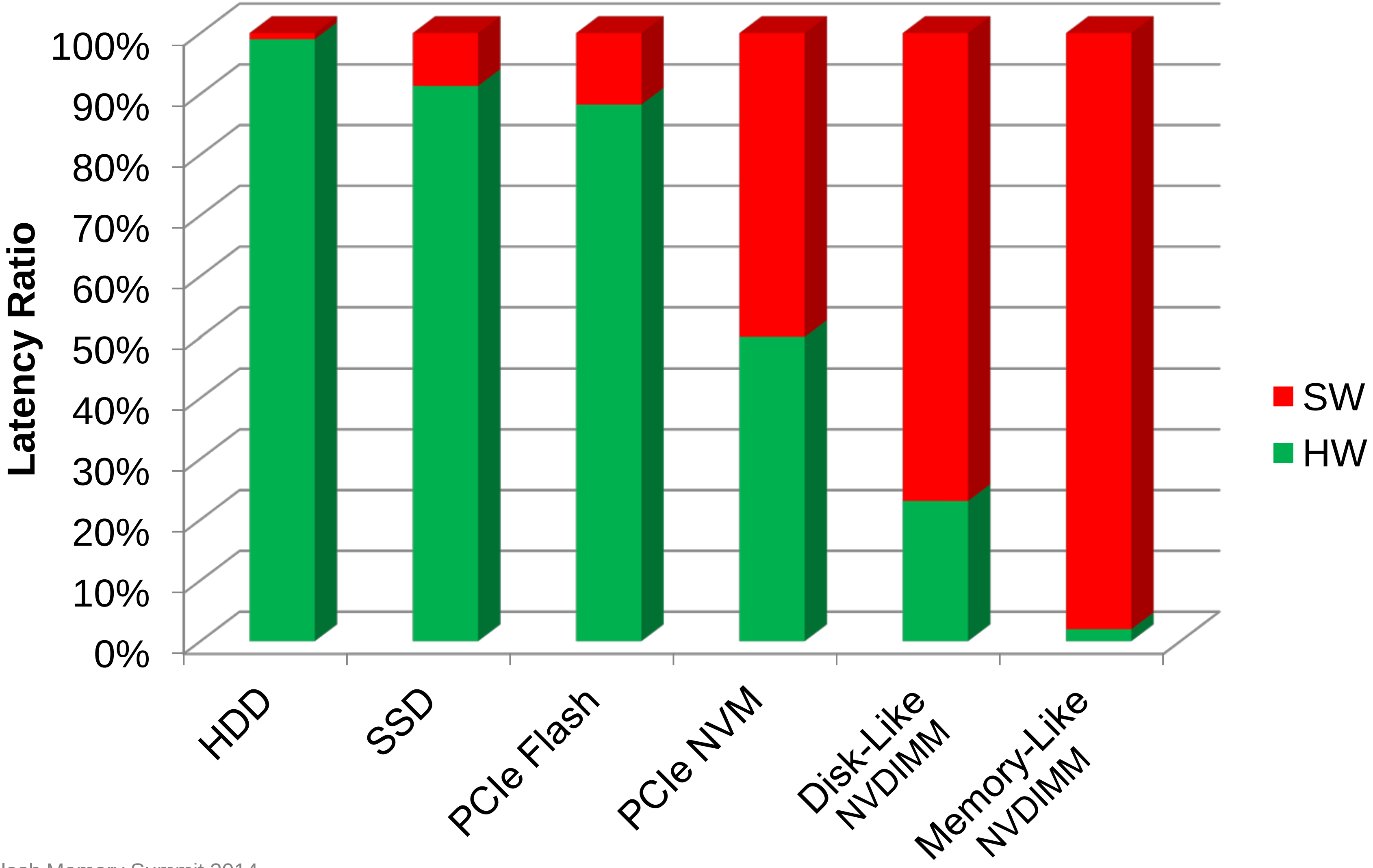
SNIA’s New NVM Programming Model

