SDC 18
SNIA EMEA

FEBRUARY 2018
TEL AVIV, ISRAEL

STORAGE DEVELOPER
CONFERENCE

Persistent Memory
what developers need to know

Mark Carlson
Co-chair SNIA Technical Council
Toshiba

Contents

- ❒ Welcome
- ❒ Persistent Memory Overview
- ❒ Non-Volatile Memory (NVM) Programming Model
- ❒ Non-Volatile DIMM (NVDIMM)

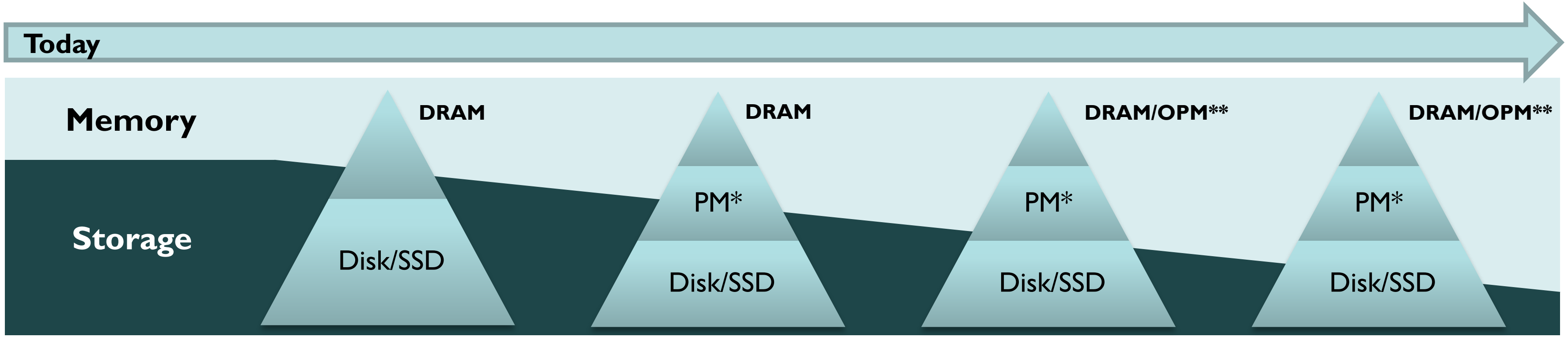
SNIA Legal Notice

- The material contained in this presentation is copyrighted by the SNIA unless otherwise noted.
- Member companies and individual members may use this material in presentations and literature under the following conditions:
  - Any slide or slides used must be reproduced in their entirety without modification
  - The SNIA must be acknowledged as the source of any material used in the body of any document containing material from these presentations
- This presentation is a project of the SNIA.
- Neither the authors nor the presenters are attorneys and nothing in this presentation is intended to be, or should be, construed as legal advice or an opinion of counsel. If you need legal advice or a legal opinion, please contact your attorney.
- The information presented herein represents the author(s) personal opinion and current understanding of the relevant issues involved. The authors, presenters, and the SNIA do not assume any responsibility or liability for damages arising out of any reliance on or use of this information.

NO WARRANTIES, EXPRESS OR IMPLIED. USE AT YOUR OWN RISK.

