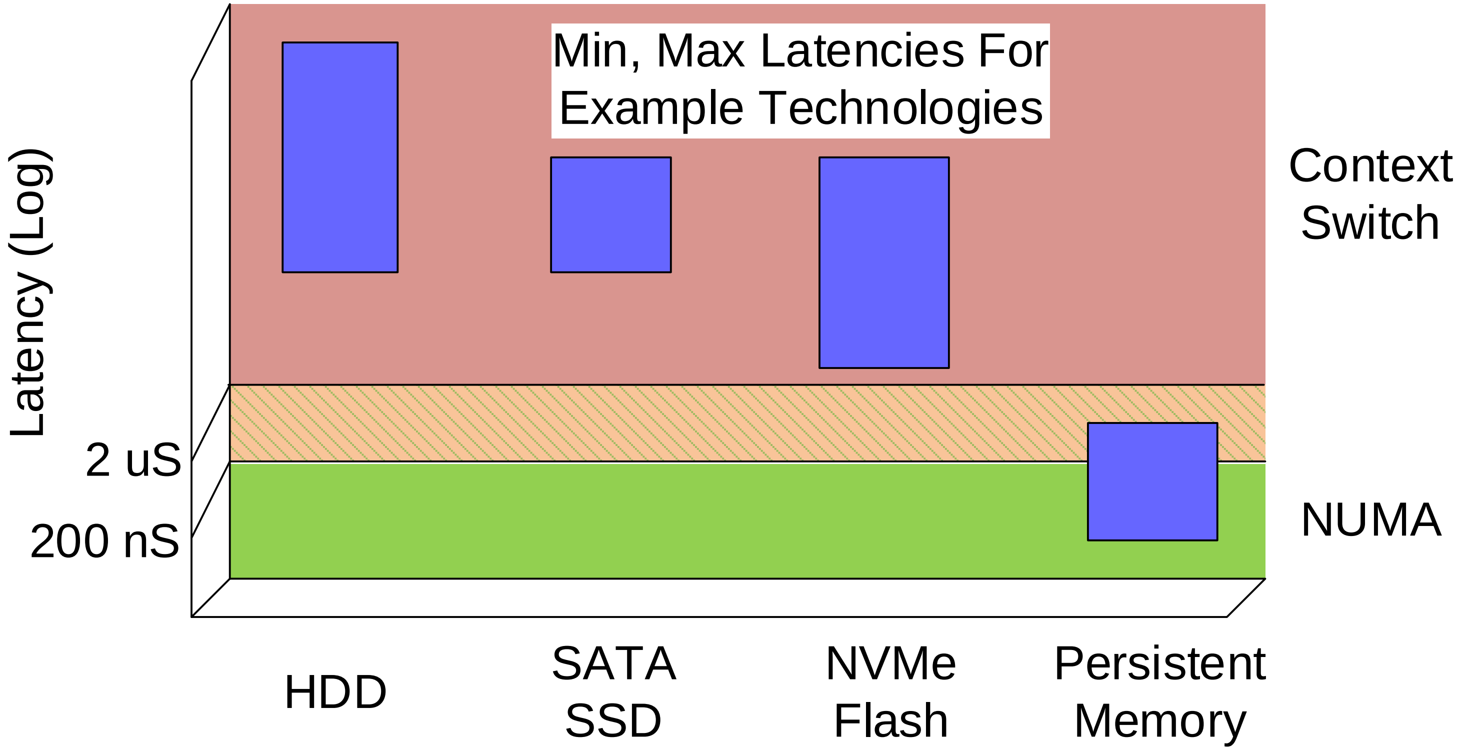
Durable Like Storage



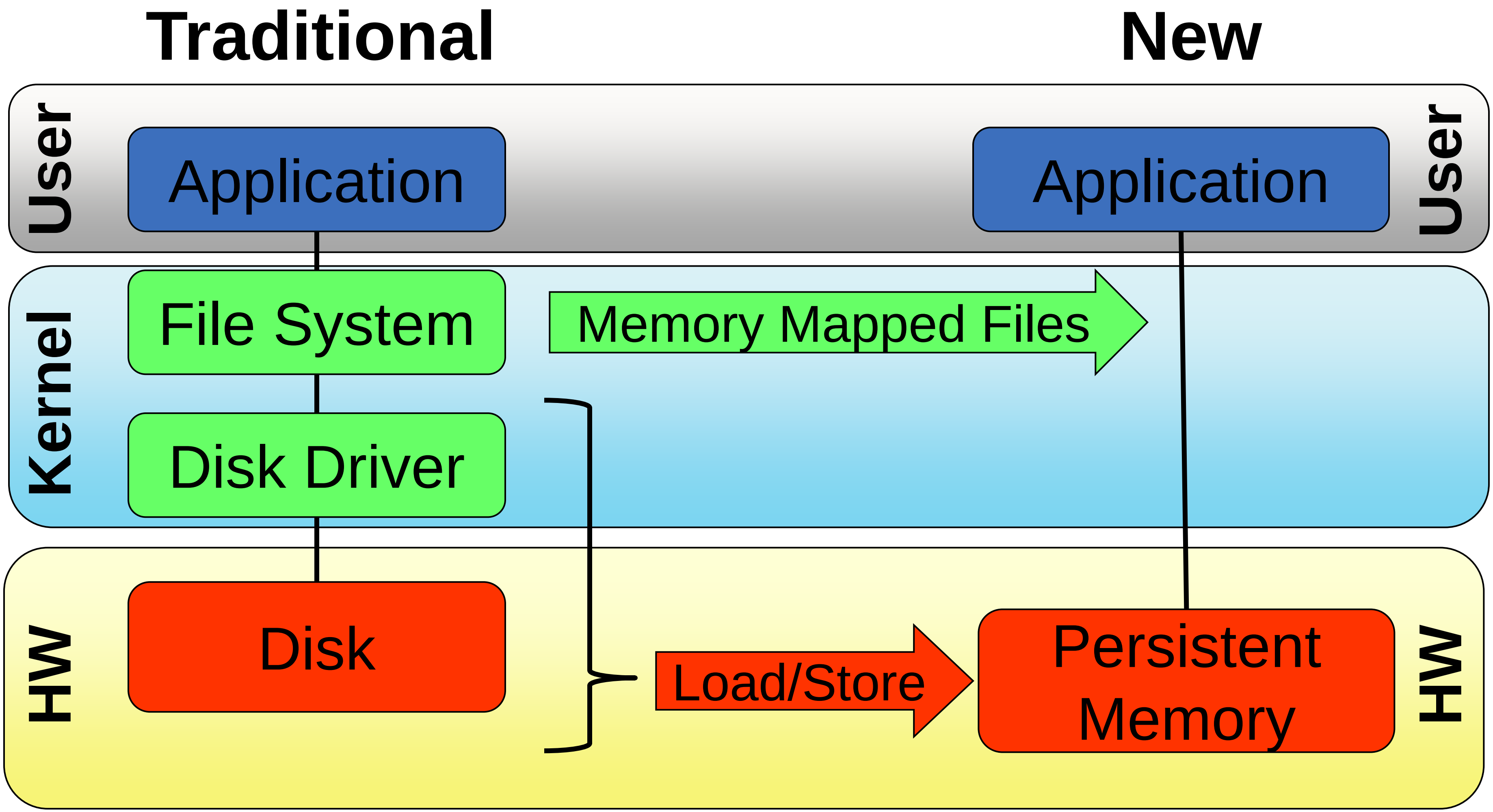
To Memory

Make Data Durable Without Doing IO!

