Paving the Way to the Non-Volatile Memory Frontier

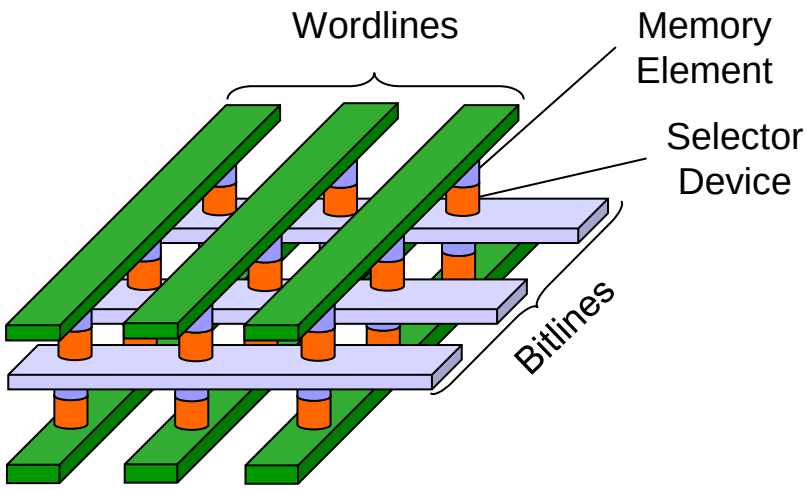
Doug Voigt

HP

What is happening to the relationship between storage and memory?

Next Generation Scalable NVM

Scalable Resistive Memory Element



Cross Point Array in Backend Layers

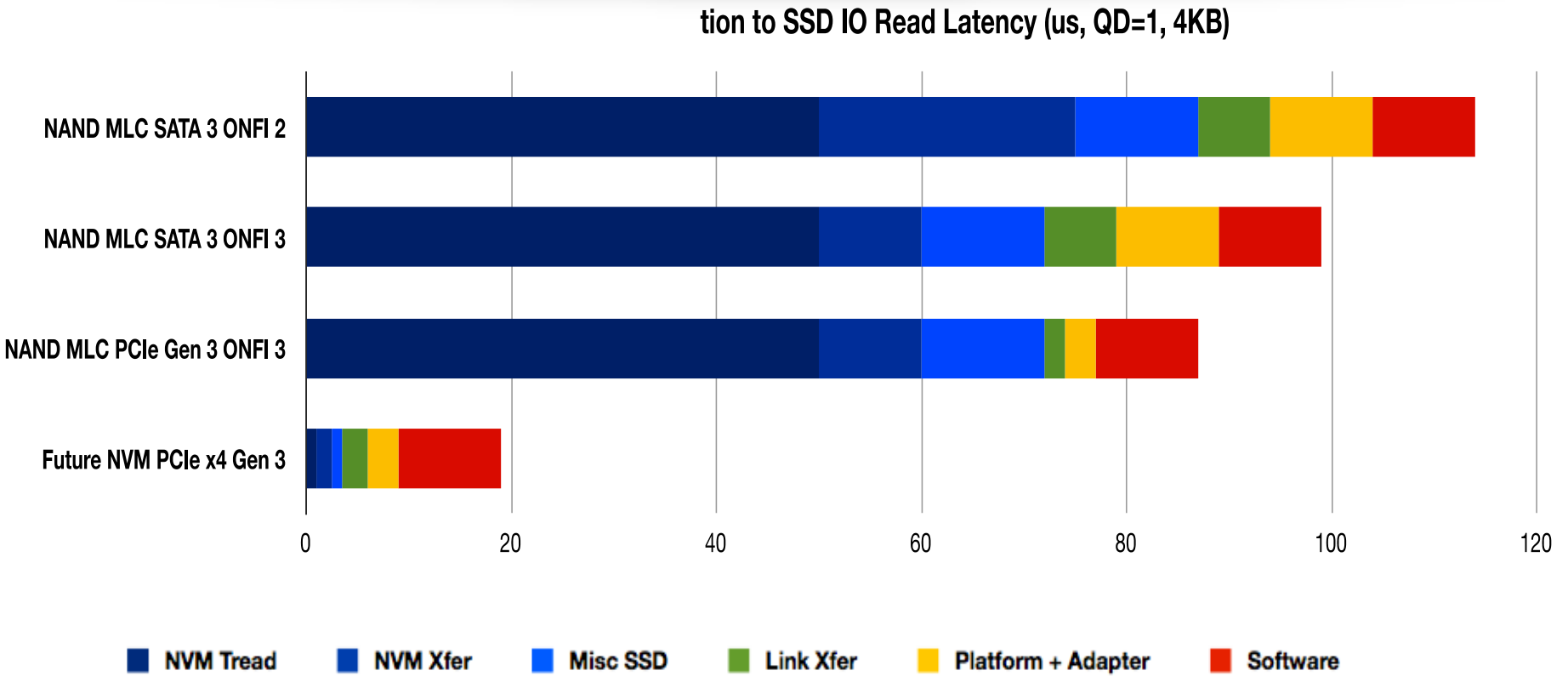
Resistive RAM NVM Options

Family	Defining Switching Characteristics
Phase Change Memory	Energy (heat) converts material between crystalline (conductive) and amorphous (resistive) phases
Magnetic Tunnel Junction (MTJ)	Switching of magnetic resistive layer by spin-polarized electrons
Electrochemical Cells (ECM)	Formation / dissolution of "nano-bridge" by electrochemistry
Binary Oxide Filament Cells	Reversible filament formation by Oxidation-Reduction
Interfacial Switching	Oxygen vacancy drift diffusion induced barrier modulation

~ 1000x speed-up over NAND.