The Trend: Memory & Storage Convergence

- Volatile and non-volatile technologies are continuing to converge



*PM = Persistent Memory

**OPM = On-Package Memory

New and Emerging Memory Technologies

HMC

HBM

RRAM

3DXPoint™ Memory

MRAM

PCM

Low Latency NAND

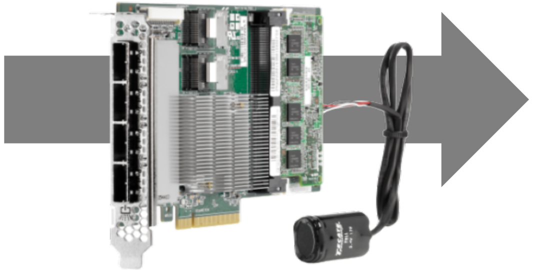
Managed DRAM

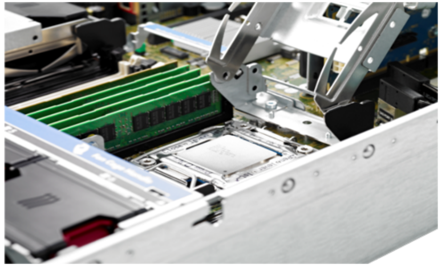
SDC 18

Source: Gen-Z Consortium 2016

Persistent Memory (PM) Vision

Persistent Memory Brings Storage





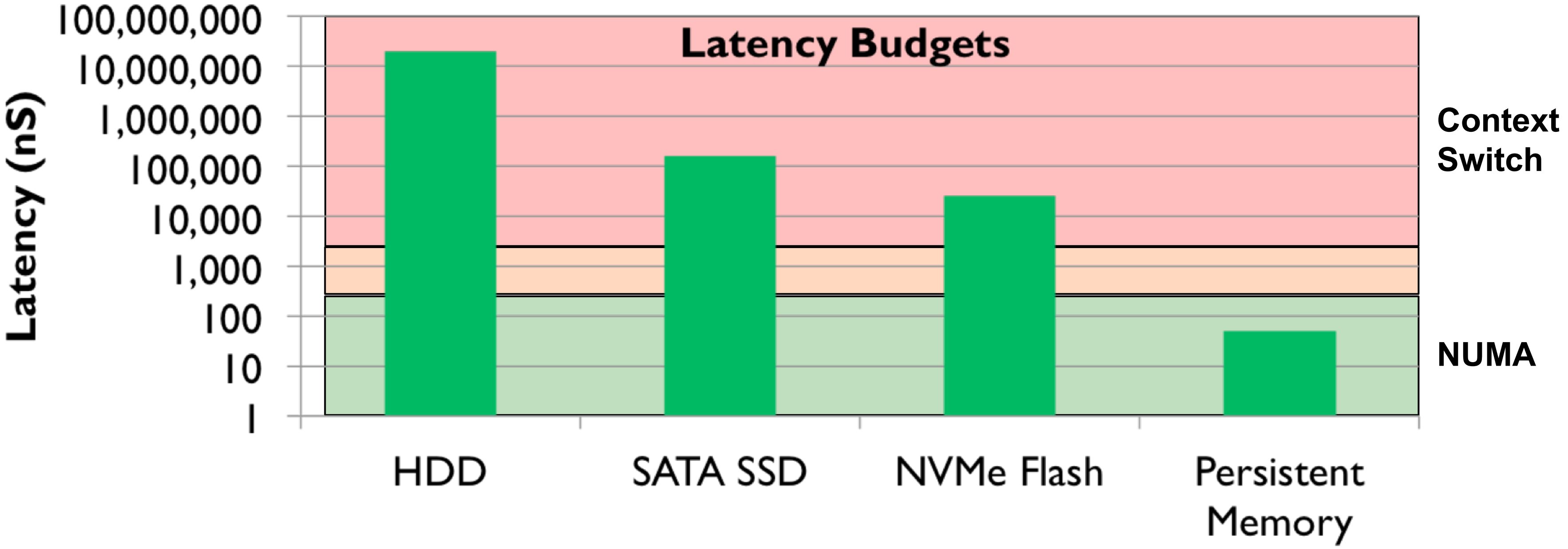
To Memory Slots

Fast
Like Memory

Persistent
Like Storage

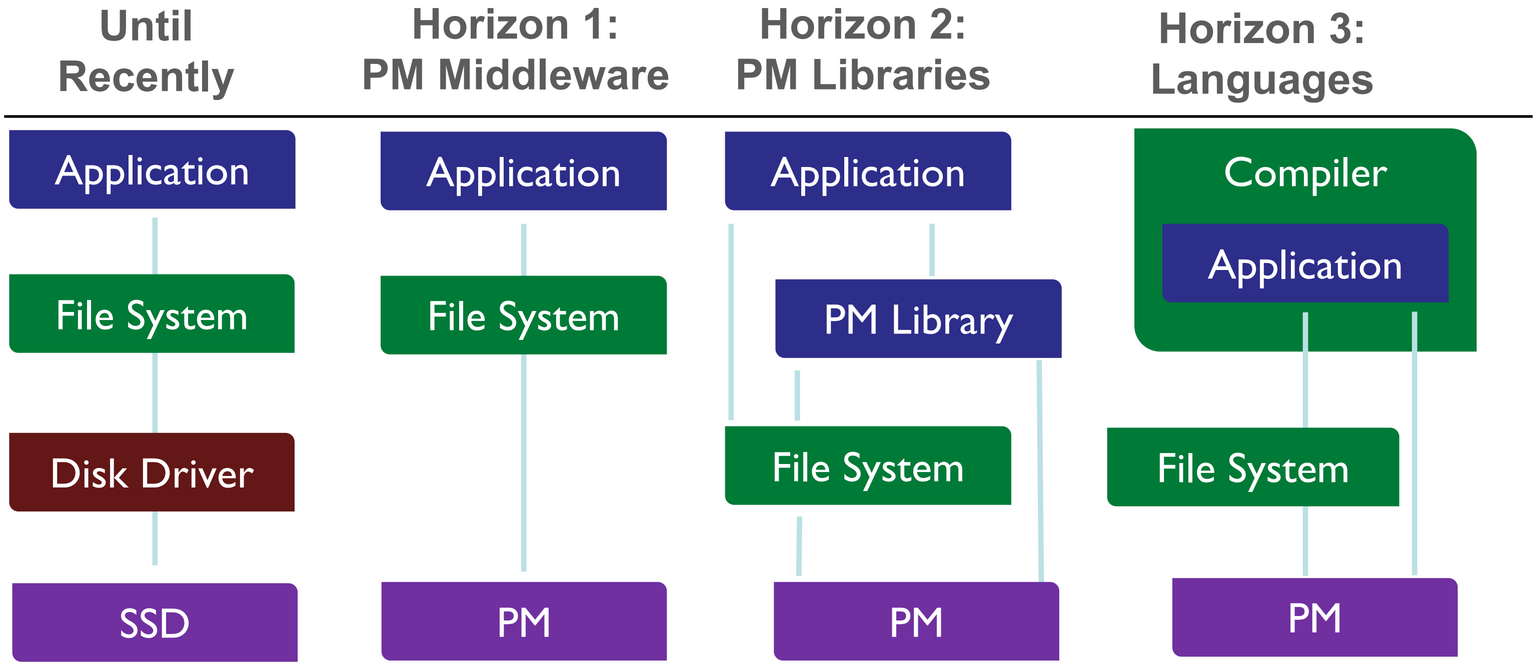
- For system acceleration
- For real-time data capture, analysis and intelligent response

Storage vs. Memory



Typical NUMA range: 0 - 200 nS
Typical context switch range: above 2-3 uS

