SFD13 SNIA

*Persistent Memory, NVM Programming Model, and NVDIMMs
Rob Peglar
Alan Bumgarner
Tom Talpey

SSSI is a very active part of what they're doing
www.snia.org/PM

Persistent Memory
Volatile and non-volatile technologies are continuing to converge

Persistent memory brings storage to memory slots
- for system acceleration
- for real-time data capture, analysis and intelligent response

PM is a type of NVM
Disk-like non-volatile memory
- president 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 IO or even DMA is required

PM characteristics
- byte addressable form 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

Role of the NVM Programming Model
Rally the industry around a view of Persistent Memory that is:
- Application centric
- Vendor netral
- Achievable today
- Beyond storage (applications, memory, networking, processors)

NVM Programming Model TWG - Mission
Accelerate the availability of software that enables Persistent Memory hardware
- Hardware includes SSDs and PM
- Software spans applications and OSes

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

SNIA NVM Programming Model (photo)
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 in APIs

4 modes
spec at http://www.snia.org/tech_activities/standards/curr_standards/npm

This is not NVMe

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 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

PM Modes
NVM.PM.VOLUME Mode
- Software abstraction for memory hardware
- Address ranges
- thin provisioning management

NVM.PM.FILE Mode
- Application behaviour 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
- optimised flush - multiple ranges from user space
- optimised flush and verify - optimised 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 mnaagemt
Use an offset from a relocatable base
- base could be a start of the memory mapped file
- pointer includes namespace reference

Failure Atomicity
Current processor and 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 soecififc
- 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
- tailback is an additional checksum or CRC

Error handling - exceptions instead of status

NVDIMM SIG

NVDIMM Types
NVDIMM-N
- host has direct access to DRAM
- CNTLR moves DRAM data to Flash on power fail
- requires backup power (typically 10s of seconds)
- CNTLR restores DRAM data from Flash on next boot
- Communication through SMBus (JEDEC std)
NVDIMM-F
- 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)

NVDIMM-P
- Combination of -N and -F
- Host accesses memory through controller
- Definition stillunder discussion
- sideband signalling for transaction ID bus
- Extended addressing for large linear addresses

Application Acess to NVDIMMs

Disk-like NVDIMMs (Type F or P)
- Appear as disk drives to applications
- Accessed using disk stack

Memory-like NVDIMMs (Type N or P)
- 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
NVDIMMs are available today

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 metadata storage
Virtualisation: Higher VM consolidation with greater memory density
High-performance Computing: Check point acceleration and/or elimination

Windows NVDIMM-N OS Support (photo)

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 queuing, no async reads/writes

More info on Windows NVDIMM-N support
- http://channel9.msdn.com/events/build/2016/p466
- http://channel9.msdn.com/events/build/2016/p470

Summary

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

Use the NVM programming model with NVDIMMs
- enable a path forward for applications
- lead the way to innovation in NVM optimised software

*Reel It In: SNIA Swordfish
Richelle Ahlvers (@rahlvers)

www.snia.org/swordfish

Customers (and vendors) are asking for improvements in storage management APIs
- make them simpler to implement and consume
- improve access efficiency (fewer transactions, with more useful information in each)
- provide useful access via a standard browser
- expand coverage to include converged, Hyperconvered, and hyper-scale
- provide compatibility with standard DevOps environments

What is SNIA Swordfish?
What?
- Refactor and leverage SMI-S schema into a simplified model that is client oriented
- Move to class of service based provisioning and monitoring
- Cover block, file and object storage
- extend traditional storage domain coverage to include converged environments (covering servers, storage and fabric together)
How?
- Leverage and extend DMTF Redfish Specification (focuses on hardware and system management)
- build using DMTF's Redfish technologies (RESTful interface over HTTPS in JSON format based on Data v4)
- Implement Swordfish as a seamless extension of the Redfish specification

Redfish Resource Map

Adding storage to Redfish

Who is behind Redfish and Swordfish?
Both - Broadcom, Dell Inc, Fujitsu, HPE, Huawei, IBM, Intel, Lenovo, Microsemi, NetApp, Texas Tech Uni, Toshiba, VMware, Western Digital

How's it Going?
0 to v1 in 9 months
v1.0.4 release in April / May 2017
v1.0.5 release targeted in July / August 2017
Development of open source tools (Swordfish emulator, web client, dashboard integrations - PowerBI, Datadog)
Incremental features / functionality added to spec after v1 is "validated" by initial implementations