From Jim Pappas, Intel, SNIA NVM Summit

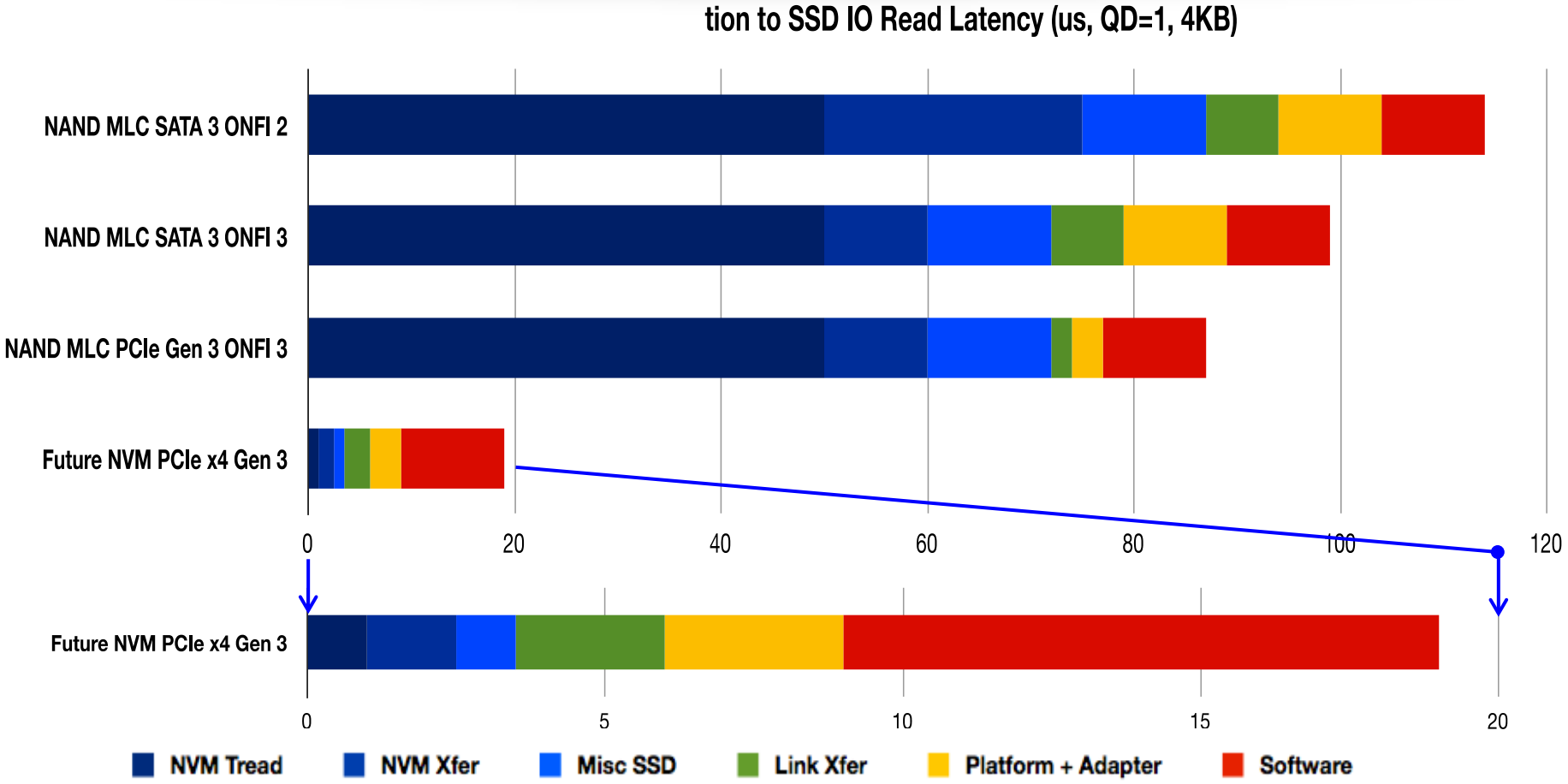
Storage Latency is Changing



SSD Latency Profile – In the next generation, media is not the bottleneck.