Application Horizons



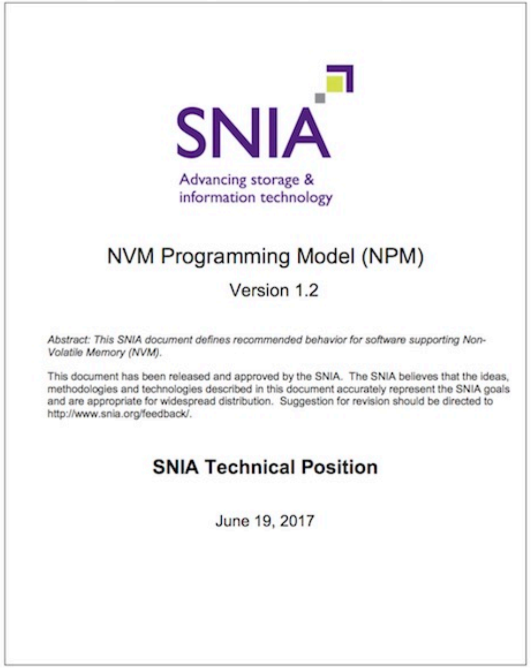
Persistent Memory (PM) Characteristics

- ❒ Byte addressable from programmer’s point of view
- ❒ Provides Load/Store access
- ❒ Has Memory-like performance
- ❒ Supports DMA including RDMA
- ❒ Not Prone to unexpected latencies associated with demand paging or page caching
- ❒ Think Power Protected RAM

NVM Programming Model – Writing Applications for Persistent Memory

Role of the NVM Programming Model

- Rally the industry around a view of Persistent Memory that is:
  - Application centric
  - Vendor neutral
  - Achievable today
  - Beyond storage
    - Applications
    - Memory
    - Networking/Fabrics
    - Processors



SNIA

Advancing storage & information technology

NVM Programming Model (NPM)