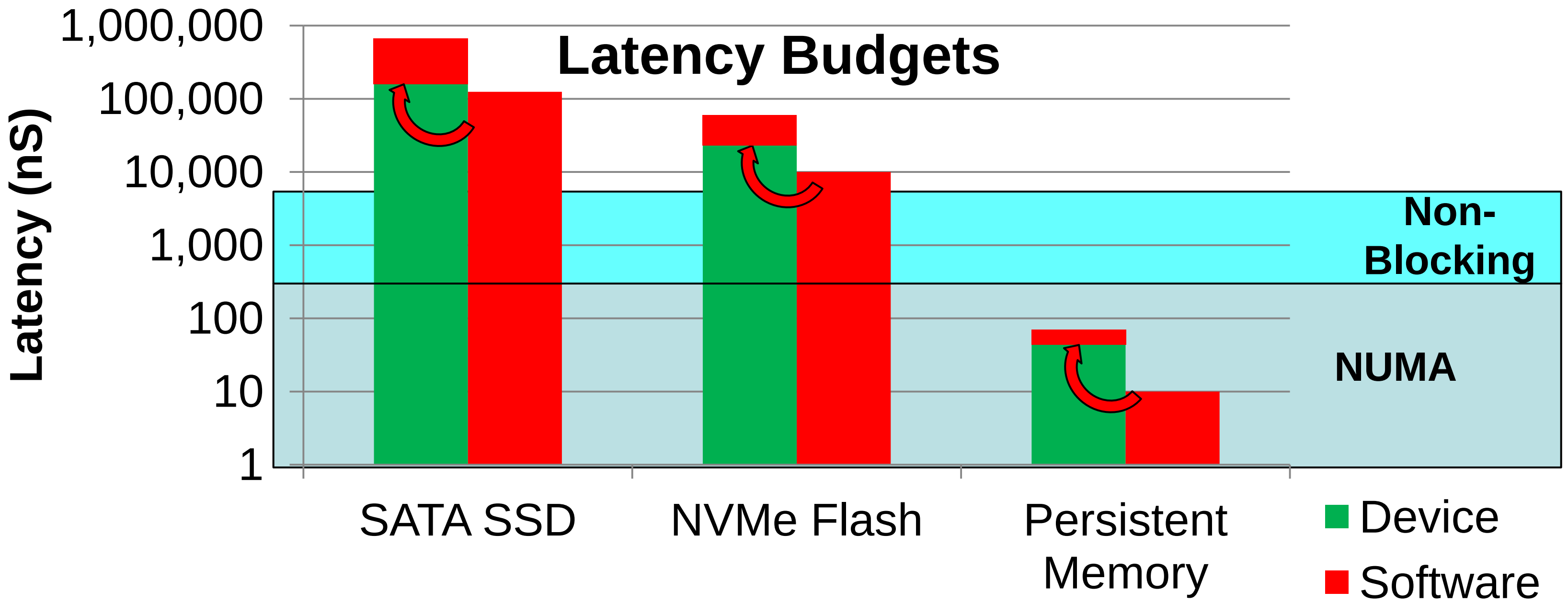
Doug Voigt

In NVM technologies, HW is getting ahead of SW … again!