Latency Thresholds Cause Disruption



Eliminate File System Latency with Memory Mapped Files



Version 1 of SNIA NVM Programming Model

- Approved by SNIA in December 2013
  - Downloadable by anyone
  - Version 1.1 approved March 2015
- Expose new block and file features to applications
  - Atomicity capability and granularity
  - Thin provisioning management
- Use of memory mapped files for persistent memory
  - Existing abstraction that can act as a bridge
  - Limits the scope of application re-invention
  - Open source implementations available for incremental innovation (e.g. Linux DAX extensions)
- Programming Model, not API
  - Described in terms of attributes, actions and use cases
  - Implementations map actions and attributes to API's

The 4 Modes

Block Mode Innovation

- Atomics
- Access hints
- NVM-oriented operations

Emerging NVM Technologies

- Performance
- Performance
- Perf… okay, cost

	Traditional	Persistent Memory
User View	NVM.FILE	NVM.PM.FILE
Kernel Protected	NVM.BLOCK	NVM.PM.VOLUME
Media Type	Disk Drive	Persistent Memory
NVDIMM	Disk-Like	Memory-Like

NVM.PM.VOLUME and NVM.PM.FILE

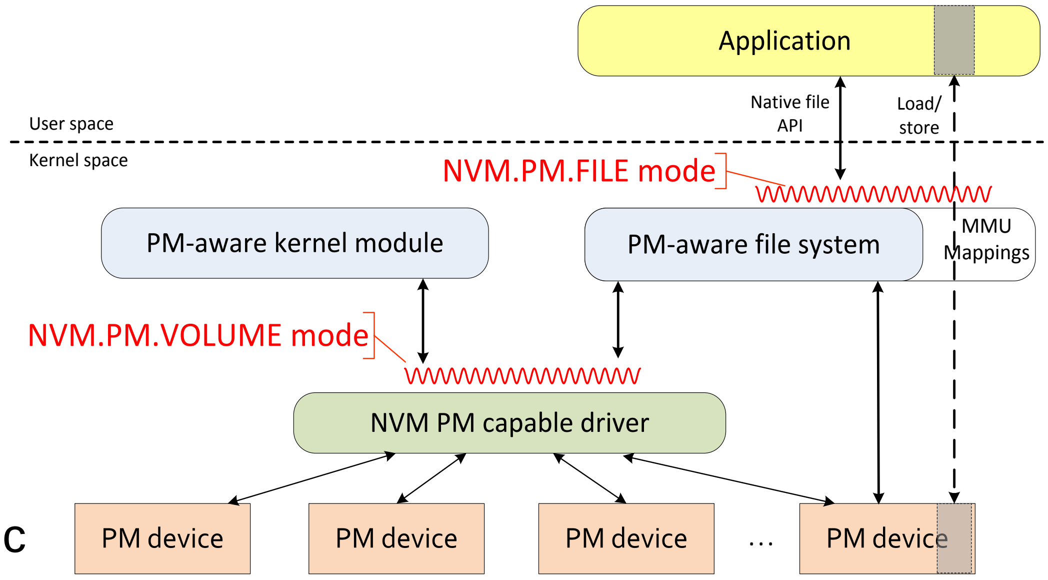
Use with memory-like NVM

NVM.PM.VOLUME Mode

- Software abstraction to OS components for Persistent Memory (PM) hardware
- List of physical address ranges for each PM volume
- Thin provisioning management

NVM.PM.FILE Mode

- Describes the behavior for applications accessing persistent memory Discovery and use of atomic write features
- Mapping PM files (or subsets of files) to virtual memory addresses
- Syncing portions of PM files to the persistence domain



Most Significant Change in NVMP Version 1.1

Data Consistency Requirement: **Atomicity of aligned operations on fundamental data types**

- Aligned Operations:
  - multiple of processor word width
  - Instruction Set Architectures already define them
- Fundamental Data Types
  - Native to languages or libraries
  - Generated by high-level language constructs
- Used by apps in addition to sync for local pfail consistency
- How to extend to remote memory?

Work in progress – Failure Atomicity

- Current processor + memory systems
  - Provide inter-process consistency
  - Not atomicity with respect to failure
    - System reset/restart/crash
    - Power Failure
    - Memory Failure
- Leverage existing research on persistent memory transactions to get failure atomicity
- Describe behaviors required to achieve atomicity of groups of persistent data structures

Related work– Persistent Data Structure Libraries

- Optimal use of PM requires a different style of data structure construction
  - Commits are stores to fundamental data types
  - No marshalling for storage or network IO
- Data structures implemented in libraries
- Examples: Linux Pmem
  - Incudes base class, log, array of blocks, transaction
  - http://pmem.io/nvml/libpmem/
  - https://github.com/pmem/linux-examples

Work in progress – Remote access for High Availability

- Use case: High Availability Memory Mapped Files
  - Built on V1.1 NVM.PM.FILE OptimizedFlush action
  - RDMA copy from local to remote PM
- Requirements:
  - Assurance of remote durability
  - Efficient byte range transfers
  - Efficient large transfers
  - Atomicity of fundamental data types
  - Resource recovery and hardware fencing after failure

Thank You

OPENFABRICS ALLIANCE

11TH ANNUAL
INTERNATIONAL OPENFABRICS SOFTWARE DEVELOPERS' WORKSHOP

OPENFABRICS ALLIANCE

RDMA and NVM Programming Model

11TH ANNUAL
INTERNATIONAL
OPENFABRICS SOFTWARE
DEVELOPERS' WORKSHOP

#OFADevWorkshop

NVM.PM.File.Map, Sync, OptimizedFlush

- Map
  - Associates memory addresses with open file
  - Caller may request specific address
- Sync
  - Flush CPU cache for indicated range
  - Additional Sync types
    - **Optimized Flush – multiple ranges from user space**
    - Optimized Flush and Verify – Optimized flush with read back from media