Version 1.2

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

June 19, 2017

NVM Programming Model TWG - Mission

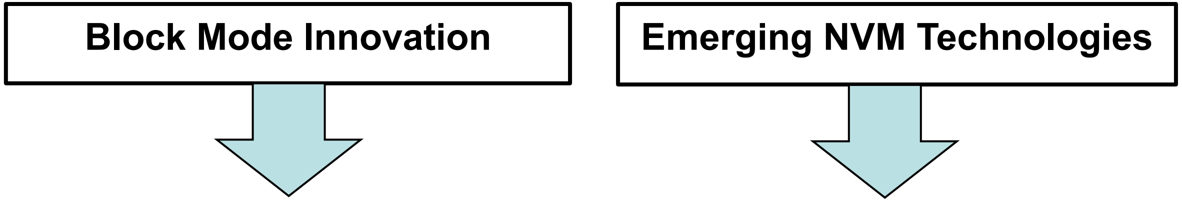
- ❐ Accelerate the availability of software that enables Persistent Memory hardware.
  - ❐ Hardware includes SSD’s and PM
  - ❐ Software spans applications (user-space) and OS’s (kernel-space)

- ❐ Create the NVM Programming Model
  - ❐ Describes application visible behaviors
  - ❐ Allows API’s to align with OS’s
  - ❐ Exposes opportunities in networks and processors

SNIA NVM Programming Model

- Version 1.2 approved by SNIA in June 2017
  - 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

The NVM Programming Model Has 4 Modes



Block Mode Innovation

Emerging NVM Technologies

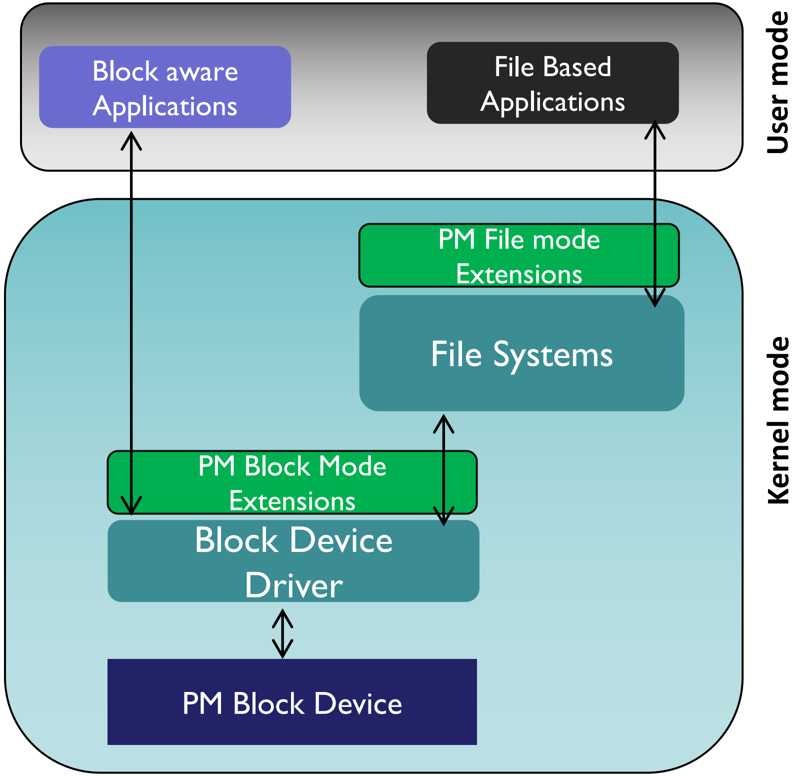
	IO	Persistent Memory
User View	**NVM.FILE**	**NVM.PM.FILE**
Kernel Protected	**NVM.BLOCK**	**NVM.PM.VOLUME**
Media Type	Disk Drive	Persistent Memory
NVDIMM	Disk-Like	Memory-Like

Programming Model Modes

- NVM.FILE and NVM.BLOCK 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
- NVM.PM.* (FILE and VOLUME) modes enable Load/Store
  - Data is loaded into or stored from processor registers
  - Processor makes software wait for data during instruction
  - No status checking – errors generate exceptions

