STORAGE INDUSTRY SUMMIT

The Future of Computing: The Convergence of Memory and Storage through Non-Volatile Memory (NVM)

JANUARY 28, 2014, SAN JOSE, CA

Jeff Moyer

Red Hat, Inc.

Principal Software Engineer

The State of Persistent Memory in Linux

SNIA™ Solid State Storage Initiative

Agenda

- Persistent Memory (PM) Introduction
- Implementation Goals
- Current Work
- Relation to the NVM Programming Model
- Challenges for the (Near) Future
- Ways to Get Involved

Persistent Memory

definition: storage technology with performance characteristics suitable for a load and store programming model

- Key Features
  - Byte addressable
  - Low latency
- Unknowns
  - Capacity
  - Cost

Goals

- Hardware Enablement
- Support Existing Storage Stack
- Allow Early Adopters to Innovate

Current State

- Block Driver
- Direct mmap

Block Driver

- The Good
  - Enables entire storage stack
- The Bad
  - Double Buffering
  - Memcpy Eats Memory Bandwidth
- Unique Challenges
  - Synchronous I/O
  - Power-fail write atomicity

Mmap

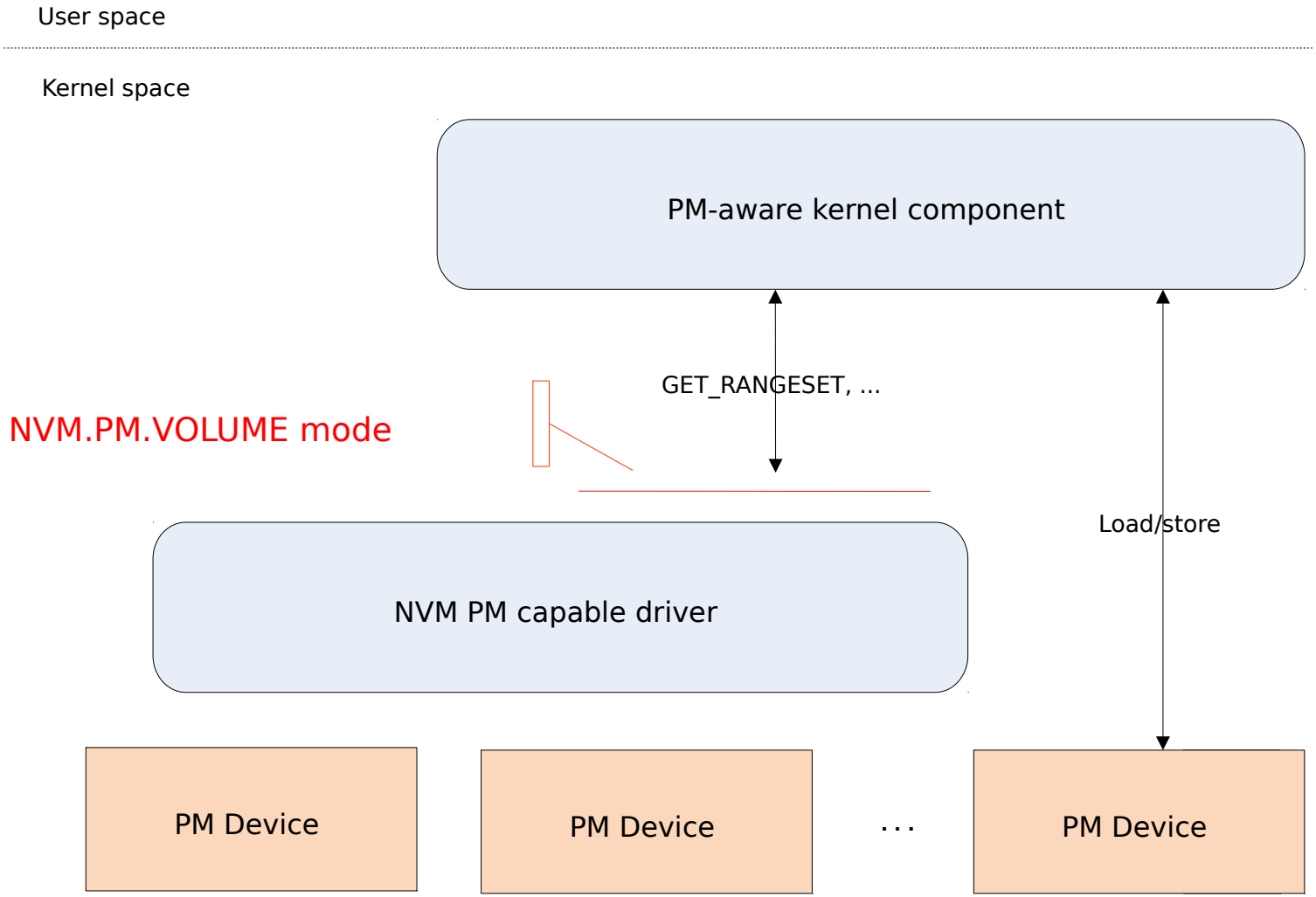
- Ext4/XFS + eXecute In Place (XIP)

