What we will cover

- PMEM – Hardware…and the associated programming model
- What everyone already should know about pmem…
- What everyone forgets…
- Ways to use pmem with no app modifications
- Ways to use pmem with app modifications
- Learnings so far
- Where we’re heading
- An introduction to Computational Storage
A Fundamental Change Requires An Ecosystem

- JEDEC JESD245B.01: Byte Addressable Energy Backed Interface (released Jul’17)
- JEDEC JESD248A: NVDIMM-N Design Standard (released Mar’18)
- SNIA NVM Programming Model (v1.2 released Jun’17)
- unfit ACPI NVDIMM Firmware Interface Table (v6.2 released May’17)

- Windows Server 2016
- Windows 10 Pro for Workstations
- Linux Kernel 4.2 and later
- VMware, Oracle, SAP HANA early enablement programs

- Multiple vendors shipping NVDIMMs
- SNIA NVDIMM Special Interest Group (formed Jan’14)
- Successful demonstrations of interoperability among vendors

- All major OEMs shipping platforms with NVDIMM support
- Requires hardware and BIOS mods
Host is decoupled from the media (agnostic to PM type)
New protocol to “hide” non-deterministic access
Capacity = PM (100’s GB+)
Latency = PM (>> 10’s of nanoseconds)
Endurance = PM (finite)
Likely to impact memory bus performance
Complex controller & buffer scheme likely required
Specifications still under definition (2H’19 release?)
No ecosystem yet, likely DDR5 timeframe

JEDEC-Defined NVDIMM Types

NVDIMM-N
- Host has direct access to DRAM
- NAND flash is only used for backup
- Capacity = DRAM (10’s - 100’s GB)
- Latency = DRAM (10’s of nanoseconds)
- Endurance = DRAM (effectively infinite)
- No impact to memory bus performance
- Low cost controller can be implemented
- Specifications completed and released
- Ecosystem moving into mature stage

NVDIMM-P
- Host has direct access to DRAM
- NAND flash is only used for backup
- Capacity = DRAM (10’s - 100’s GB)
- Latency = DRAM (10’s of nanoseconds)
- Endurance = DRAM (effectively infinite)
- No impact to memory bus performance
- Low cost controller can be implemented
- Specifications completed and released
- Ecosystem moving into mature stage

NVDIMM Types Are Complementary, Not Competing
NVDIMM Target Application Areas

- **Database**
  - USE CASES: Log Acceleration, In-Memory Commit

- **Storage**
  - USE CASES: Filesystems, Fast Caching, SSD Wear-Out

- **Virtualization**
  - USE CASES: Higher VM Consolidation, More Virtual Users/System

- **Big Data**
  - USE CASES: Fast IOPs Workloads, In-Memory Processing

- **Cloud Computing/IoT**
  - USE CASES: Byte-Level Data Processing, Metadata Store

- **Artificial Intelligence**
  - USE CASES: Low Latency Look-Up & Processing

The same factors driving NAND Flash adoption apply to NVDIMMs: IOPS, Latency, Performance. NVDIMM addressing is exactly like DRAM.
Everyone should know…

❖ Persistent memory…
  ❦ Allows load/store access like memory
  ❦ Is persistent like storage
  ❦ Exposed to applications using SNIA NVM TWG model

❖ What isn’t persistent memory:
  ❦ Something that can only speak blocks (like a disk/SSD)
  ❦ Something that is too slow for load/store access
    ❧ TWG’s language: Would reasonably stall the CPU waiting for a load to complete
Often forgotten

- The programming model includes the storage APIs!
Often forgotten: Storage Access

The programming model includes the storage APIs!

Use PM Like an SSD
Often forgotten: DAX Access

The programming model includes the storage APIs!

Use PM Like an SSD

Use PM Like an SSD (no page cache) “DAX”
Memory Mode: Volatile Capacity
No Application Modification

- **Using PM as a fast SSD**
  - Storage APIs work as expected
  - Memory-mapping files will page them into DRAM

- **Using PM as DAX**
  - Storage APIs work as expected
  - No paging (DAX stands for “Direct Access”)

- **Using PM as volatile capacity**
  - Just big main memory
  - Vendor-specific feature
Often forgotten: DAX Access

The programming model includes the storage APIs!

Use PM Like an SSD

Use PM Like an SSD (no page cache)

“DAX”
Optimized Flush: Flushing from Userspace

The programming model includes the storage APIs!