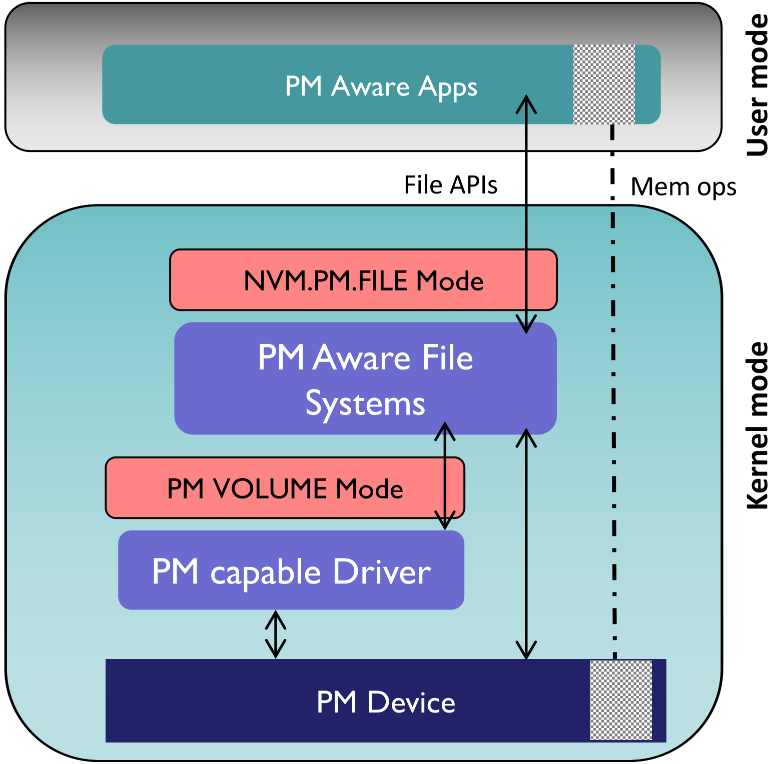
File and Block Mode Extensions

- NVM.BLOCK Mode
  - Targeted for file systems and block-aware applications
  - Atomic writes
  - Length and alignment granularities
  - Thin provisioning management
- NVM.FILE Mode
  - Targeted for file based apps.
  - Discovery and use of atomic write features
  - Discovery of granularities



Persistent Memory (PM) Modes

- NVM.PM.VOLUME Mode
  - Software abstraction for persistent memory hardware
  - Address ranges
  - Thin provisioning management
- NVM.PM.FILE Mode
  - Application behavior for accessing PM
  - Mapping PM files to application address space
  - Syncing PM files



