PERSISTENT MEMORY...

- What is it?
- Why is it interesting?
- How does a program use it?
- What are the challenges?
- What’s the state of the ecosystem?
PERSISTENT MEMORY...

- What is it?
  - Why is it interesting?
  - How does a program use it?
  - What are the challenges?
  - What’s the state of the ecosystem?
PROGRESSION OF STORAGE

- HDDs: ~2 ms
- SSDs: ~75us (read)
- PCIe: ~65us (read)
- SSDs: ~300MB/s (read)
- HDDs: ~120MB/s
- PCIe: ~6 GB/s (read)

Latency

Persistent Memory

Bandwidth
DEFINITION OF PERSISTENT MEMORY

- **Byte-addressable**
  - As far as the programmer is concerned
- **Load/Store access**
  - Not demand-paged
- **Memory-like performance**
  - Would reasonably stall a CPU load waiting for pmem
- **Probably DMA-able**
  - Including RDMA

- For modeling, think: Battery-backed DRAM
PERSISTENT MEMORY...

- What is it?

- Why is it interesting?

- How does a program use it?

- What are the challenges?

- What’s the state of the ecosystem?
3D XPoint™

- Persistent, Large Capacity & Byte Addressable
- **6 TB** per two-socket system
- DDR4 Socket Compatible
- Can Co-exist with Conventional DDR4 DRAM DIMMs
- Demonstrated at SAP Sapphire and Oracle Open World 2017
- Cheaper than DRAM
- Availability: 2018
Reach full potential of 3D XPoint™ Technology by connecting it as Memory

Sources: “Storage as Fast as the rest of the system” 2016 IEEE 8th International Memory Workshop and measurement, Intel® Optane™ SSD measurements and Intel P3700 measurements, and technology projections
THE VALUE OF PERSISTENT MEMORY

- Data sets addressable with no DRAM footprint
  - At least, up to application if data copied to DRAM

- Typically DMA (and RDMA) to pmem works as expected
  - RDMA directly to persistence – no buffer copy required!

- The “Warm Cache” effect
  - No time spend loading up memory

- Byte addressable

- Direct user-mode access
  - No kernel code in data path
TRANSPARENCY LEVELS

Increasing barrier to adoption

3D XPoint

File System

Driver

Kernel Space

Application

User Space

- application
- middleware
- libraries

- file systems
- in-kernel usages
- block storage

Increasing barrier to adoption
Increasing **leverage**

**TRANSPARENCY LEVELS**

- Application
- Middleware
- Libraries
- File systems
- In-kernel usages
- Block storage
- 3D XPoint

*User Space*

*Kernel Space*
PERSISTENT MEMORY...

- What is it?
- Why is it interesting?
  - How does a program use it?
  - What are the challenges?
  - What’s the state of the ecosystem?
THE STORAGE STACK (50,000 FOOT VIEW)

Management UI

Management Library

Application

Standard Raw Device Access

Application

Standard File API

File System

Driver

Storage

User Space

Kernel Space
fd = open("/my/file", O_RDWR);
...
count = read(fd, buf, bufsize);
...
count = write(fd, buf, bufsize);
...
close(fd);
fd = open("/my/file", O_RDWR);
...
base = mmap(NULL, filesize, PROT_READ|PROT_WRITE,
             MAP_SHARED, fd, 0);
close(fd);
...
base[100] = 'X';
strcpy(base, "hello there");
*structp = *base_structp;
...
THE PROGRAMMING MODEL

- Management UI
- Application
- Application
- Application
- Management Library
- File System
- File System
- pmem-Aware File System
- NVDIMM Driver
- NVDIMMs

- User Space
- Kernel Space
- “DAX”
- Load/Store
- Standard File API
- Standard File API
- Standard Raw Device Access
- MMU Mappings
- Mgmt.
- Storage
- File
- Memory

OpenFabrics Alliance Workshop 2018
PERSISTENT MEMORY...