Data from Intel and HP Presentations at NVM Summit 2014

Application View of IO Elimination



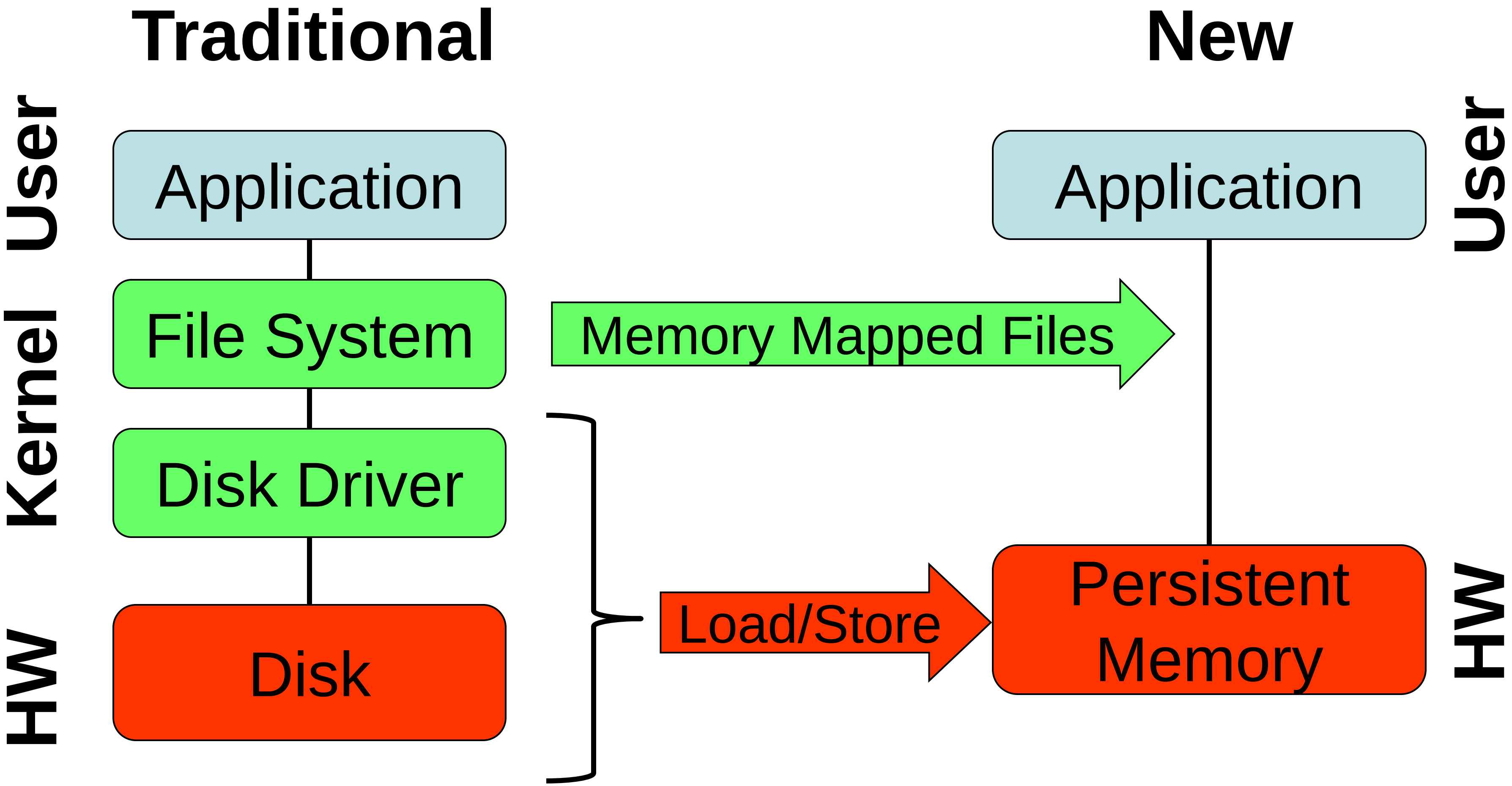
Software overheads are being driven to keep pace with devices
NUMA latencies up to 200 nS have historically been tolerated
Anything above 2-3 uS will probably need to context switch
Latencies below these thresholds cause disruption

Application Access to NVDIMMS

- Disk-like NVDIMMs
    - Appear as disk drives to applications
    - Accessed using disk stack
- Memory-like NVDIMMs
    - Appear as memory to applications
    - Applications store variables directly in RAM
    - No IO or even DMA is required

Memory-like NVDIMMs are
a type of persistent memory

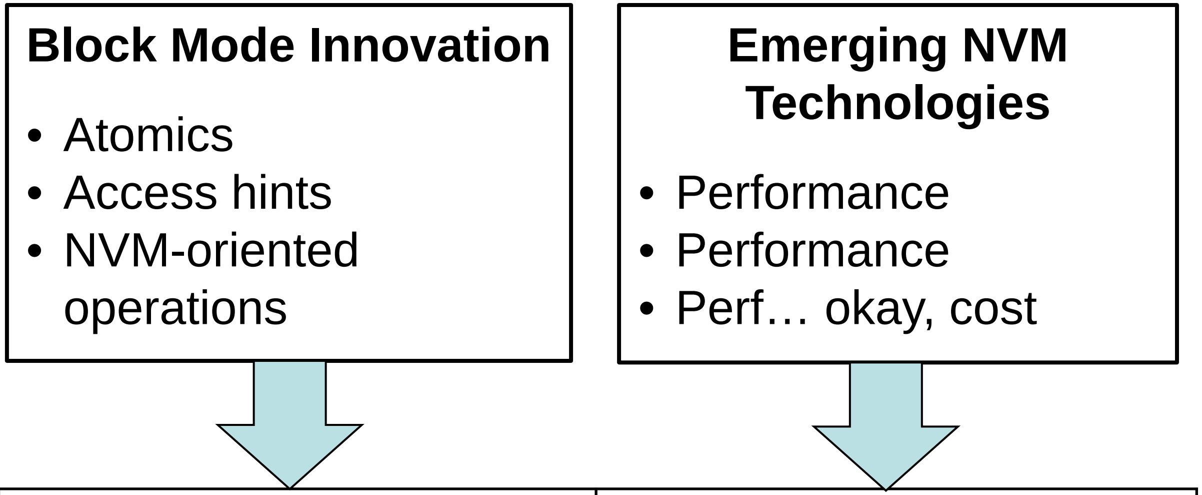
Eliminate File System Latency with Memory Mapped Files