Use PM Like an SSD

Use PM Like an SSD (no page cache) "DAX"
Application Modification

**Language Bindings**

- C
- C++
- LLPL
- PCJ
- Python

**Low-level support**

- Support for volatile memory
  - libmemkind
- Low level support for local persistent memory
  - libpmem
- Low level support for remote access to persistent memory
  - librpmem

**Transaction Support**

- Interface to create a persistent memory resident log file, e.g. Write Ahead Logging (WAL)
  - libpmemlog
- Interface for persistent memory allocation, transactions and general facilities
  - libpmemobj
- Interface to create arrays of pmem-resident blocks, of same size, atomically updated
  - libpmemblk

**In Development:**

- PCJ – Persistent Collection for Java
- LLPL – Low-Level Persistence Java Library
Application Modification: pmemkv

- **libpmemkv**
  - Experimental
  - General-purpose key-value store
  - Multiple pluggable engines
  - Multiple language bindings
  - Productization underway

- **Caller uses simple API**
  - But gets benefits of persistent memory
Full Stack Example

**libpmem**
- Abstract away hardware details
- Provide transactions, persistent memory allocator
- Expose Persistent Memory as memory-mapped files (DAX)

**LLPL**
- Provide Java transactions, allocations
- Caller just sees the same APIs, uses them as before

**Cassandra**
- Use Java containers to create pmem-aware Cassandra

**pmmem**
- Abstract away hardware details

**pmmem-aware File System**
- Expose Persistent Memory as memory-mapped files (DAX)

**libpmemobj**
- Provide transactions, persistent memory allocator

**App**
- Unmodified App, uses Cassandra APIs

**Unmodified App, uses Cassandra APIs**

**SNIA Programming Model**

**PMDK**

© 2018 Storage Networking Industry Association. All Rights Reserved.
Learnings so far…

- Lots of ways to use PM without app modifications
- Try first to use existing APIs
  - Example: app that can be configured for SSD tier
- Try next to use highest abstraction possible
  - Key-value store, simple block or log interfaces
- Try next to use a transaction library
  - libpmemobj
- Finally, if you must program to raw mapped access
Where we’re heading

❖ More transparent use cases
  ❖ Either kernel or library features, transparent to app

❖ More high-level abstractions
  ❖ Easier to program, less error prone

❖ More support for experts as well
  ❖ More features in transaction libraries
  ❖ More language integration
  ❖ Faster remote (RPM) access
RPM…Some Challenges, But Usable

- NUMA, by definition
  - Probably okay, just be aware of it
- Generally requires asynchronous operation
  - Including delayed completions
- Networks introduce unavoidable latencies
  - As long as the application can tolerate it
- Transaction model will often favor pull vs push operations
  - Not necessarily native to the way application writers think

Net-net, probably can’t treat remote and local PM exactly the same. Not quite transparent, but close.
Java Access to Persistent Memory

- Java is a very popular language on servers, especially for databases, data grids, etc., e.g. Apache projects:
  - Cassandra
  - Ignite
  - HBase
  - Lucene
  - Spark
  - HDFS

- Want to offer benefits of persistent memory to such applications
PM Storage Engine for Cassandra

- Cassandra is a popular distributed NoSQL database written in Java
- Uses a storage engine based on a Log Structured Merge Tree with DRAM and disk levels
- Could persistent memory offer Cassandra opportunities for simpler code and improved performance?
Cassandra Write Path

1. Commit Log
2. Memtable
3. SSTable

DRAM

Disk

© 2018 Storage Networking Industry Association. All Rights Reserved.
Cassandra Write Path – PM Storage Engine
Software - Persistent Memory Storage Engine

Cassandra Pluggable Storage Engine API
https://issues.apache.org/jira/browse/CASSANDRA-13474

Cassandra Persistent Memory Storage Engine
https://github.com/shyla226/cassandra/tree/13981_llpl_engine

Low-Level Persistence Library (LLPL)
https://github.com/pmem/llpl

Java VM (JDK 8 or later)

Persistent Memory Development Kit (PMDK)
https://github.com/pmem/pmdk

Linux OS

Persistent Memory
Want to learn more about PM?

SNIA – Persistent Memory Resource Page
https://www.snia.org/PM

2019 Persistent Memory Summit
https://www.snia.org/pm-summit
Computational Storage?
Compute, Meet Data

- Based on the premise that storage capacity is growing, but storage architecture has remained mostly unchanged dating back to pre-tape and floppy...

- How would you define changes to take advantage Compute at Data?
What is Computational Storage

Central Processor-Driven

- Large Data Transfers, PCIe is bottleneck
- Data pre/post processing & analysis
- Data can bypass the host - video delivery
- Ability to move Software App to Storage

When to USE NVMDIMM

- Compute heavy with small data-transfer
- Small data compute - in-memory compute
- Little to no parallelism
40+ Participating Companies
128+ Individual Members
A New Product Category

- Computational Storage Device (CSx)
  - Computational Storage Drive (CSD)
  - Computational Storage Processor (CSP)
  - Computational Storage Array (CSA)
Possible Architectures
Many Paths to Computational Storage
In Summary – Call to Action

- Computational Storage is a Real Market
  - Customers are deploying today

- Solutions exist and will continue to grow
  - Making the ‘uniform’ helps adoption

- Standardizing the host interaction is vital
  - We NEED more Support from Users/SW Solutions

- Working with all TWG/SIGs & initiatives is key
  - Joining forces and cross-membership adds to success
Thank You!!

www.SNIA.org/Computational
www.SNIA.org/forums/sssi/NVDIMM