- Limitations
  - No “struct page” for PM
  - File system is not PM-aware

NVM Programming Model Specification

- Block Storage
  - NVM.BLOCK
  - NVM.FILE
- PM
  - NVM.PM.VOLUME
  - NVM.PM.FILE

NVM.PM.VOLUME



NVM.PM.VOLUME Mapping

- Attributes
  - `NVM.PM.VOLUME.VOLUME_SIZE`
    - struct block_device.bd_inode->i_size
  - `NVM.PM.VOLUME.INTERRUPTED_STORE_ATOMICITY`
    - hardware dependent, not currently exported
  - `NVM.PM.VOLUME.FUNDAMENTAL_ERROR_RANGE`
    - queue_limits.physical_block_size
  - `NVM.PM.VOLUME.FUNDAMENTAL_ERROR_RANGE_OFFSET`

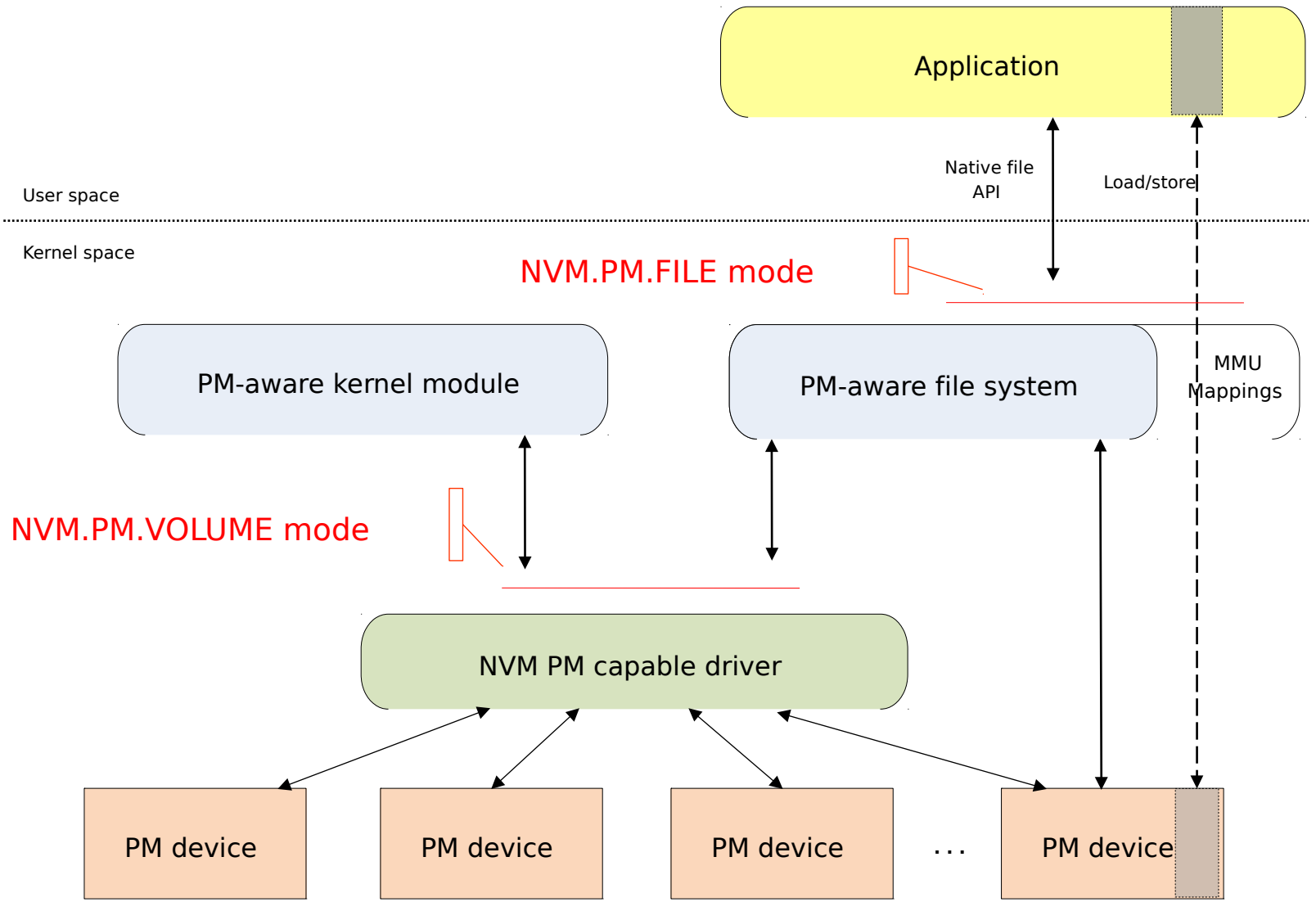
NVM.PM.VOLUME Mapping (cont'd)