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

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

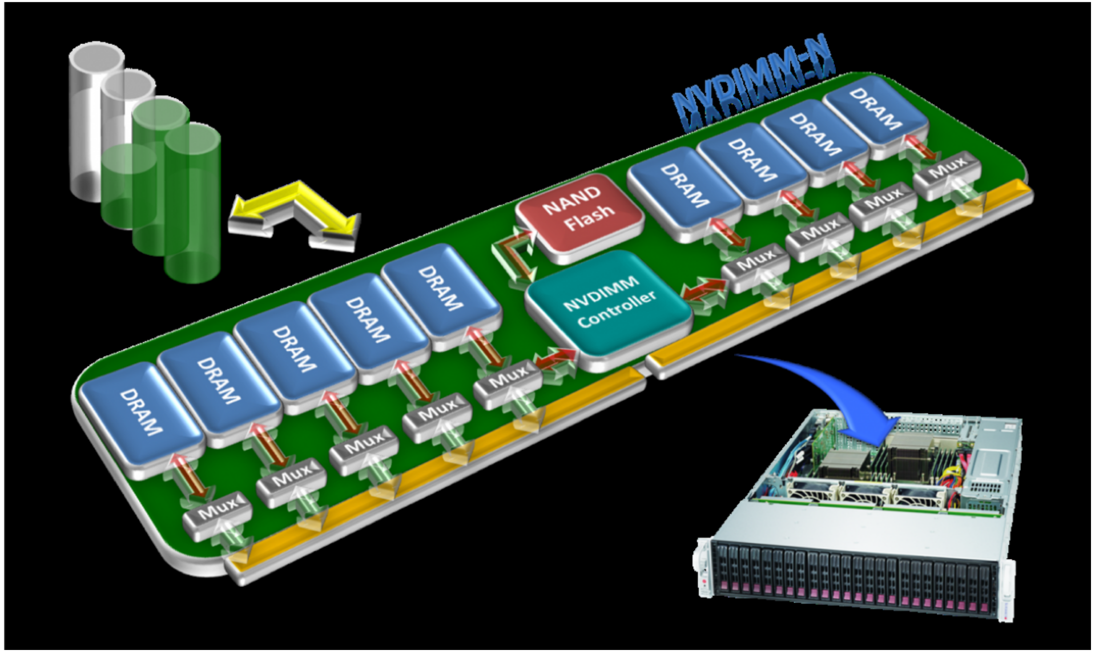
Ongoing SNIA NVMP TWG Work

- NVM Programming Model Specification
  - NVM Interfaces between OS components
  - Application Interfaces to NVM related OS, hypervisor and hardware components
- Remote Access for High Availability white paper
  - Create models and requirements for communication with remote persistent memory for the purpose of High Availability
- Asynchronous Flush, Persistence Failure
  - Describe considerations for supporting atomics and transactions using extensions to the NVM Programming Model Specification
- PM Security for Multi-Tenancy
  - Describe models for PM security when multiple tenants are present
  - See companion SNIA Storage Developer Conference talk
    - https://www.snia.org/events/storage-developer/presentations17

Summary

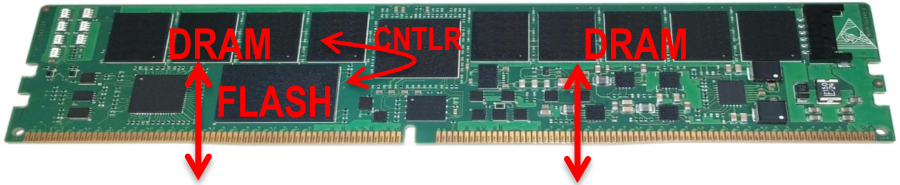
- The NVM Programming Model is aligning the industry (http://pmem.io/)
  - Gaining common terminology
  - Not forcing specific APIs
  - http://snia.org/forums/sssi/nvmp
- What are we doing with it?
  - PM models expose it
    - DAX-aware file-systems in Linux (see FS_DAX for more info)
  - New PM models build on existing ones
    - Linux Pmem Examples (see examples folder) https://github.com/pmem/nvml
    - New TWG work items
- Emerging technologies will drive increasing work in this area as cost comes down (e.g. materials and memory-centric fabrics)

NVDIMM Example



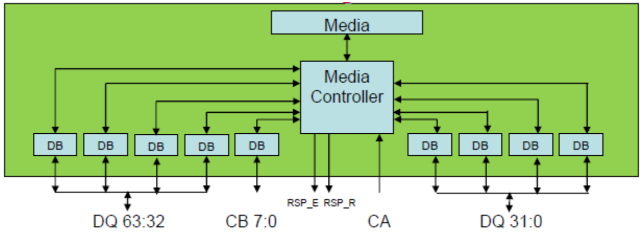
NVDIMM Types

NVDIMM-N



- Host has direct access to DRAM
- CNTLR moves DRAM data to Flash on power fail
- Requires backup power (typically 10’s of seconds)
- CNTLR restores DRAM data from Flash on next boot
- Communication through SMBus (JEDEC standard)
- Byte-addressable DRAM for lowest latency with NAND for persistence backup

NVDIMM-P



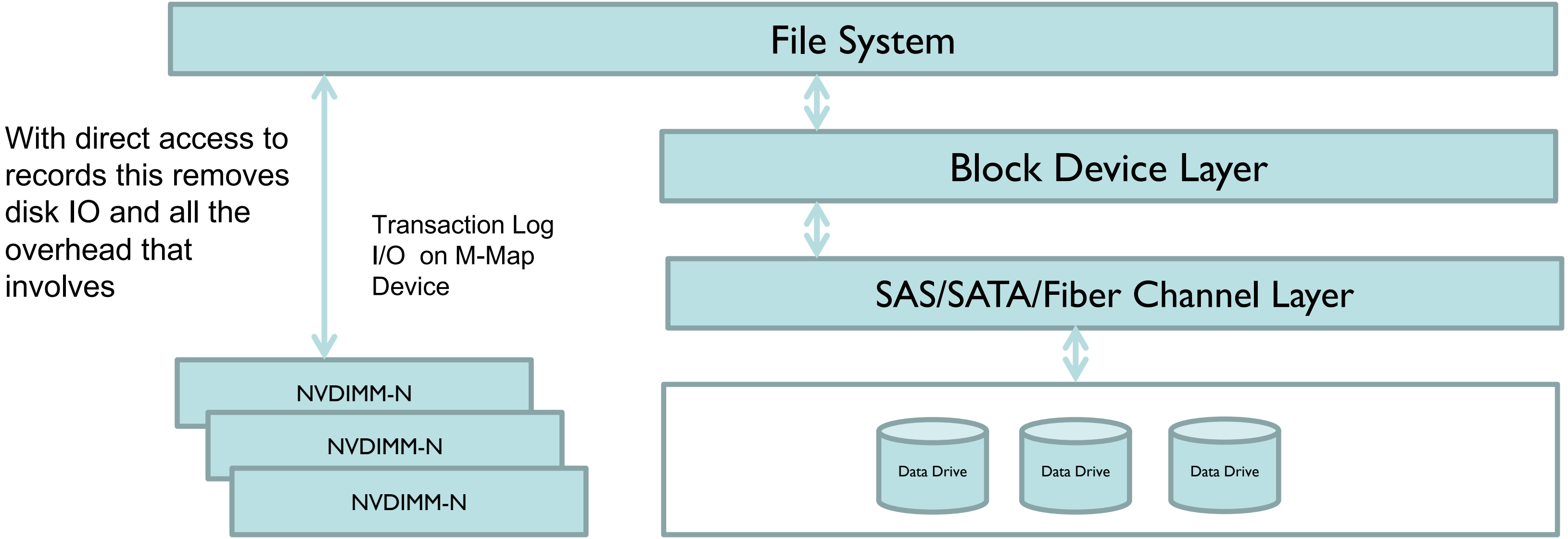
Block diagram example, JEDEC Server Forum Jul’17

