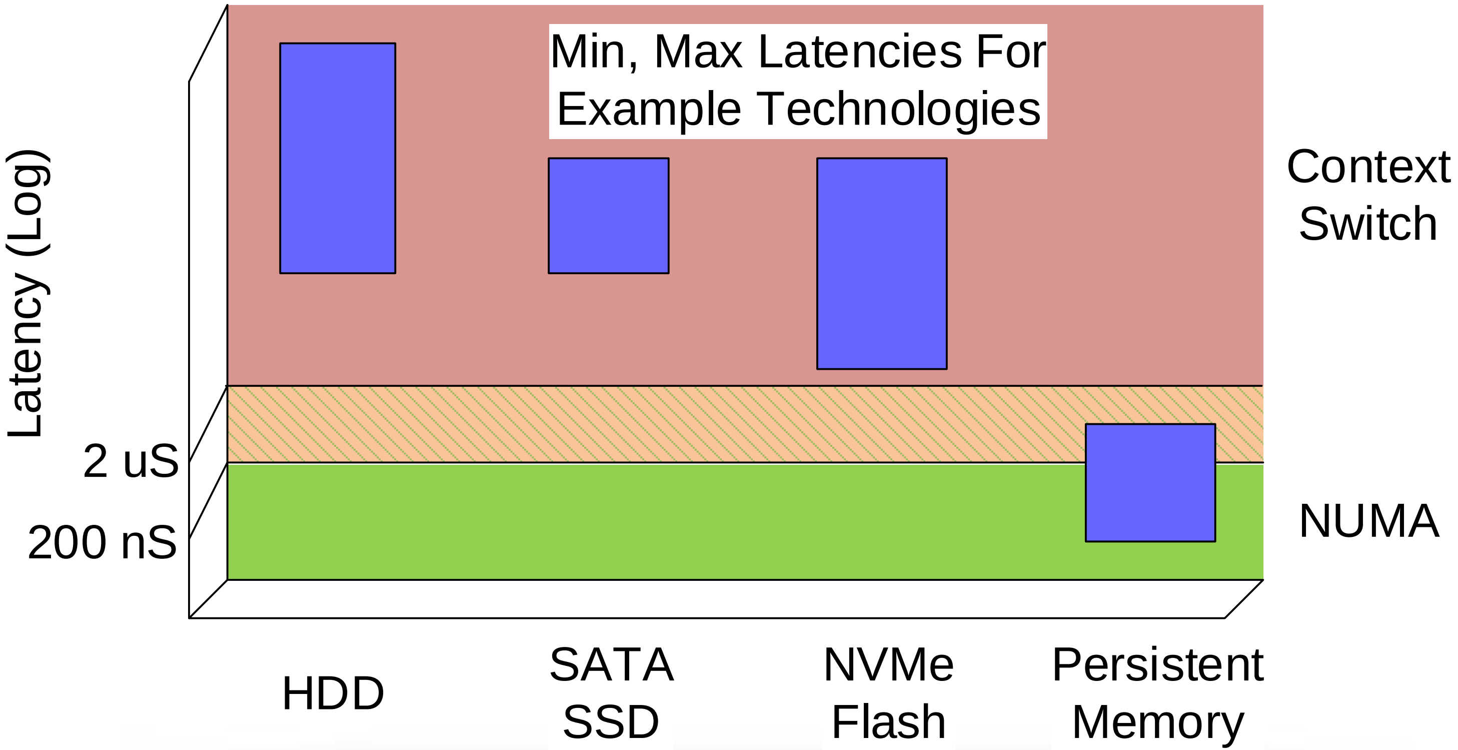
# The SNIA NVM Programming Model

Doug Voigt
Hewlett Packard Enterprise

# Contents

- What and Why
- Implications of the NVM Programming Model
  - Map and Sync
  - Pointers
  - Atomicity
  - Exception Handling
- Ongoing Work
  - Persistent Memory Data Structures
  - High Availability

# Latency Thresholds Cause Disruption



# Persistent Memory (PM) is a type of Non-Volatile Memory (NVM)

- Disk-like non-volatile memory
  - Appears as disk drives to applications
  - Accessed as traditional array of blocks
- Memory-like non-volatile memory (PM)
  - Appears as memory to applications
  - Applications store data directly in byte-addressable memory
  - No IO or even DMA is required

# The SNIA NVM Programming Model

- Version 1.1 approved by SNIA in March 2015
  - http://www.snia.org/tech_activities/standards/curr_standards/npm
- 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
- Programming Model, not API
  - Described in terms of attributes, actions and use cases
  - Implementations map actions and attributes to API's