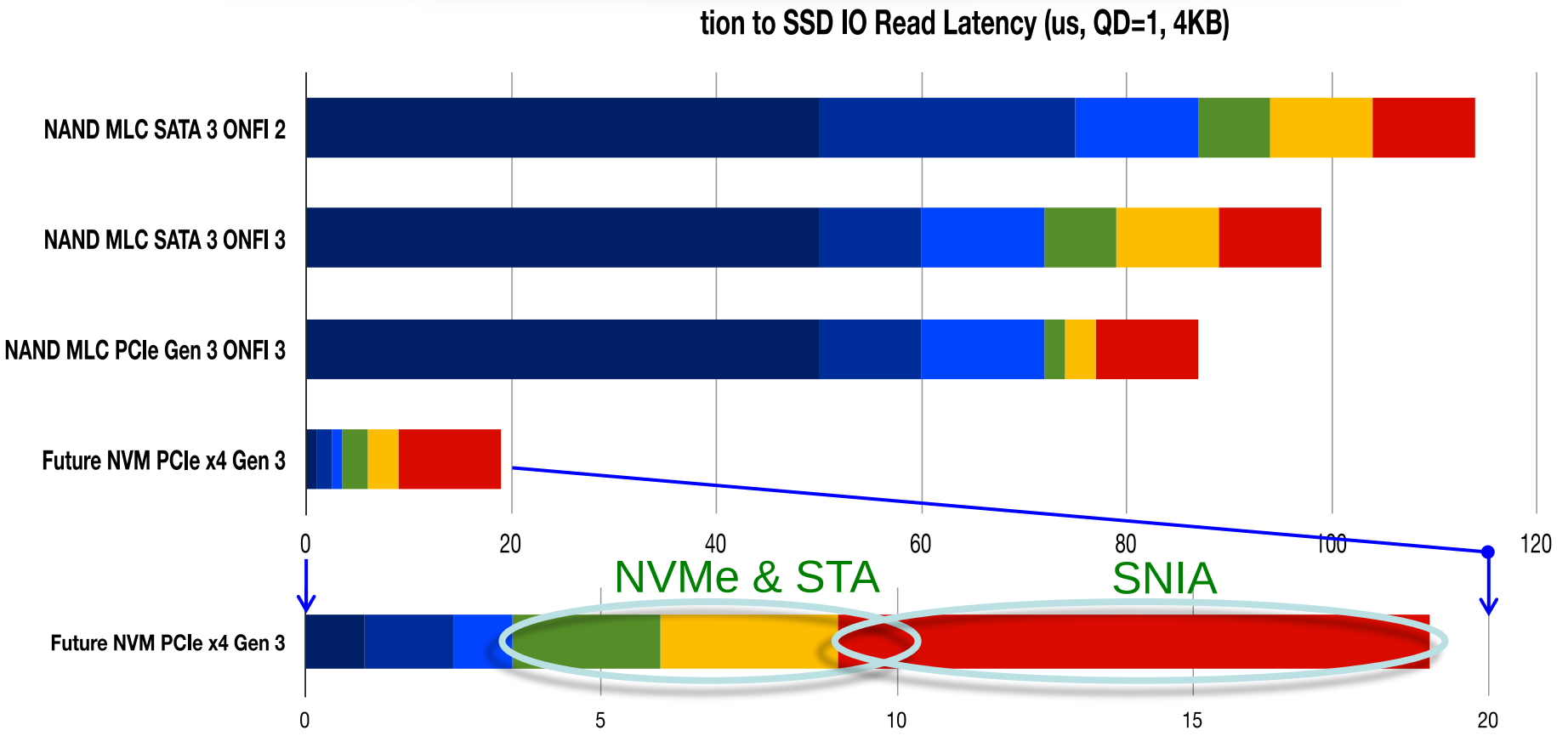
From Jim Pappas, Intel, SNIA NVM Summit

Interconnect and Software Will Soon Dominate Latency



From Jim Pappas, Intel, SNIA NVM Summit

Opportunities with Next Generation NVM

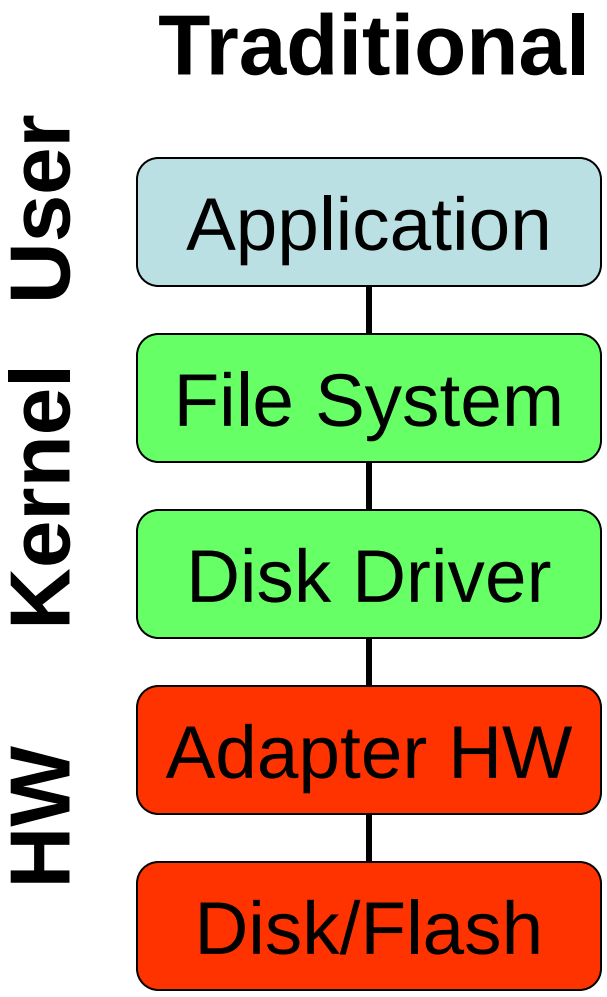


NVM Express/SCSI Express: Optimized storage interconnect & driver
SNIA NVM Programming TWG: Optimized system & application software