Low Latency Remote OptimizedFlush

- Remote Access for HA examines OptimizedFlush implementation
  - Goal is to minimize latency
  - Requires at least 2 round trips with today's implementations
  - Main issue is assurance of durability at remote site.
- Use today's RDMA to explore this use case
  - Agnostic to specific implementation (IB, ROCE, iWARP)
  - Optimal implementation may not actually be RDMA

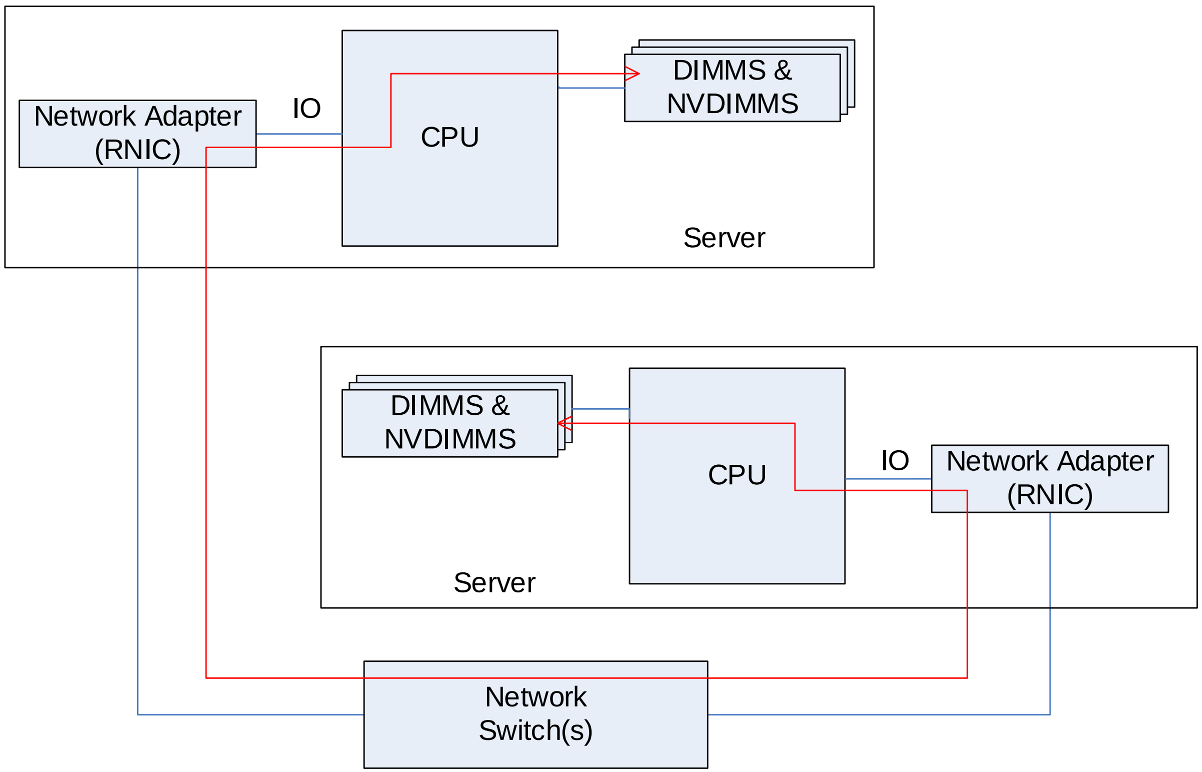
Recovery AND Consistency

- Application level goal is recovery from failure
  - Requires robust local and remote error handling
  - High Availability (as opposed to High Durability) requires application involvement.
- Consistency is an application specific constraint
  - Uncertainty of data state after failure
  - Crash consistency
  - Higher order consistency points
  - Atomicity of Aligned Fundamental Data Types

Application Recovery Scenarios

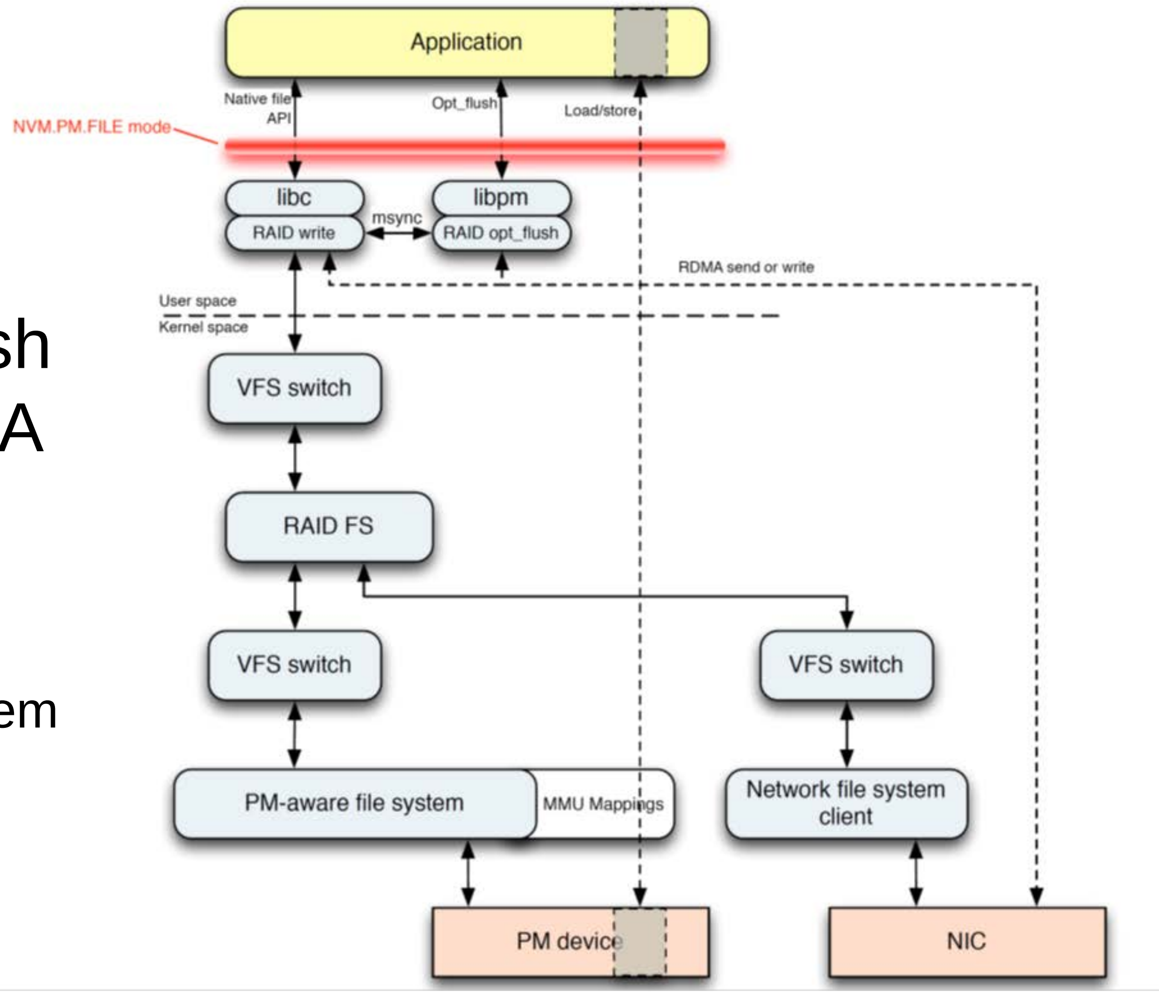
Scenario	Redundancy freshness	Exception	Application backtrack without restart	Server Restart	Server Failure
In Line Recovery	Better than sync	Precise and contained	NA	No	No
Backtracking Recovery	Consistency point	Imprecise and contained	Yes	No	No
Local application restart	Consistency point	Not contained	No	NA	No
		NA	NA	Yes	No
Application Failover	Consistency point	NA	NA	NA	Yes