# Programming Model Modes

- Block and File modes use IO
  - Data is read or written using RAM buffers
  - Software controls how to wait (context switch or poll)
  - Status is explicitly checked by software
- Volume and PM modes enable Ld/St
  - Data is loaded into or stored from processor registers
  - Processor makes software wait for data during instruction
  - No status checking – errors generate exceptions

# Block Access NVM

No Application Functionality Change

# Implications of the NVM Programming Model for Persistent Memory Applications

# Persistent Memory Modes

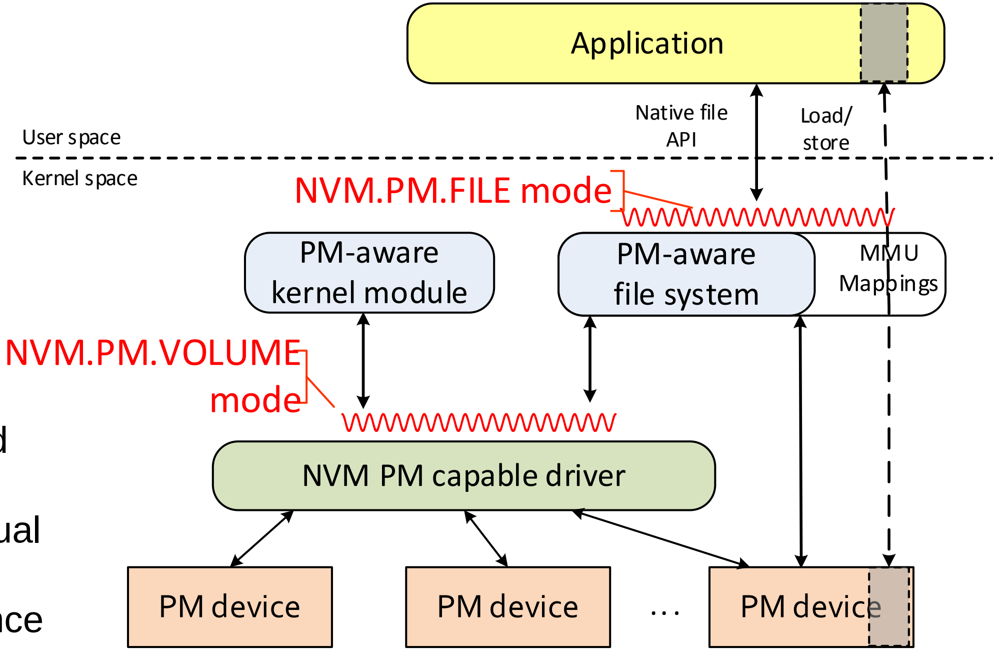
## Use with memory-like NVM

### 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



# Map and Sync

- 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
- Warning!  Sync does not guarantee order
  - Parts of CPU cache may be flushed out of order
  - This may occur before the sync action is taken by the application
  - Sync only guarantees that all data in the indicated range has been flushed some time before the sync completes

# PM Pointers

How can one persistent memory mapped data structure refer to another?

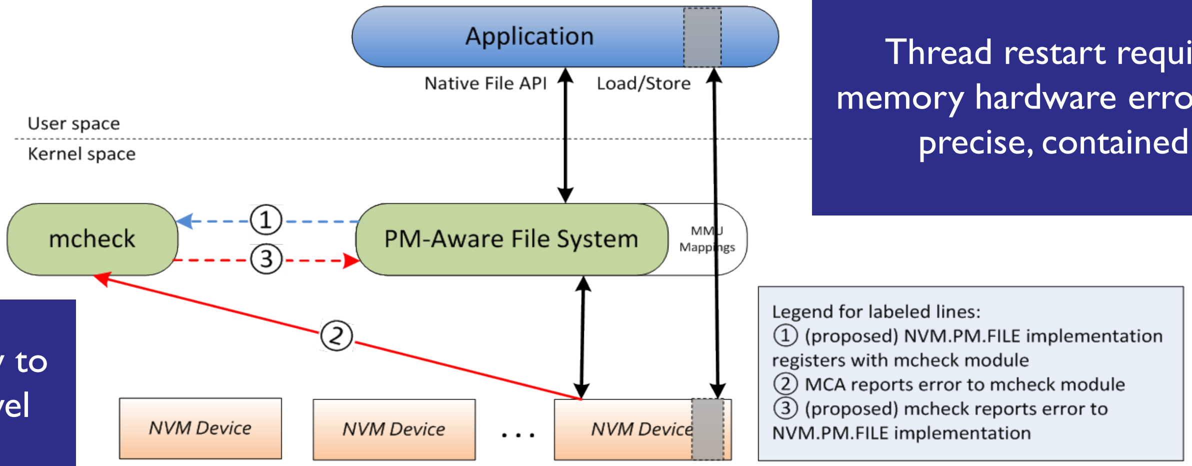
- Use its virtual address as a pointer
  - Assumes it will get the same address every time it is memory mapped
  - Requires special virtual address space management
- Use an offset from a relocatable base
  - Base could be the start of the memory mapped file
  - Pointer includes namespace reference