- NVDIMM-P interface specification targeting persistent memories and high capacity DRAM memory on DDR4 and DDR5 channels
- It extends the DDR protocol to enable transactional access
  - Host is decoupled from the media
  - Multiple media types supported
- Supports any latency (ns ~ us)
- JEDEC specification publication in 2018

NVDIMM-N Applications

- In Memory Database: Journaling, reduced recovery time, Ex-large tables
- Traditional Database: Log acceleration by write combining and caching
- Enterprise Storage: Tiering, caching, write buffering and meta data storage
- Virtualization: Higher VM consolidation with greater memory density
- High-Performance Computing: Check point acceleration and/or elimination

NVDIMM-N Use Case
File System Transaction Log



With direct access to records this removes disk IO and all the overhead that involves

A transaction log is a history of actions executed by a DBMS used to guarantee Atomicity, Consistency, Isolation, and Durability (ACID) over a hardware failure. When these logs can be stored in NVDIMMs vs storage then system performance can be dramatically improved.

Linux Kernel 4.4+ NVDIMM-N OS Support

- ❒ Linux 4.4 + subsystems added support of NVDIMMs
- ❒ NVDIMM modules presented as device links: /dev/pmem0, /dev/pmem1
- ❒ QEMO support (experimental)
- ❒ XFS-DAX and EXT4-DAX available

DAX

File system extensions to bypass the page cache and block layer to memory map persistent memory, from a PMEM block device, directly into a process address space.

BTT (Block,Atomic)

Block Translation Table: Persistent memory is byte addressable. Existing software may have an expectation that the power-fail-atomicity of writes is at least one sector, 512 bytes. The BTT is an indirection table with atomic update semantics to front a PMEM/BLK block device driver and present arbitrary atomic sector sizes.

PMEM

A system-physical-address range where writes are persistent. A block device composed of PMEM is capable of DAX. A PMEM address range may span an interleave of several DIMMs.

BLK

A set of one or more programmable memory mapped apertures provided by a DIMM to access its media. This indirection precludes the performance benefit of interleaving, but enables DIMM-bounded failure modes.