Remote Access Hardware

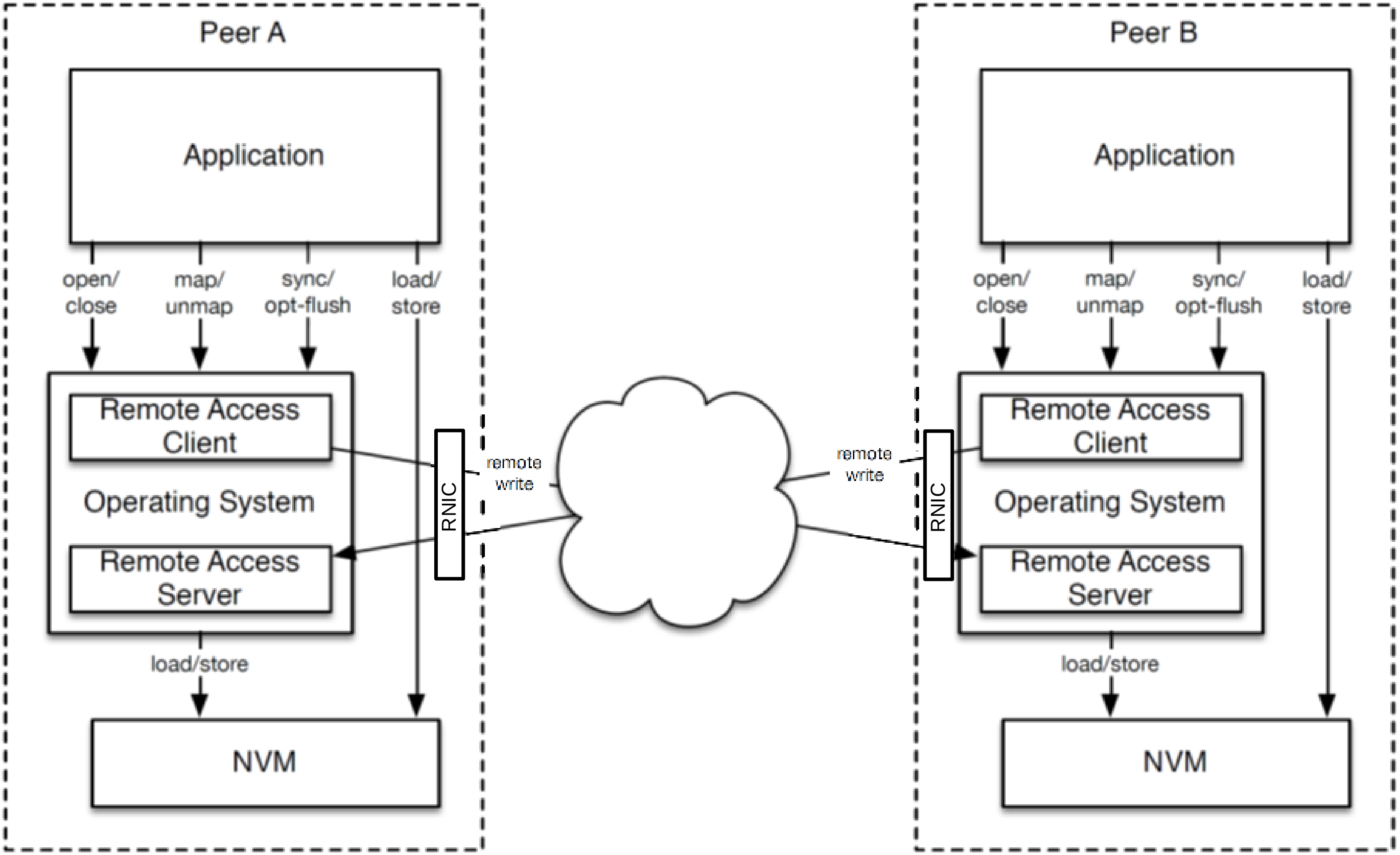


Software Context Example

- Standard file API
- NVM Programming Model optimized flush
- RAID software for HA
  - user space libraries
  - local file system
  - remote file system
    - via network file system client and NIC