# Failure Atomicity

- Current processor + memory systems
  - Guarantee inter-process consistency (SMP)
  - But only provide limited atomicity with respect to failure
    - System reset/restart/crash
    - Power Failure
    - Memory Failure
- Failure atomicity is processor architecture specific
  - Processors provide failure atomicity of aligned fundamental data types
  - Fundamental data types include pointers and integers
  - PM programs use these to create larger atomic updates or transactions
  - Fallback is an additional checksum or CRC

# Error handling – exceptions instead of status

Figure 1 Linux Machine Check error flow with proposed new interface



Legend for labeled lines:
① (proposed) NVM.PM.FILE implementation registers with mcheck module
② MCA reports error to mcheck module
③ (proposed) mcheck reports error to NVM.PM.FILE implementation

Thread restart required unless memory hardware error detection is precise, contained and live

New machine check flow to integrate file and PM level recovery

Contained: exact memory location(s) are identified
Precise: instruction execution can be resumed (RTI)
Live: reported without restart

Application gets exception if file level recovery fails or backtracking is needed

# Ongoing Work

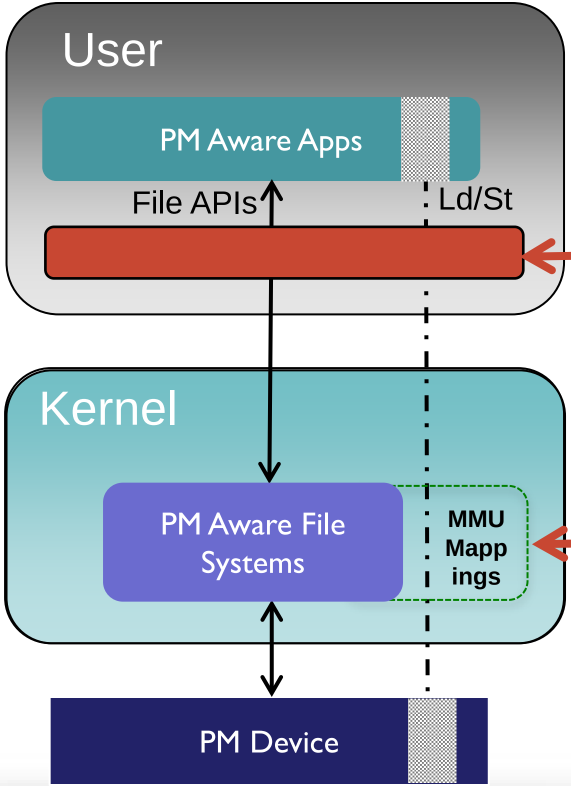
# Recent NVMP TWG Work in Progress

- Atomicity White Paper in Final Review

  Transactional PM Libraries

- Remote Access for HA White Paper Published

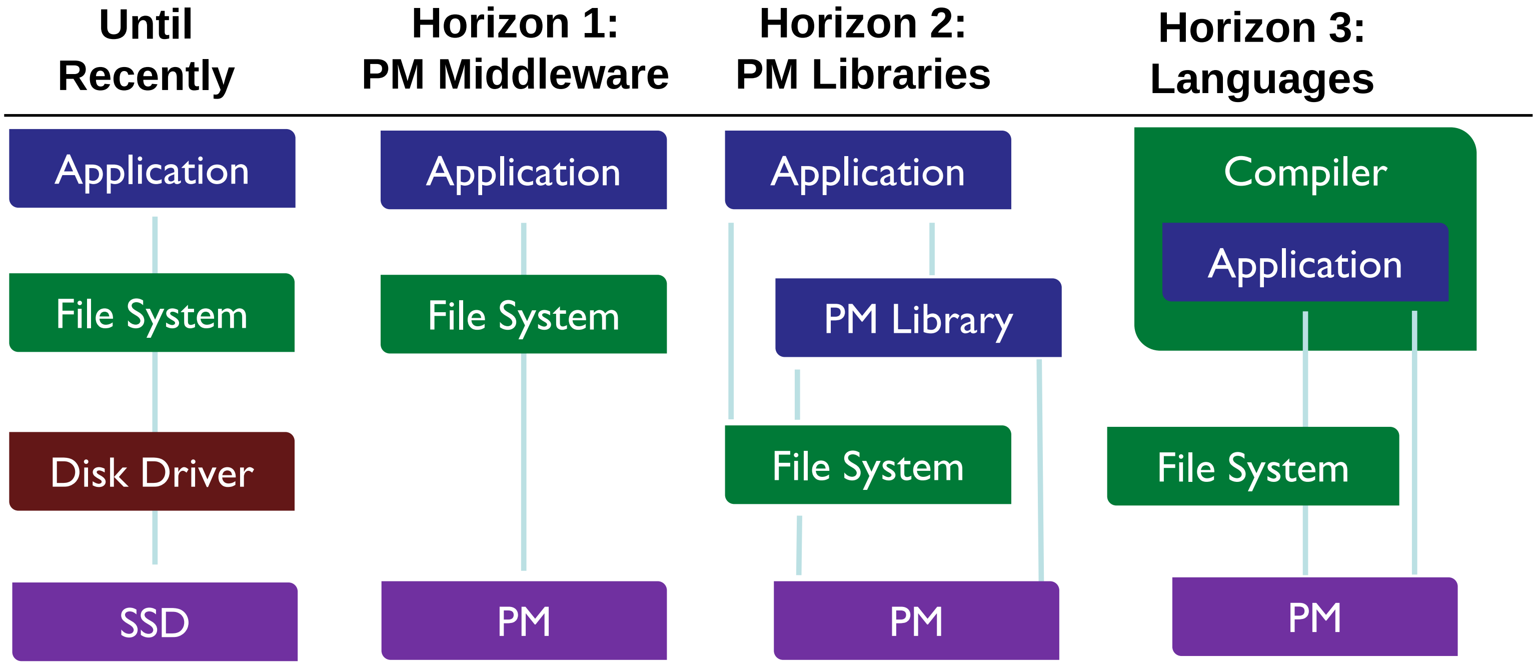