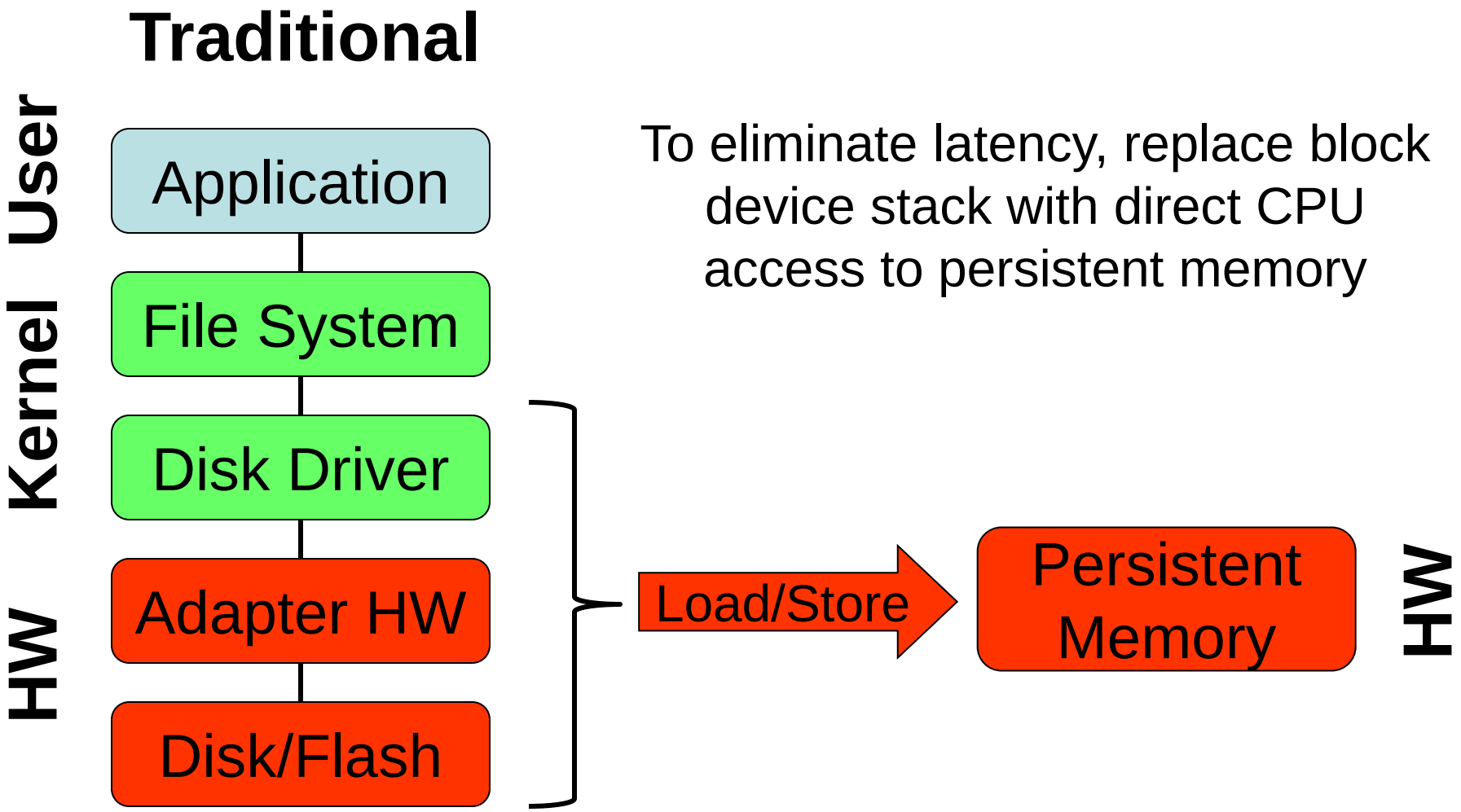
From Jim Pappas, Intel, SNIA NVM Summit

What is SNIA doing about this?

Today's Storage Stack



Replace Application IO with Load and Store Instructions



To eliminate latency, replace block device stack with direct CPU access to persistent memory

Load/Store (Ld/St) Access

- Load and Store are references to CPU instructions
  - Shorthand for all operands that cause memory access
  - Includes various memory move instructions
  - Manipulate byte (not block) granularity data
  - Actual instructions are CPU architecture specific
- How are they used?
  - Generated by compilers and interpreters
  - Get application data in and out of CPU registers
- Application sees PM as persistent data structures
- Ld/St is 100-1000 times faster than IO

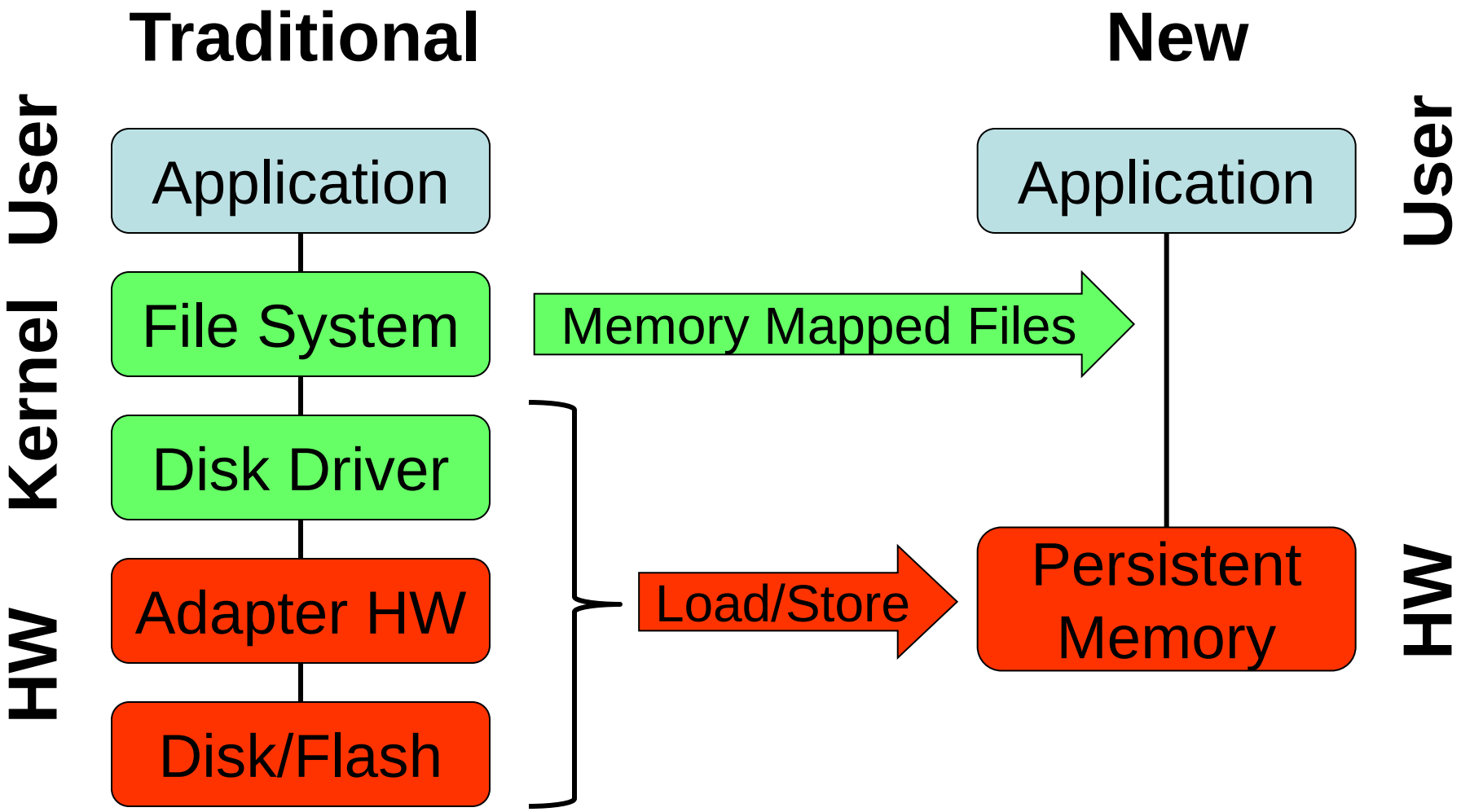
Persistent Memory (PM)