Windows
NVDIMM-N OS Support

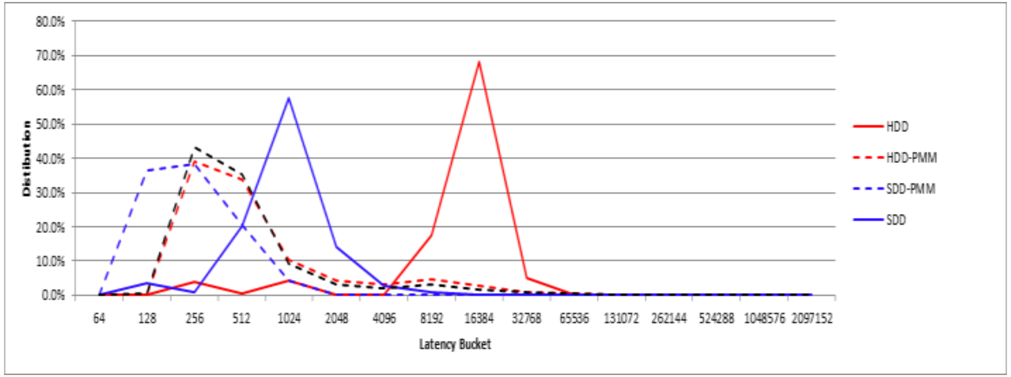
Microsoft

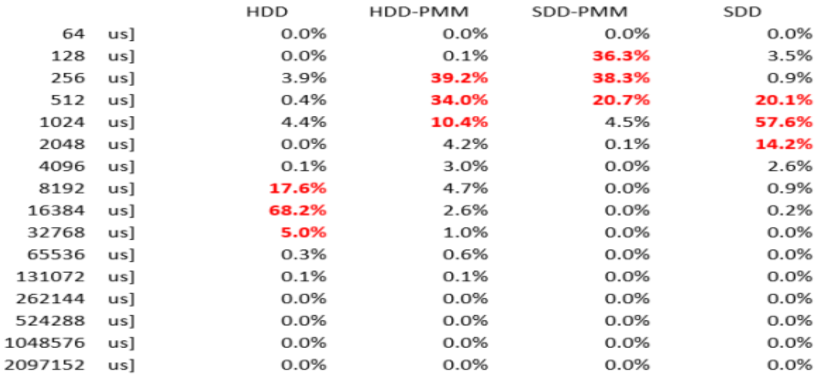
- Windows Server 2016 supports DDR4 NVDIMM-N
- Block Mode
  - No code change, fast I/O device (4K sectors)
  - Still have software overhead of I/O path
- Direct Access
  - Achieve full performance potential of NVDIMM using memory-mapped files on Direct Access volumes (NTFS-DAX)
  - No I/O, no queueing, no async reads/writes
- More info on Windows NVDIMM-N support:
  - https://channel9.msdn.com/events/build/2016/p466
  - https://channel9.msdn.com/events/build/2016/p470

Application Benefits – Windows Example

Tail of Log in SQL 2016

- Writes updates to SQL log through persistent memory first
- Uses memory instructions to issue log updates to persistent memory directly
- Utilizes memory-mapped files on NTFS Direct Access (DAX) volume





		HDD	HDD-PMM	SDD-PMM	SDD
64	us]	0.0%	0.0%	0.0%	0.0%
128	us]	0.0%	0.1%	**36.3%**	3.5%
256	us]	3.9%	**39.2%**	**38.3%**	0.9%
512	us]	0.4%	**34.0%**	**20.7%**	**20.1%**
1024	us]	4.4%	**10.4%**	4.5%	**57.6%**
2048	us]	0.0%	4.2%	0.1%	**14.2%**
4096	us]	0.1%	3.0%	0.0%	2.6%
8192	us]	**17.6%**	4.7%	0.0%	0.9%
16384	us]	**68.2%**	2.6%	0.0%	0.2%
32768	us]	**5.0%**	1.0%	0.0%	0.0%
65536	us]	0.3%	0.6%	0.0%	0.0%
131072	us]	0.1%	0.1%	0.0%	0.0%
262144	us]	0.0%	0.0%	0.0%	0.0%
524288	us]	0.0%	0.0%	0.0%	0.0%
1048576	us]	0.0%	0.0%	0.0%	0.0%
2097152	us]	0.0%	0.0%	0.0%	0.0%

Source; Microsoft

Summary

- The NVM Programming Model is perfect for NVDIMMs
  - Block and File mode atomicity features
  - PM Mode memory mapped storage
- Use the NVM programming model with NVDIMMs
  - Enable a path forward for applications
  - Lead the way to innovation in NVM optimized software

SNIA-at-a-Glance

160
unique member companies

2,500
active contributing members

50,000
IT end users & storage pros worldwide

Join Our Work in

Advancing Solid State and Persistent Memory

SNIA SSSI | SOLID STATE STORAGE

SNIA Persistent Memory

Access technology visionaries and leading companies in the industry

Collaborate to guide technology strategic & technical directions

Influence Industry messaging and best practices

Email asksssi@snia.org to be included in open calls on technology topics and to learn more about how your company can join SNIA, the SSSI, and a SNIA Regional Affiliate

Thank You!

Visit snia.org/PM for the latest on SNIA Persistent Memory activities