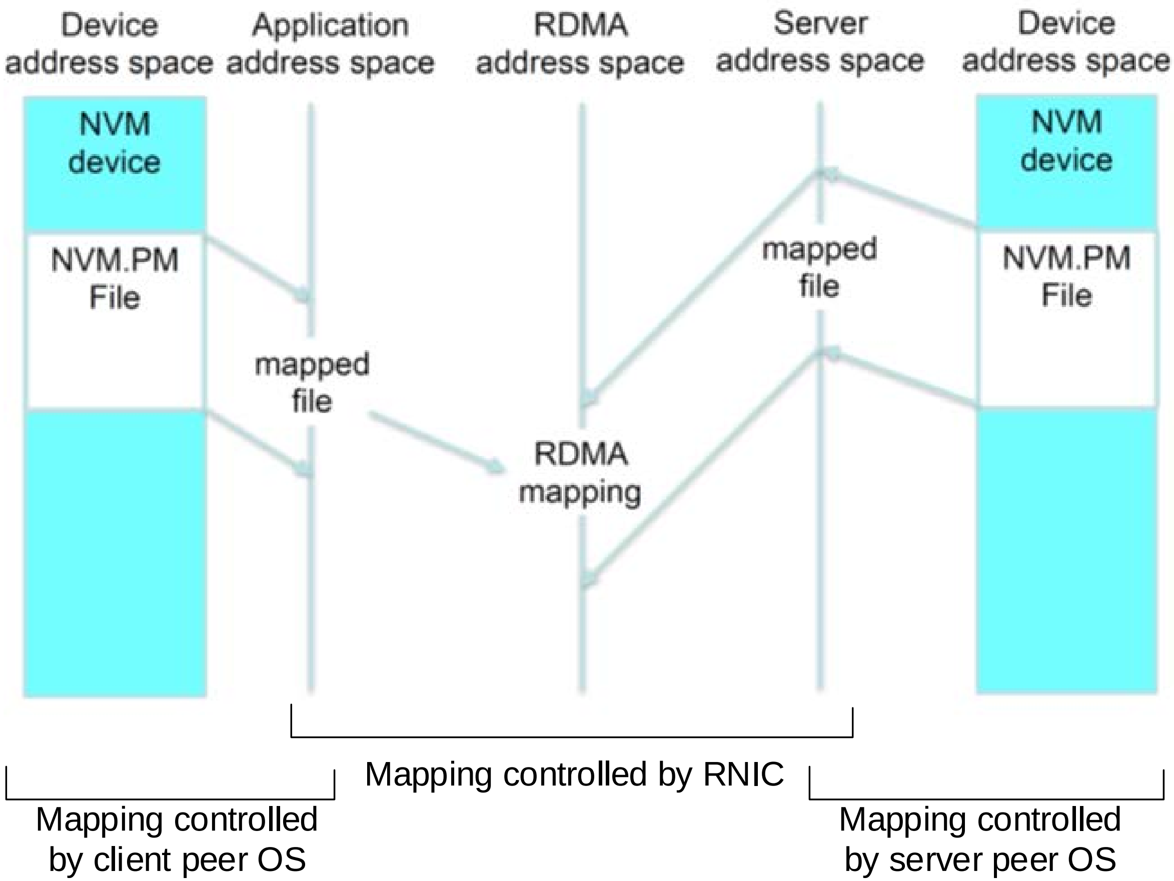
HW/SW View for Data Flow Sequence Diagram



Various Virtual Address Spaces

Only the “Device” address spaces must match

- Sufficiently to allow restoration and failover
- Orchestrated by peer file/operating systems



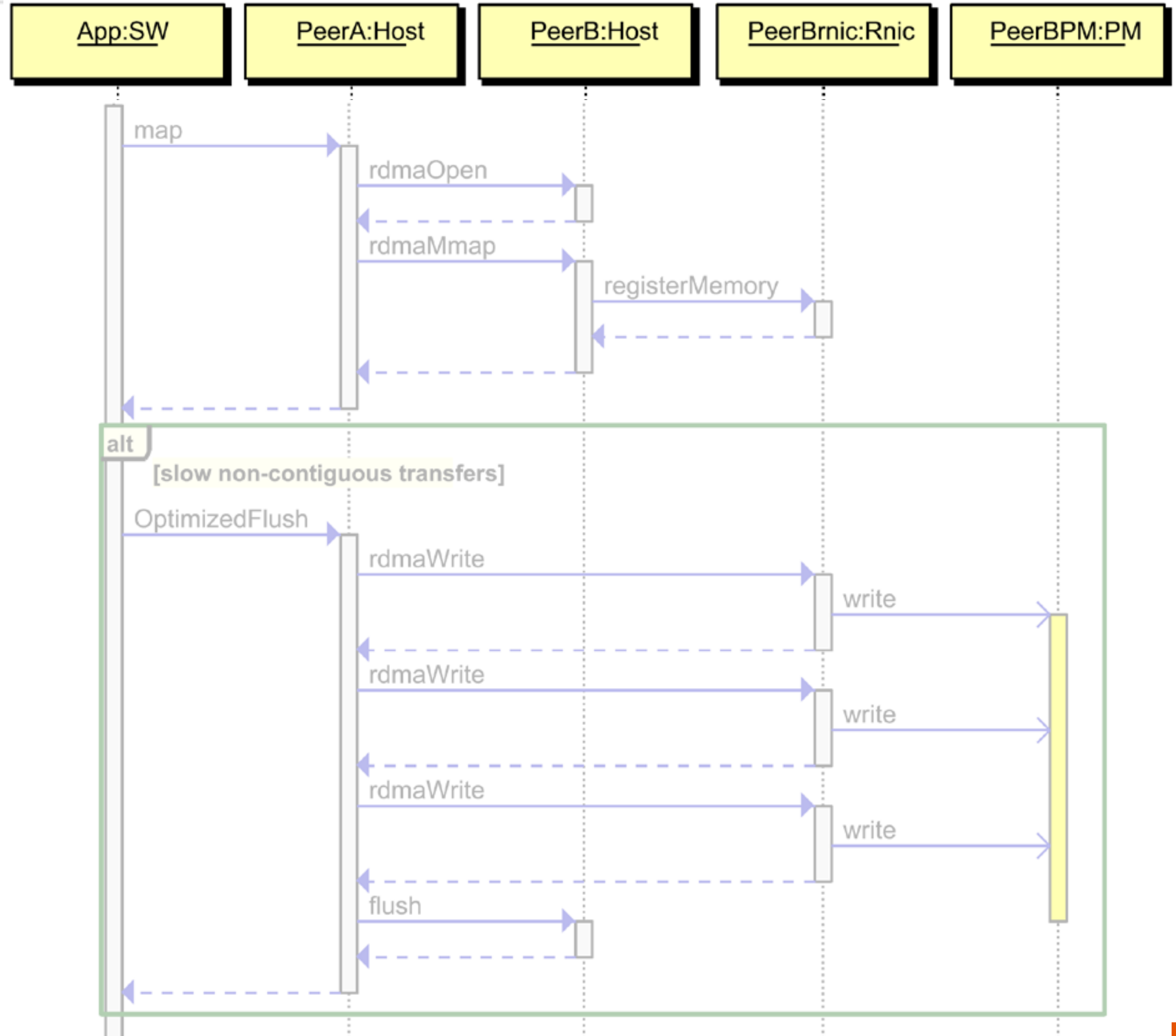
RDMA Flow for HA Optimized Flush

Sequence Diagram actors:
PM aware application
2 hosts mirroring PM
RDMA Adapter (Rnic)