- Is
  - Memory speed (stalls CPU loads)
  - Byte addressable from programmer's point of view
  - Load/Store (Ld/St), DMA and RDMA access
- Is Not
  - Prone to unexpected latencies
    - Demand paging
    - Page Caching
  - Durable until data is flushed
  - Flash (at least not directly)
  - Tablet-like memory for the entire system

Examples of PM

- Battery-backed DRAM
  - Practical or not, a good model for software design
- DRAM saved in the event of power failure
  - DRAM speeds at run-time
  - Expense = DRAM + additional logic
- NVM with significant caching
  - Reliance on cache involves workload assumptions
- Next generation NVM
  - Bits and pieces of information available
  - Emerging technologies mean some unknowns

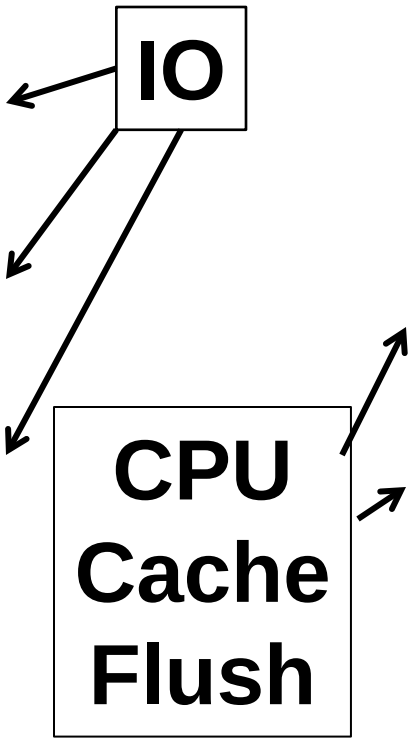
Eliminate File System Latency with Memory Mapped Files



Resulting Application Change

Traditional

Open("file name");
Read(mem range,offset);
<compute>
Write(mem range,offset);
<compute>
Write(mem range,offset);
Close();



Memory Mapped

Open("file name");
Map(offset, PM range);
<compute>
Sync(PM range);
<compute>
Sync(PM range);
Unmap();
Close();

Challenges of Persistent Memory

- New Programming Models
  - Built on existing models
  - Implemented using new APIs
- Some Programming Challenges
  - File system and database engineers have always done this
  - Now other application engineers need to learn it

The SNIA NVM Programming Model

SNIA NVM Programming TWG Members

EMC, Fujitsu, Fusion-io, HP, HGST, Inphi, Intel, Intuitive Cognition Consulting, LSI, Microsoft, NetApp,PMC-Sierra, Qlogic, Red Hat, Samsung, SanDisk, Seagate, Sony, Symantec, Viking, Virident, Vmware

Calypso Systems, Cisco, Contour Asset Management, Data Storage Institute, Dell, FalconStor, Hitachi, Huawei, IBM, Integrated Device Technology,IDT, Marvell, Micron, NEC, OCZ, Oracle, Reduxio, SK Hynix,Tata Consultancy Services, Toshiba, Wipro

NVM Programming TWG

- Charter:

  Develop specifications for new software “programming models” as NVM becomes a standard feature of platforms

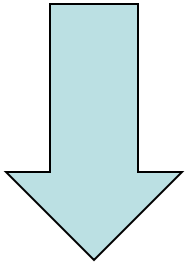
- Scope:
  - In-kernel NVM programming models
  - Kernel-to-application programming models

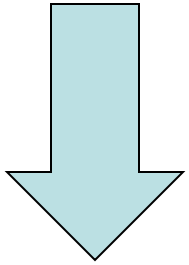
Programming Model vs API

- Definitions
  - Programming model: exact technical behavior specification, up to (but not including) OS specific APIs
  - API: Code created by an OSV to implement the programming models on a specific OS
- Why a programming model?
  - Committees cannot push APIs on OSVs
  - Different OS's require different API styles
  - "When in Rome" principle

Two Movements Afoot

Basic programming model is decades old!





Block Mode Innovation

- Atomics
- Access hints
- NVM-oriented operations

Emerging NVM Technologies

- Performance
- Performance
- Perf… okay, cost

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

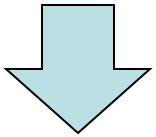
The Four Modes

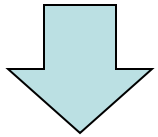
Block Mode Innovation

- Atomics
- Access hints
- NVM-oriented operations

Emerging NVM Technologies

- Performance
- Performance
- Perf… okay, cost





	Traditional	Persistent Memory
Kernel Protected	NVM.BLOCK	NVM.PM.VOLUME
User View	NVM.FILE	NVM.PM.FILE