SNIA NVM Programming Model

- Version 1 approved by SNIA in December 2013
  - Downloadable by anyone
- 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. PMFS)
- Programming Model, not API
  - Described in terms of attributes, actions and use cases
  - Implementations map actions and attributes to API’s

The Four 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

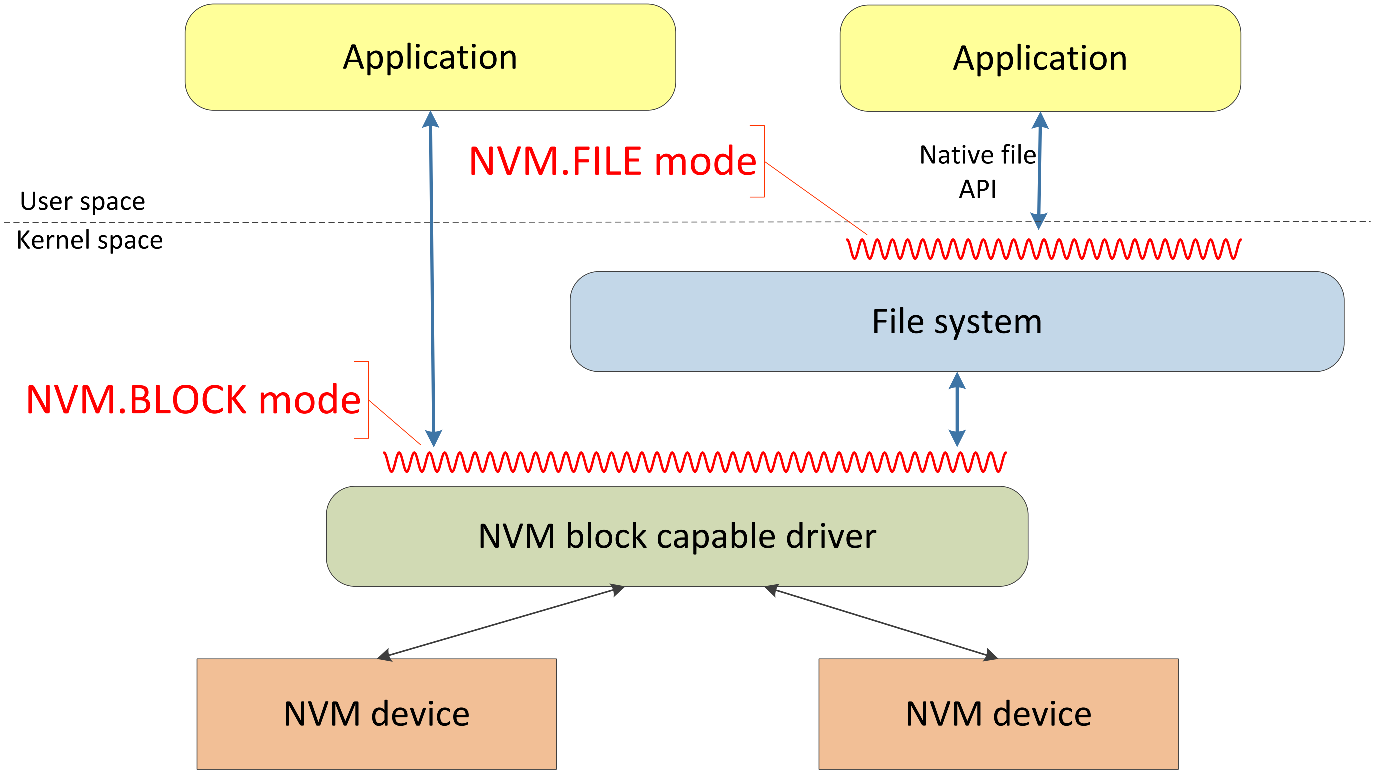
Conventional Block and File Modes Use with disk-like NVDIMMs