Map triggers RDMA Registration

Optimized Flush triggers dis-contiguous RDMA writes

Flush to guarantee durability and HA



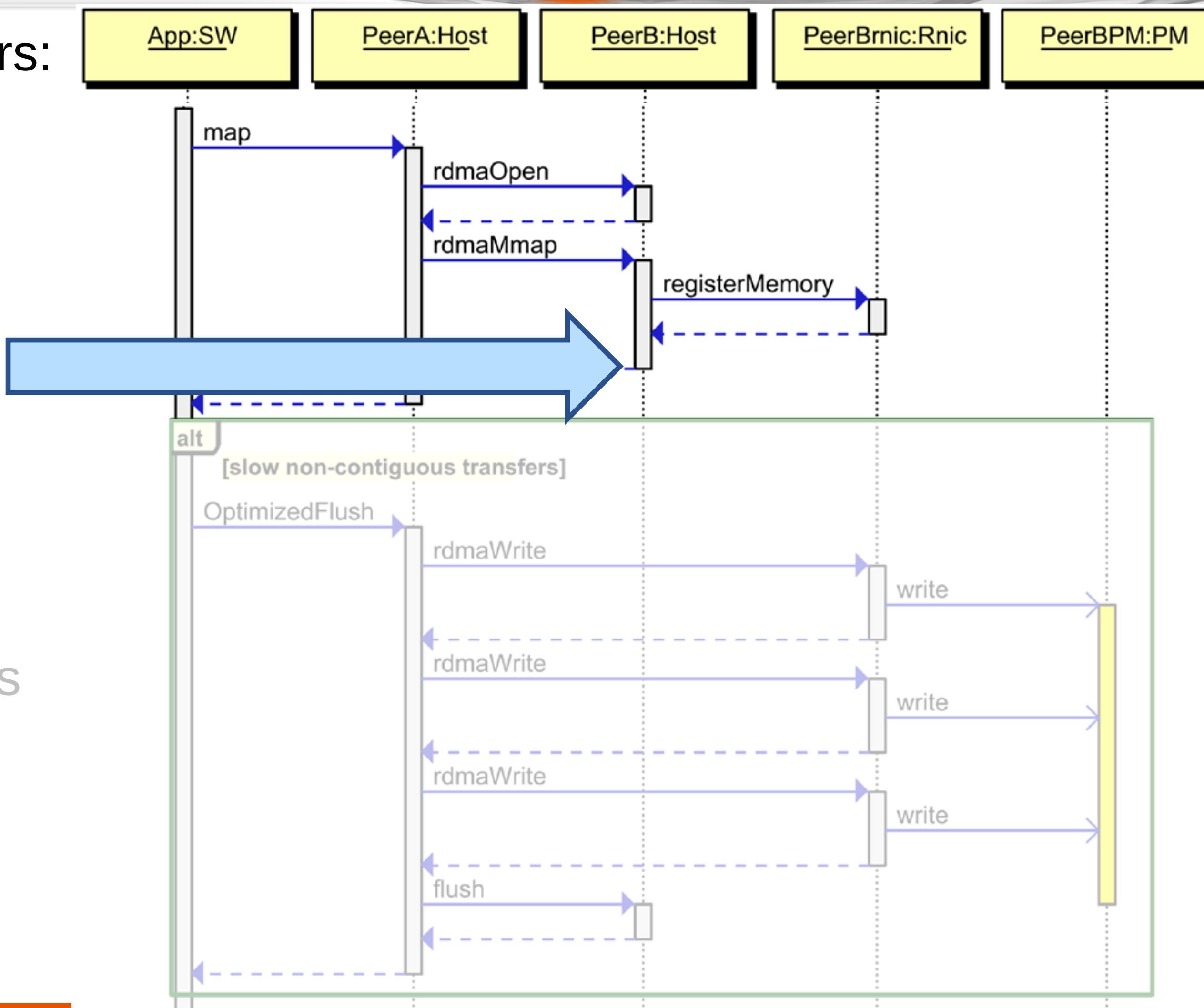
RDMA Flow for HA Optimized Flush

Sequence Diagram actors:
PM aware application
2 hosts mirroring PM
RDMA Adapter (Rnic)

Map triggers RDMA Registration

Optimized Flush triggers dis-contiguous RDMA writes

Flush to guarantee durability and HA



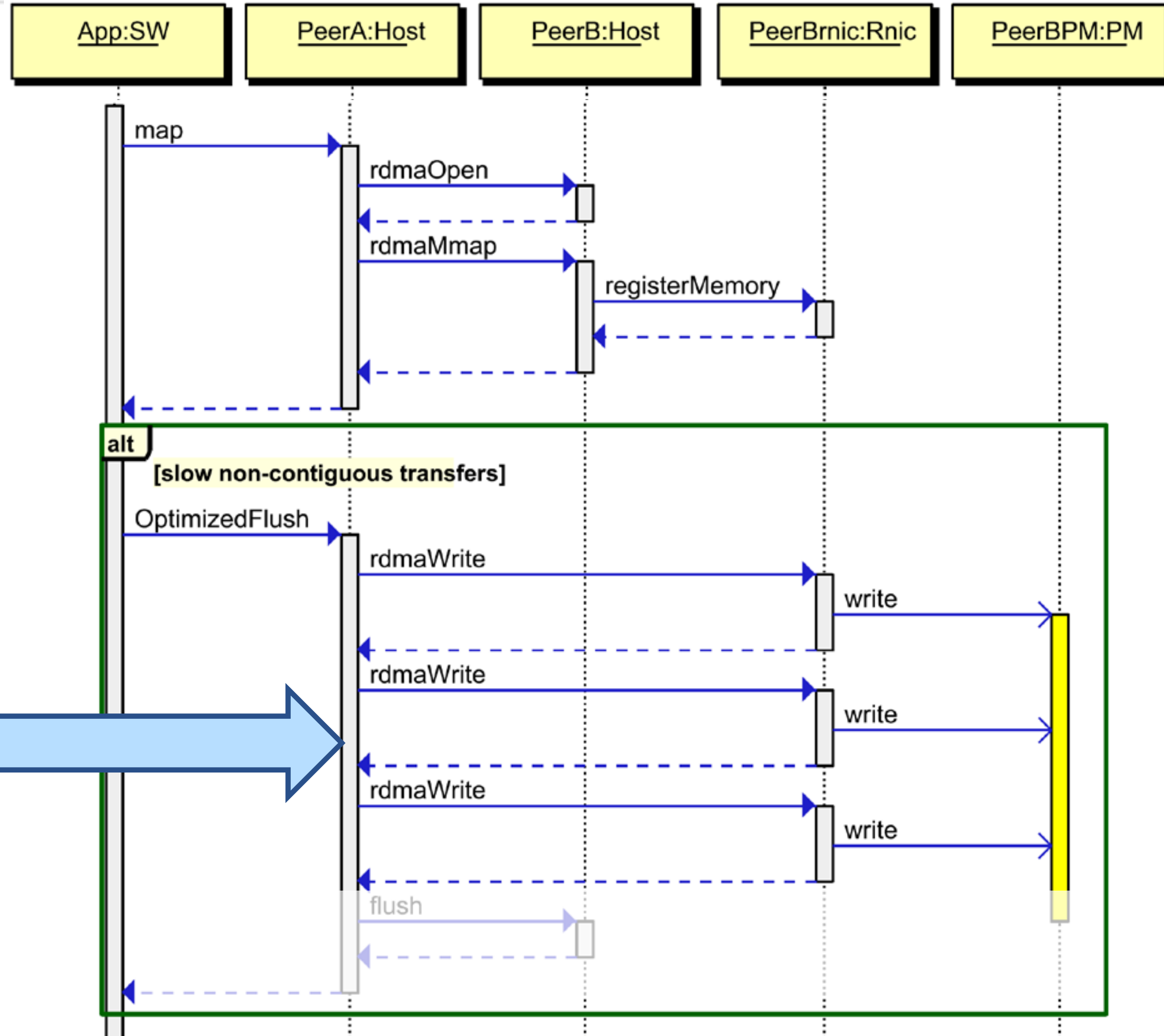
RDMA Flow for HA Optimized Flush

Sequence Diagram actors:
PM aware application
2 hosts mirroring PM
RDMA Adapter (Rnic)

