SNIA NVM Programming Model v1.2 and Beyond

Tom Talpey
SNIA NVMP TWG
Contents

- Persistent Memory Overview
- SNIA NVM Programming Model
- NVDIMM technologies and OS support
Persistent Memory
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
- 3DXPoint™ Memory
- Low Latency NAND
- HBM
- MRAM
- Managed DRAM
- RRAM
- PCM

Source: Gen-Z Consortium 2016
Persistent Memory (PM) Vision

- For system acceleration
- For real-time data capture, analysis and intelligent response
Persistent Memory (PM) is a type of Non-Volatile Memory (NVM)

- Disk-like non-volatile memory
  - Persistent RAM disk
  - 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 I/O operation is required

- Industry terminology trend
  - NVM is **block** oriented (e.g. NVMe)
  - PM is **byte** oriented (e.g. NVDIMM)
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 I/O queuing, demand paging or page caching
- Think: Power Protected RAM
NVM Programming Model

Writing Applications for Persistent Memory
SNIA NVM Programming Model

- Developed to address the ongoing proliferation of new non-volatile memory functionality and new persistent memory technologies.
- The NVM Programming Model is necessary to enable an industry-wide community of persistent memory producers and consumers to move forward together through a number of significant storage and memory system architecture changes.
- The specification defines recommended behavior between various user space and operating system (OS) kernel components supporting persistent memory.
- The specification describes several operational modes of access. Each mode is described in terms of use cases, actions and attributes that inform user and kernel space components of functionality that is provided by a given compliant implementation.
Role of the SNIA 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
    - Processors
SNIA NVM Programming Model TWG - Mission

- Accelerate the availability of software that enables Persistent Memory hardware.
  - Hardware includes SSD and PM
  - Software spans applications and Operating Systems

- Create the NVM Programming Model
  - Describes application-visible behaviors
  - Allows APIs to align with OSes
  - Exposes opportunities in networks and processors
SNIA NVM Programming Model

- 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 APIs
NVM Programming Model v1.2: 4 modes

<table>
<thead>
<tr>
<th></th>
<th>IO</th>
<th>Persistent Memory</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>User View</strong></td>
<td>NVM.FILE</td>
<td>NVM.PM.FILE</td>
</tr>
<tr>
<td><strong>Kernel Protected</strong></td>
<td>NVM.BLOCK</td>
<td>NVM.PM.VOLUME</td>
</tr>
<tr>
<td><strong>Media Type</strong></td>
<td>Disk Drive</td>
<td>Persistent Memory</td>
</tr>
<tr>
<td><strong>NVDIMM</strong></td>
<td>Disk-Like</td>
<td>Memory-Like</td>
</tr>
</tbody>
</table>
Programming Model Modes

- Block and File modes use I/O model
  - 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 Load/Store access
  - Data is loaded into or stored from processor registers
  - Processor pends software 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 applications
  - 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 file data to application address space
  - Syncing (Flushing) PM file data
Expected Usage of PM modes

- Uses for NVM.PM.VOLUME
  - Kernel modules
  - PM aware file systems
  - Storage stack components
- Uses for NVM.PM.File
  - Applications
    - Persistent datasets, directly addressable, no DRAM footprint
    - Persistent caches (warm cache effect)
  - Reconnect-able BLOBs of persistence
    (Binary Large Object – set of binary data stored as a single entity)
    - Naming
    - Permissions
Persistent Memory Modes

**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
Publication Timeline

- NVM Programming Model v1 technical position December 2013
- NVM Programming Model v1.1 technical position March 2015
- Remote Access for High Availability white paper 2016
- PM Atomics and Transactions white paper 2016
- NVM Programming Model v1.2 published June 2017
SNIA NVM Programming Model v1.2

- Published June 2017
- Major new installment on error handling
- Optimized Flush Allowed attribute
  - Greater application fidelity to OS required semantics
- Deep Flush operation
  - Greater persistence reliability (at cost of latency)
Ongoing SNIA NVMP TWG Work Items

- 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 SDC 2017 talk
Summary

- The SNIA NVM Programming Model is aligning the industry
  - [http://snia.org/forums/sssi/nvmp](http://snia.org/forums/sssi/nvmp)
  - Defining common terminology
  - Not forcing specific APIs
  - One example: [http://pmem.io/](http://pmem.io/)

- What are we doing with it?
  - PM models expose it
    - Linux PMFS at [https://github.com/linux-pmfs](https://github.com/linux-pmfs)
  - New PM models build on existing ones
    - Linux Pmem Examples: [https://github.com/pmem/linux-examples](https://github.com/pmem/linux-examples)
    - New TWG work items

- Emerging technologies will drive increasing work in this area
NVDIMM Types

- Host has direct access to DRAM
- Controller moves DRAM data to Flash on power fail
- Requires backup power (typically 10’s of seconds)
- Controller restores DRAM data from Flash on next boot
- Communication through SMBus (JEDEC std)

- Host accesses Flash through controller
- Block-access to Flash, similar to an SSD
- Enables NAND capacity in the memory channel (even volatile operation)
- Communication through SMBus (JEDEC std TBD)

- Combination of -N and -F
- Host accesses memory through controller
- Definition still under discussion
- Sideband signaling for transaction ID bus
- Extended addressing for large linear addresses
NVDIMM Cookbook

File System
- Page Cache
- Block-based FS
- Block wrapper

OS
- PM-based FS
- DAX-enabled FS
- NVDIMM Driver

BIOS
- MRC + BIOS (Memory Ref. Code)

Server
- Memory Controller
- Power Supply
- DIMM interface (Inc. SAVE trigger)

NVDIMM
- NVDIMM-N
- Energy Module

Legend:
- Software
- Hardware
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: Checkpoint acceleration and/or elimination
Linux Kernel 4.4+ NVDIMM-N Support

- Linux 4.2 + subsystems added support of NVDIMMs. Mostly stable from 4.4
- NVDIMM modules presented as device links: /dev/pmem0, /dev/pmem1
- QEMU support (experimental)
- XFS-DAX and EXT4-DAX available

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.

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.

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.

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 Support

- Windows Server 2016 and Windows 10 Anniversary update (Fall 2016) support JEDEC DDR4 NVDIMM-N
- Block Mode – Persistent RAM disk
  - No code change, fast I/O device (4K sectors)
  - Still have software overhead of I/O path
- Direct Access – Mapped file 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/p466)
  - NVML open source library: [https://github.com/pmem/nvml/releases](https://github.com/pmem/nvml/releases)

<table>
<thead>
<tr>
<th>4K Random Write</th>
<th>Thread Count</th>
<th>IOPS</th>
<th>Latency (us)</th>
</tr>
</thead>
<tbody>
<tr>
<td>qd=1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>NVDIMM-N (block)</td>
<td>1</td>
<td>187,302</td>
<td>5.01</td>
</tr>
<tr>
<td>NVDIMM-N (DAX)</td>
<td>1</td>
<td>1,667,788</td>
<td>0.52</td>
</tr>
</tbody>
</table>
Summary

- NVDIMM support is enabled in Operating Systems today

- The SNIA NVM Programming Model is perfect for NVDIMMs
  - Block and File mode atomicity features for Type F
  - PM mode memory mapped storage for Type N

- Use the SNIA NVM Programming Model
  - Enable a path forward for applications
  - Lead the way to innovation in PM-optimized software
Thank You!