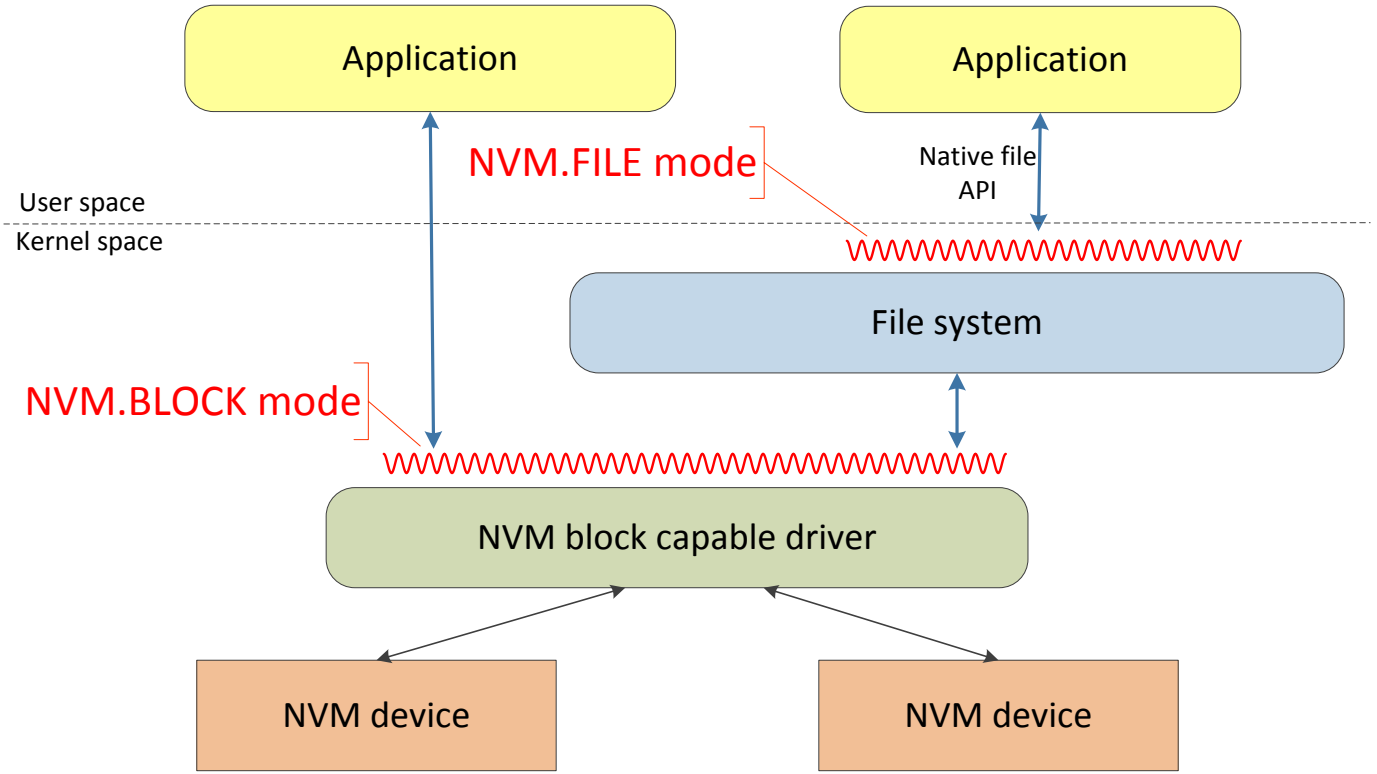
Conventional Block and File Modes

BLOCK mode describes extensions:

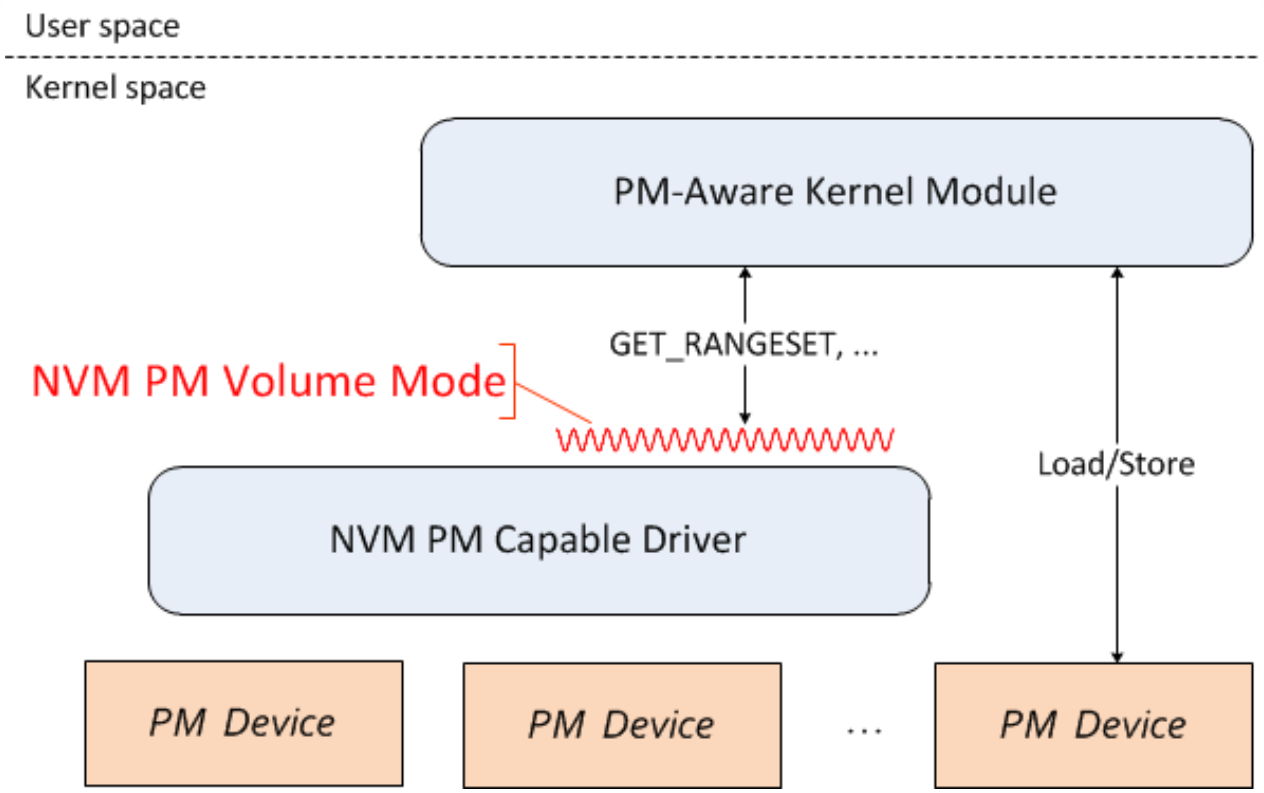
- Atomic write features
- Granularities (length, alignment)
- Thin Provisioning Management