Map triggers RDMA Registration

Optimized Flush triggers dis-contiguous RDMA writes

Flush to guarantee durability and HA



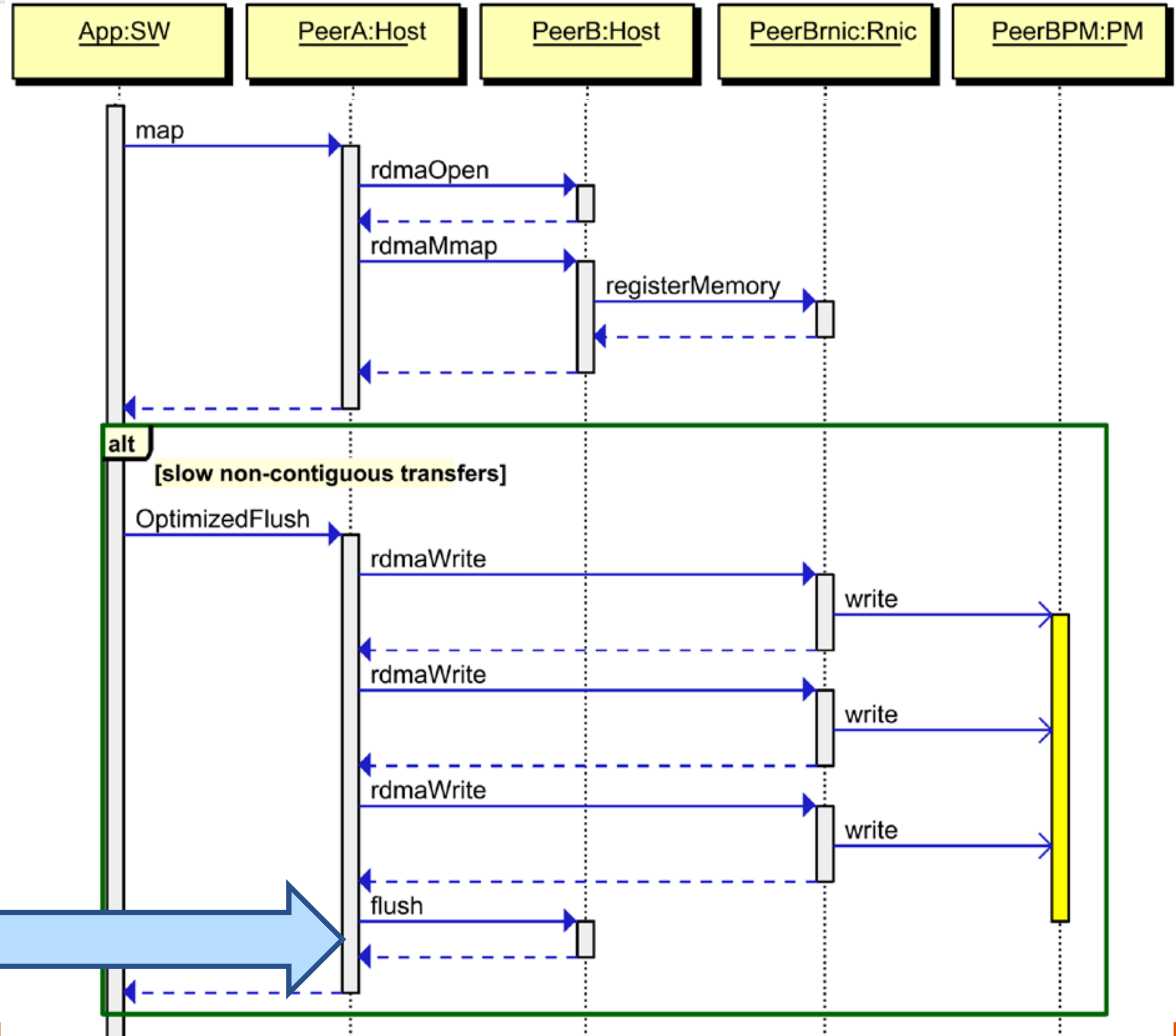
RDMA Flow for HA Optimized Flush

Sequence Diagram actors:
PM aware application
2 hosts mirroring PM
RDMA Adapter (Rnic)