BLOCK mode describes extensions:

- Atomic write features
- Granularities (length, alignment)
- Thin Provisioning Management

FILE mode describes extensions:

- Discovery and use of atomic write features
- The discovery of granularities (length, alignment characteristics)



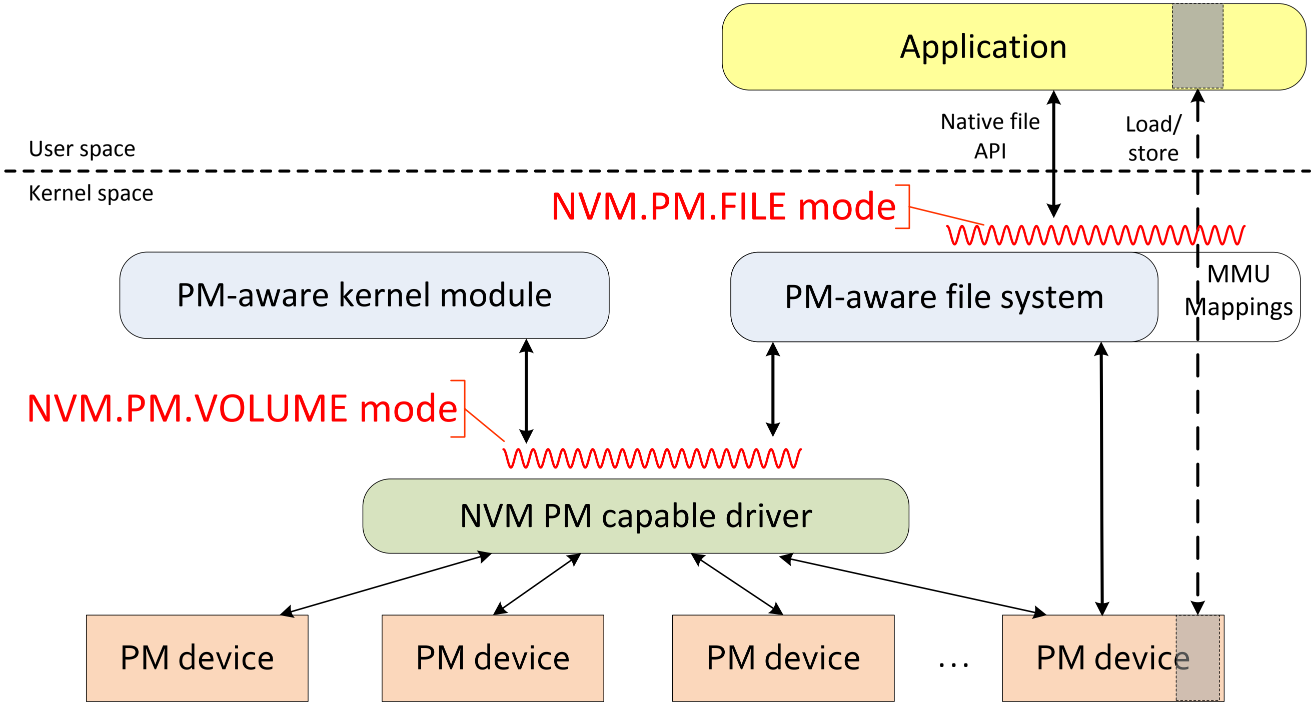
Persistent Memory Modes
Use with memory-like NVDIMMS

NVM.PM.VOLUME mode provides a software abstraction to OS components for Persistent Memory (PM) hardware:

- List of physical address ranges for each PM volume
- Thin provisioning management