- Attributes (cont'd)
  - NVM.PM.VOLUME.DISCARD_IF_YOU_CAN_CAPABLE
    - *QUEUE_FLAG_DISCARD*
  - NVM.PM.VOLUME.DISCARD_IMMEDIATELY_CAPABLE
    - *QUEUE_FLAG_SECDISCARD or QUEUE_FLAG_DISCARD* and discard_zeroes_data
  - NVM.PM.VOLUME.DISCARD_IMMEDIATELY_RETURNS
    - discard_zeroes_data
  - NVM.PM.VOLUME.EXISTS_CAPABLE
    - Not Implemented

NVM.PM.VOLUME Mapping (cont'd 2)

- Actions
  - NVM.PM.VOLUME.GET_RANGESET
  - NVM.PM.VOLUME.VIRTUAL/PHYSICAL_ADDRESS_SYNC
  - NVM.PM.VOLUME.DISCARD_IF_YOU_CAN/IMMEDIATELY
  - NVM.PM.VOLUME.EXISTS

NVM.PM.FILE


Application
Native file API
Load/store
User space
Kernel space
NVM.PM.FILE mode
PM-aware kernel module
PM-aware file system
MMU Mappings
NVM.PM.VOLUME mode
NVM PM capable driver
PM device
PM device
PM device
...
PM device

NVM.PM.FILE Mapping

- Attributes
  - NVM.PM.FILE.MAP_COPY_ON_WRITE_CAPABLE
  - NVM.PM.FILE.OPTIMIZED_FLUSH_CAPABLE
  - NVM.PM.FILE.ERROR_EVENT_CAPABLE
  - NVM.PM.FILE.
    OPTIMIZED_FLUSH_AND_VERIFY_CAPABLE

Summary

- Goals
  - Enable existing storage stack
  - Allow innovation
- Block driver implemented
- XIP exports PM to userspace

Future Work

- Performance Analysis
- PM-specific APIs
- Add PM Support to Existing Storage Libraries
  - Transaction Managers
  - Device Mapper Targets

Getting Involved

- Mailing Lists
  - linux-fsdevel@vger.kernel.org, linux-kernel@vger.kernel.org
- Linux Foundation Events
  - LSF/MM Summit, Collab Summit, Plumbers
  - http://events.linuxfoundation.org/
- Engage with Red Hat