Map triggers RDMA Registration

Optimized Flush triggers dis-contiguous RDMA writes

Flush to guarantee durability and HA



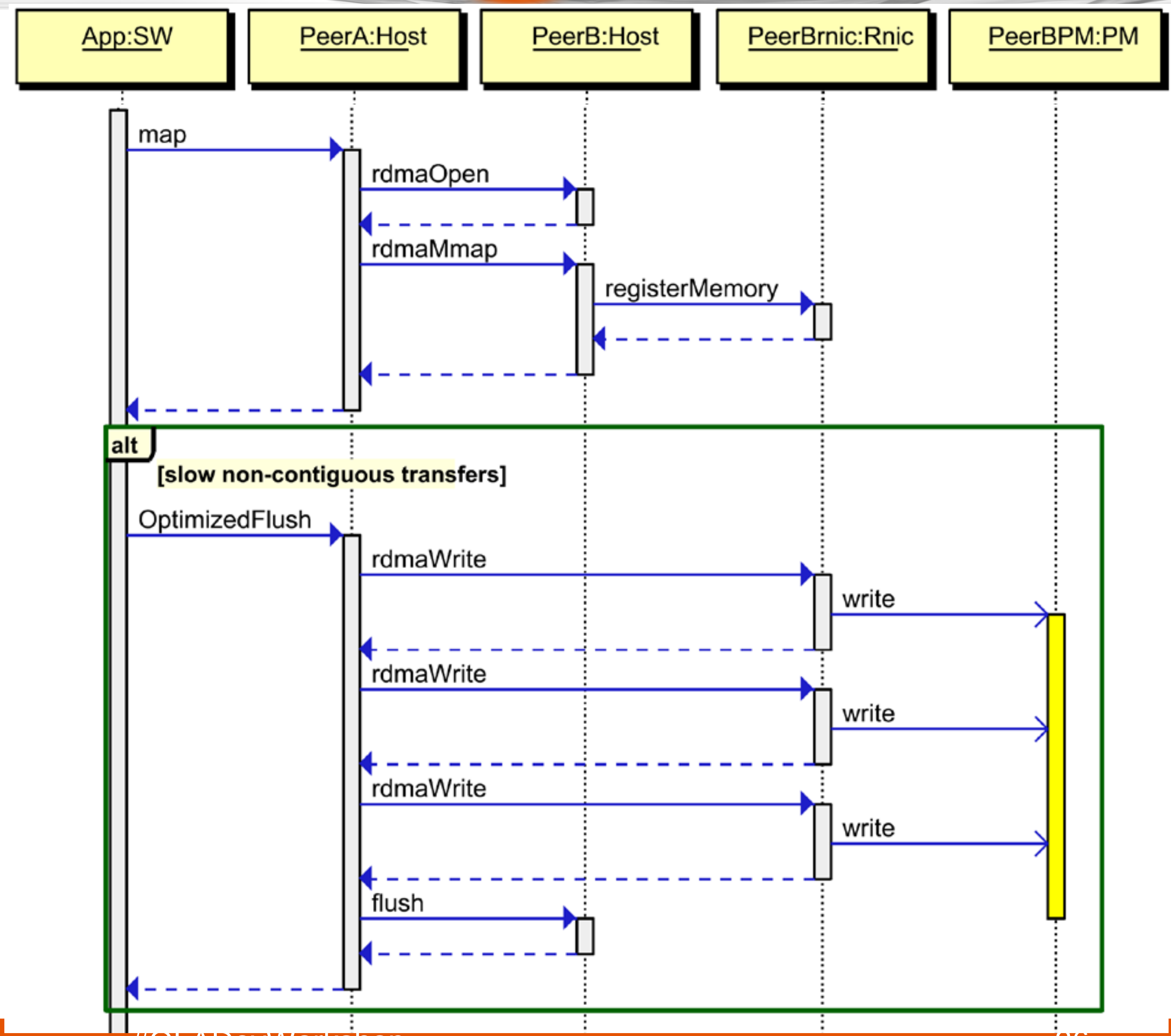
RDMA Flow for HA Optimized Flush

Sequence Diagram actors:
PM aware application
2 hosts mirroring PM
RDMA Adapter (Rnic)

Map triggers RDMA Registration

Optimized Flush triggers dis-contiguous RDMA writes

Flush to guarantee durability and HA



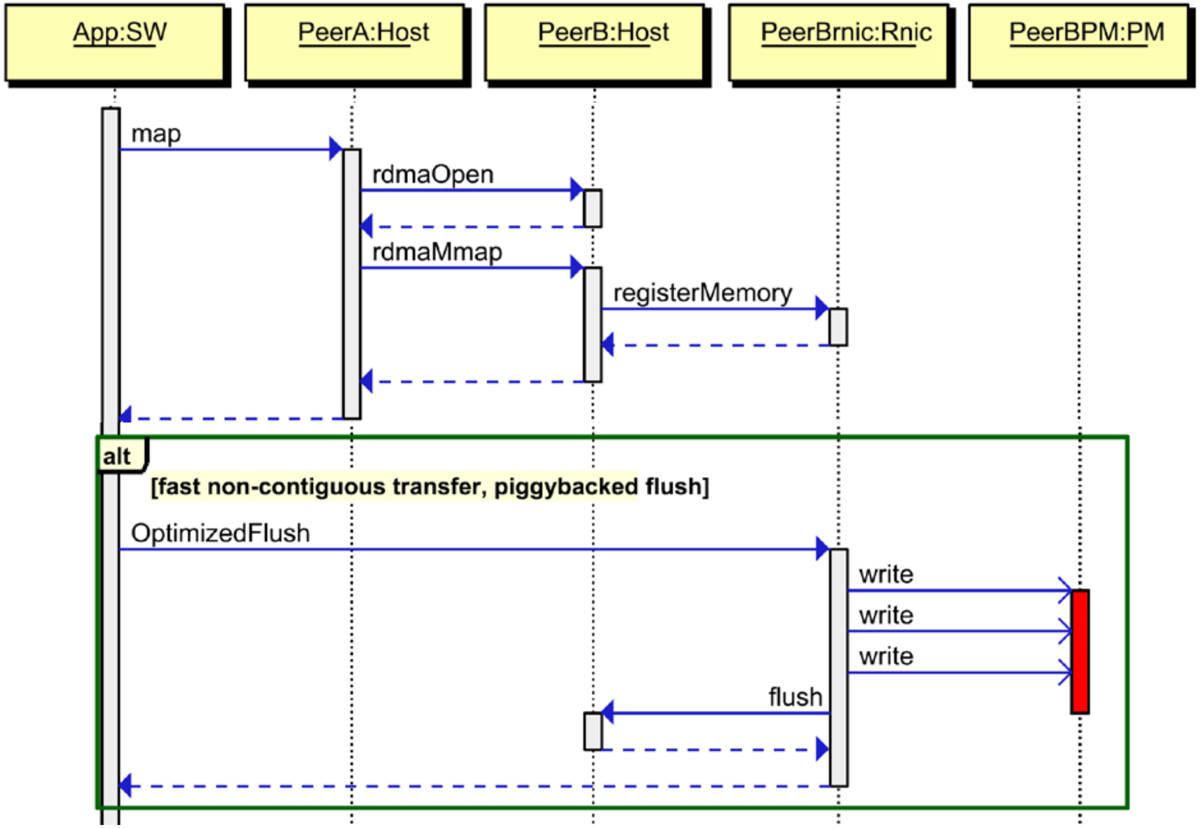
RDMA Flow for HA MORE Optimized Flush

Sequence Diagram actors:
PM aware application
2 hosts mirroring PM
RDMA Adapter (Rnic)

Map triggers RDMA Registration

Optimized Flush triggers multi-range RDMA writes

Piggybacked with remote flush



Work in progress – Remote access for High Availability

- Use case: High Availability Memory Mapped Files
  - Built on V1.1 NVM.PM.FILE OptimizedFlush action
  - RDMA copy from local to remote PM
- Requirements:
  - Assurance of remote durability
  - Efficient byte range transfers
  - Efficient large transfers
  - Atomicity of fundamental data types
  - Resource recovery and hardware fencing after failure
- NVM PM Remote Access for High Availability

Thank You

OPENFABRICS ALLIANCE

11TH ANNUAL INTERNATIONAL OPENFABRICS SOFTWARE DEVELOPERS' WORKSHOP