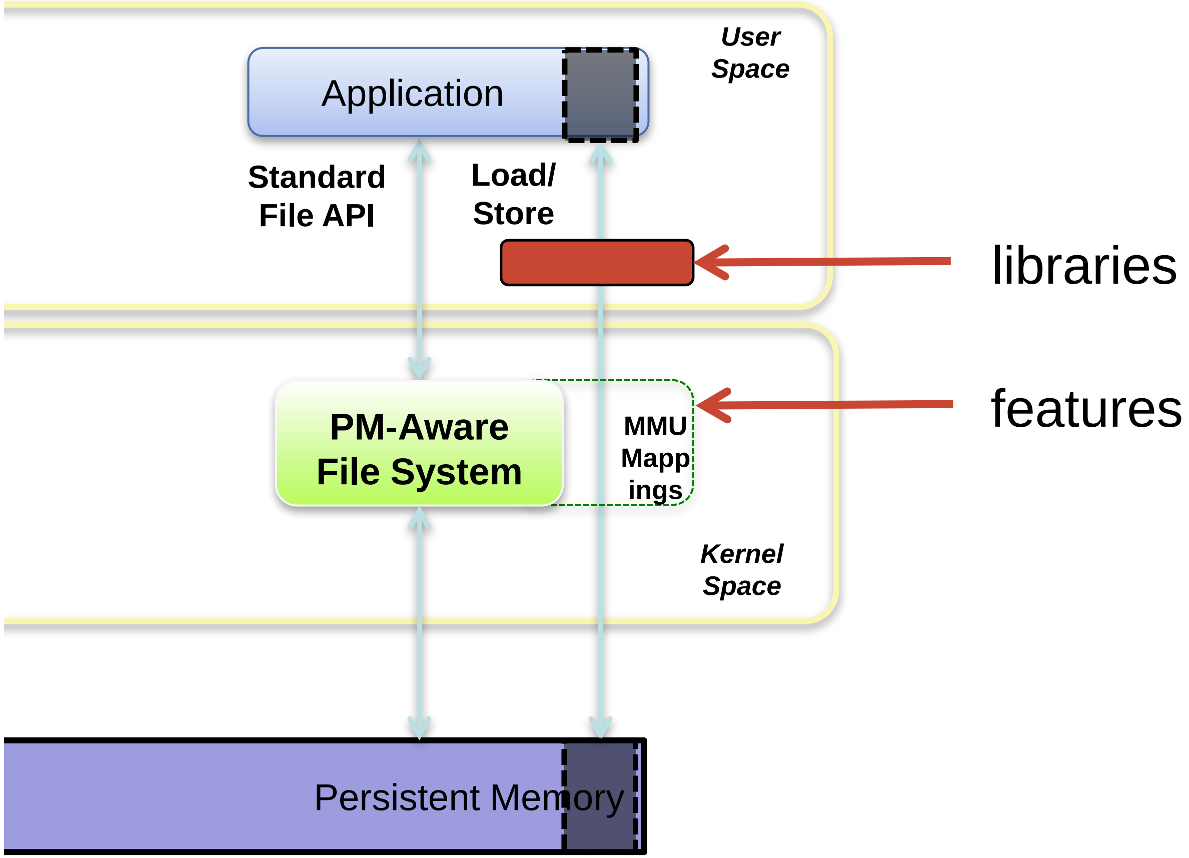
NVM.PM.FILE mode describes the behavior for applications accessing persistent memory including:

- mapping PM files (or subsets of files) to virtual memory addresses
- syncing portions of PM files to the persistence domain



Memory Mapping in NVM.PM.FILE mode enables direct access to persistent memory using CPU instructions

Building on the Basic PM Model



- NVM.PM.FILE programming model “surfaces” PM to application
- Refine presentation with additional libraries that evolve into language extensions
- Add compatible functionality to PM file systems

Examples of innovation enabled by the NVM programming model

- Under consideration within SNIA NVM Programming TWG
  - Atomic transactional behavior
    - Add atomicity and recovery to programming model
    - Not addressed by current sync semantics
  - Remote access
    - Disaggregated memory
    - Fabic attached NVM
    - High availability, clustering, capacity expansion use cases
- Open source contributions
  - Linux PMFS at https://github.com/linux-pmfs
  - Linux Pmem Examples: https://github.com/pmem/linux-examples

Start with NVDIMMs to enter a growing field of HW and SW technologies and solutions

Summary

- The NVM Programming Model is perfect for NVDIMMs
  - Block and File mode atomicity features
  - PM Mode memory mapped storage
  - http://snia.org/forums/sssi/nvmp

- Use the NVM programming model with NVDIMMS to enable a path forward for applications that leads to industry wide innovation in NVM optimized software

Thank You