FILE mode describes extensions:

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



Persistent Memory Modes
NVM.PM.VOLUME



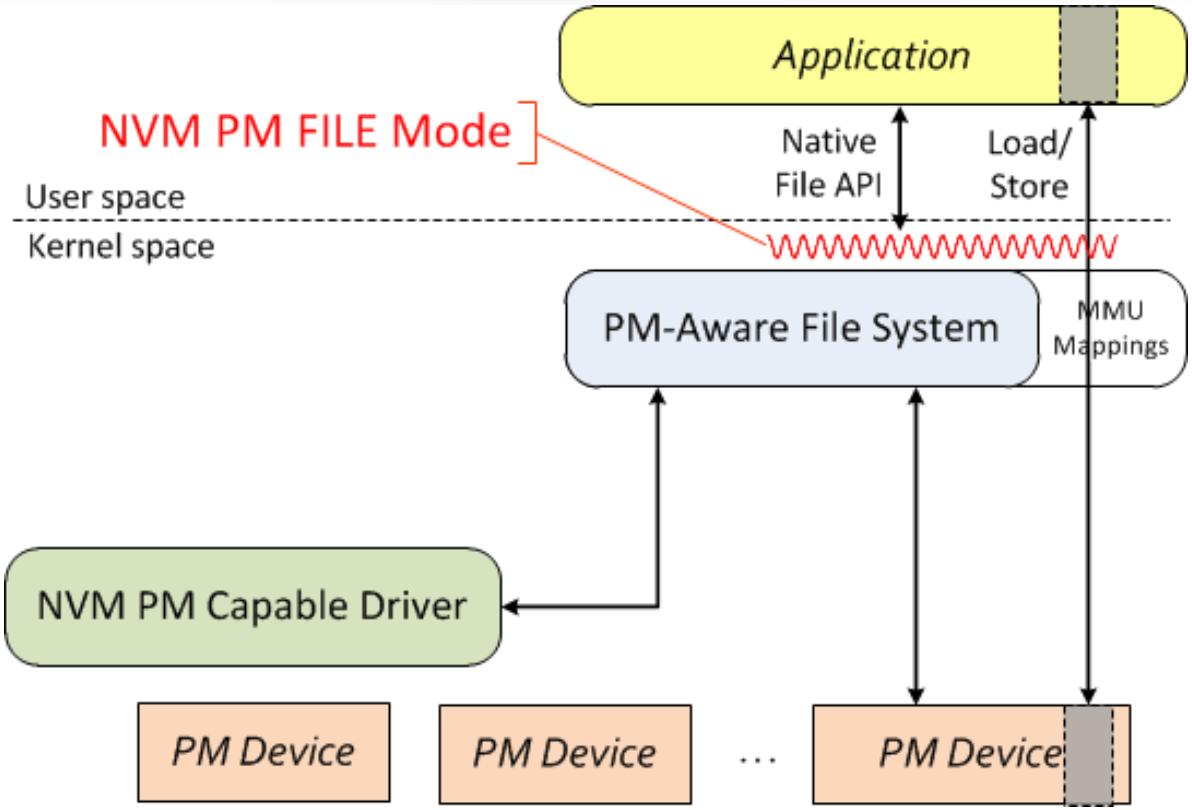
NVM.PM.VOLUME mode provides a software abstraction to OS components for Persistent Memory (PM) hardware:

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

Who Uses NVM.PM.Volume?

- Kernel modules
- File systems
  - Maybe to expose PM
  - Maybe just to use it internally
- Memory management
  - Example: Multi-tiered page cache
- Other storage stack components
  - RAID
  - Caches
  - Clustered I/O
- Future NVM Programming models we haven't thought of