- What is it?

- Why is it interesting?

- How does a program use it?

- What are the challenges?

- What’s the state of the ecosystem?
CHALLENGES

▪ Allocation
  • Like malloc/free, but persistent memory aware

▪ Consistency across failure
  • Memory-resident data structures, but transactional updates

▪ DMA and RDMA
  • “just works” if persistence doesn’t matter
  • Gets interesting when persistence matters
    • See Tom’s talk on this next
THE PLATFORM HARDWARE

Core

L1
L1
L2

L3

CPU CACHES

CLWB + fence
-or-
CLFLUSHOPT + fence
-or-
CLFLUSH
-or-
NT stores + fence
-or-
WBINVD (kernel only)

Custom
Power fail protected domain indicated by ACPI property:
CPU Cache Hierarchy

Minimum Required
Power fail protected domain:
Memory subsystem

MOV

WPQ

ADR
-or-
WPQ Flush (kernel only)

DIMM
PERSISTENT MEMORY...

- What is it?
- Why is it interesting?
- How does a program use it?
- What are the challenges?
- What’s the state of the ecosystem?
<table>
<thead>
<tr>
<th>ECOSYSTEM</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS Detection of NVDIMMs</strong></td>
<td>ACPI 6.0+</td>
</tr>
<tr>
<td><strong>OS Exposes pmem to apps</strong></td>
<td><strong>DAX provides SNIA Programming Model</strong>&lt;br&gt;Fully supported:&lt;br&gt;• Linux (ext4, XFS)&lt;br&gt;• Windows (NTFS)</td>
</tr>
<tr>
<td><strong>OS Supports Optimized Flush</strong></td>
<td>Specified, but evolving (ask when safe)&lt;br&gt;• Linux: <strong>safe</strong> with MAP_SYNC&lt;br&gt;• Windows: <strong>safe</strong></td>
</tr>
<tr>
<td><strong>Remote Flush</strong></td>
<td>Proposals under discussion&lt;br&gt;(works today with extra round trip)</td>
</tr>
<tr>
<td><strong>Deep Flush</strong></td>
<td>In latest specification (SNIA NVMP and ACPI)</td>
</tr>
<tr>
<td><strong>Transactions, Allocators</strong></td>
<td>PMDK: <a href="http://pmem.io">http://pmem.io</a>&lt;br&gt;C, C++, Java (early access), Python (very early access)</td>
</tr>
<tr>
<td><strong>Virtualization</strong></td>
<td>All VMMs planning to support PM in guest&lt;br&gt;(KVM changes upstream, Xen in review, others too…)</td>
</tr>
</tbody>
</table>
PERSISTENT MEMORY DEVELOPER KIT (PMDK)

- [http://pmem.io](http://pmem.io)

- **PMDK Provides a Menu of Libraries**
  - Instead of re-inventing the wheel
  - PMDK libraries are fully validated
  - PMDK libraries are tuned for Intel hardware
  - Accelerates ISV readiness
  - Developers pull in just what they need
    - Transaction APIs
    - Persistent memory allocators

- **PMDK Provides Tools for Developers**
- **PMDK is Open Source and Product-Neutral**
PMDK REPLICATION

Application

Standard File API

Load/Store

pmem-Aware File System

MMU Mappings

NVDIMM

Remote Machine

NVDIMM

Same API with and without replication

obj

pmem
SUMMARY

- Persistent Memory technologies are emerging
  - Some are available now
  - Some are available soon
  - Capacity explosion

- The ecosystem has been preparing
  - Pretty far along for local usage
  - Getting interesting for remote usages (Tom’s talk)
ANNOUNCING - SNIA & OPENFABRICS ALLIANCE

SNIA NVMP TWG

- Develop RPM use cases
- Create user-driven API
- Create and Document Programming models

OpenFabrics Alliance

- Open Source Frameworks & APIs
- Vendors develop n/w solutions
THANK YOU

Andy Rudoff

Intel