  High Availability PM - Remote Optimized Flush



PM data structure libraries

Middleware features e.g. RAID

# Application Horizons



# Atomicity

- Inter-Process Atomicity vs. Failure Atomicity
- References NVM Library
- PM Data Structures with Intrinsic atomicity
  - Log, Block and more
- Transactions for more complex atomicity
  - Multiple Data Structures

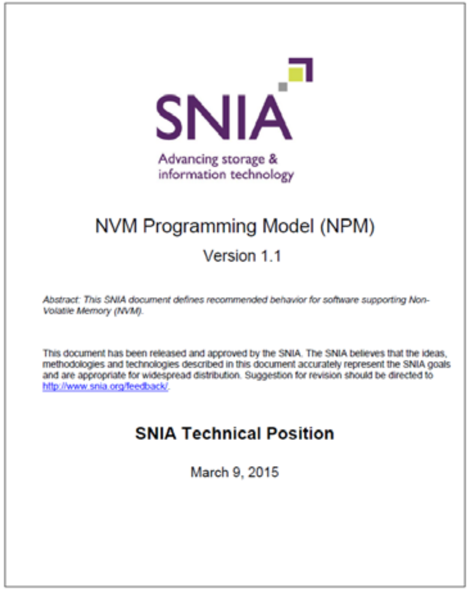
# Remote Access

- SNIA PM Remote Access for High Availability
  - Remote access taxonomy
  - Data recoverability requirements
  - Model and requirements for remote flush
- Multiple industry parties are responding
  - Open Fabrics Alliance
  - InfiniBand Trade Association
  - Several vendors

# Role of the NVM Programming Model

Rally the industry around a view of NVM that is:

- Application centric
- Vendor neutral
- Achievable today
- Beyond storage
  - Applications
  - Memory
  - Networking
  - Processors



SNIA
Advancing storage & information technology

NVM Programming Model (NPM)

Version 1.1

*Abstract: This SNIA document defines recommended behavior for software supporting Non-Volatile Memory (NVM).*

This document has been released and approved by the SNIA. The SNIA believes that the ideas, methodologies and technologies described in this document accurately represent the SNIA goals and are appropriate for widespread distribution. Suggestion for revision should be directed to http://www.snia.org/feedback/.

**SNIA Technical Position**

March 9, 2015

# Thank You