Persistent Memory

Industry Status and Update
Alex McDonald, NetApp
SNIA EMEA & SSSI
What we will cover

- What everyone already **should** know about pmem...
- What everyone forgets...
- PMEM – Hardware…and the associated programing model
- Ways to use pmem with no app modifications
- Ways to use pmem with app modifications
- Learnings so far
- Where we’re heading
Everyone should know…

 Persisting memory…

 ✦ Allows load/store access like memory
 ✦ Is persistent like storage
 ✦ Exposed to applications using SNIA NVM TWG model

 What isn’t persistent memoria:

 ✦ 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
A Fundamental Change Requires An Ecosystem

SOFTWARE

- 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)

HARDWARE

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

PLATFORMS

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

STANDARDS

- All major OEMs shipping platforms with NVDIMM support
- Requires hardware and BIOS mods

OTHERS

- 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

- 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)

- All major OEMs shipping platforms with NVDIMM support
- Requires hardware and BIOS mods

© 2018 Storage Networking Industry Association. All Rights Reserved.
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 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 Are Complementary, Not Competing**
The same factors driving NAND Flash adoption apply to NVDIMMs: IOPS, Latency, Performance
NVDIMM addressing is exactly like DRAM
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”
The programming model includes the storage APIs!

Use PM Like an SSD

Use PM Like an SSD
(no page cache)
“DAX”

Optimized flush
Application Modification

Language Bindings

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

Support for volatile memory
- libmemkind

Low level support for local persistent memory
- libpmem

Low level support for remote access to persistent memory
- librpmem

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

Persistent Memory

In Development:
- PCJ – Persistent Collection for Java
- LLPL – Low-Level Persistence Java Library

PMDK

Application

Load/Store

Kernel Space

User Space

Standard File API

MRU Mappings

pmem-Aware File System

Transaction Support

Load/Store

Standard File API

MUSB Mappings

pmem-Aware File System

Transaction Support
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
SNIA Programming Model

PMDK

libpmem

libpmemobj

LLPL

Cassandra

App

Unmodified App, uses Cassandra APIs

Use Java containers to create pmem-aware Cassandra
Caller just sees the same APIs, uses them as before

Provide Java transactions, allocations

Provide transactions, persistent memory allocator

Abstract away hardware details

Expose Persistent Memory as memory-mapped files (DAX)

Persistent Memory
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?
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

PM Hackathons…March…August…online/on-demand…
Get hands-on training and experience