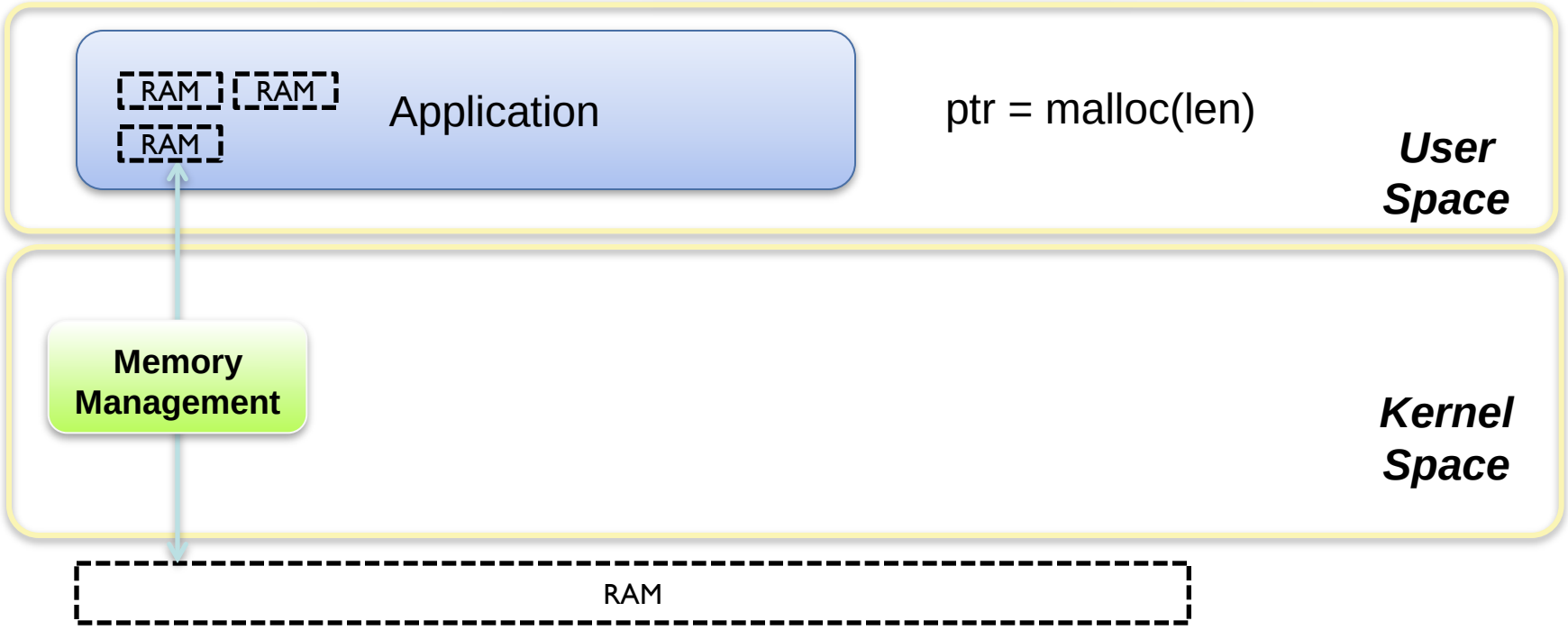
Persistent Memory Modes
NVM.PM.FILE



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

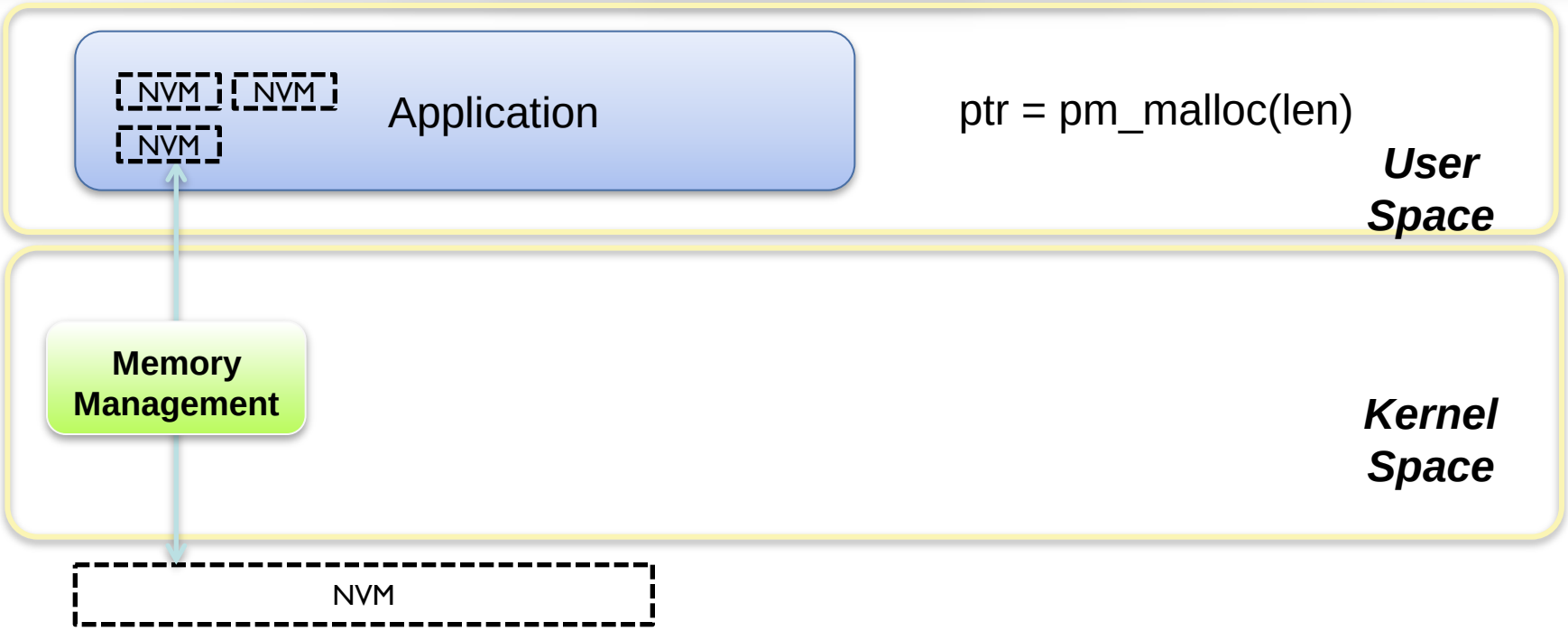
- mapping PM files (or subsets of files) to virtual memory addresses
- syncing portions of PM files to the persistence domain

Application Memory Allocation



- Well-worn interface, around for decades
- Memory is gone when application exits
  - Or machine goes down

Application PM Allocation



- Simple, familiar interface, but then what?
  - Persistent, so apps want to “attach” to regions
  - Need to manage permissions for regions
  - Need to resize, remove, …, backup the data

Who Uses NVM.PM.FILE?

- Applications
  - Persistent data sets requiring addressability without impacting DRAM footprint
  - Persistent caches
- Usages that must reconnect with blobs of persistence
  - Naming
  - Permissions
- Potentially kernel modules requiring some of the above features

Beyond V1: Investigating New Work Items

SNIA DSI '14 DATA STORAGE INNOVATION CONFERENCE

Three new work items are under investigation

1) Software hints

- Application usage, access patterns
- Optimization based on discovered device attributes
- Present hints emerging in standards (SCSI, NVMe) to applications

2) Atomic transactional behavior

- Add atomicity and recovery to programming model
- Not addressed by current sync semantics

3) Remote access

- Disaggregated memory
- RDMA direct to NVM
- High availability, clustering, capacity expansion use cases

Summary

- The NVM Programming Model is aligning the industry
  - Gaining common terminology
  - Not forcing specific APIs
  - http://snia.org/forums/sssi/nvmp
- What are we doing with it?
  - PM models expose it
    - Linux PMFS at https://github.com/linux-pmfs
  - New PM models build on existing ones
    - Linux Pmem Examples: https://github.com/pmem/linux-examples
    - New TWG work items
- Emerging technologies will drive increasing work in this area as cost comes down

Thank You