Persistent Memory Programming: The Current State of the Ecosystem

MSST 2017

Andy Rudoff
Non-Volatile Memory Software Architect
Intel Corporation
Persistent Memory
Optimized System Interconnect

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
Java PersistentSortedMap

PersistentSortedMap employees = new PersistentSortedMap();
...
employees.put(id, data);

No flush calls. Transactional. Java library handles it all.

See “pilot” project at: https://github.com/pmem/pcj
How Did We Get from a DIMM to a Java API?

Persistent Memory → High-Level Language Support

```java
PersistentSortedMap
employees.put(id, data);
```
How Did We Get from a DIMM to a Java API?

Persistent Memory  →  High-Level Language Support

PersistentSortedMap
employees.put(id, data);

Programming Models
“Programming Model”
At least four meanings...

1. Interface between HW and SW

2. Instruction Set Architecture (ISA)

3. Exposed to Applications (by the OS)

4. The Programmer Experience
Programming Model: SW Interface to HW

Core

L1
L1
L2

Core

L1
L1
L2

L3

Memory Subsystem

NVDIMM

Memory bus
Programming Model: SW Interface to HW

Result:
Persistent Memory hardware accessed like memory (cache coherent).
Exposed by ACPI 6.0+ on x86.
Programming Model: Instruction Set Architecture

- MOV
- CLWB
- L3
- NVDIMM
- Memory Subsystem
- Persistent domain
Programming Model: Instruction Set Architecture

MOV

Result:
Stores flushed from CPU cache, globally-visible ➔ Persistent (on x86)
Programming Model: Exposing to Applications

Management UI

Management Library

Application

NVDIMM Driver

File System

PM-Aware File System

NVDIMM

Load/Store

User Space

Kernel Space

Standard Raw Device Access

Standard File API

Standard File API

MMU Mappings
SNIA NVM Programming Model

(Persistent Memory Portion)
Memory-Mapped Files
The Heart of the PM Programming Model

Result: Direct Access (DAX)
Programming Model: The Programmer Experience

Result:
Safer, less error-prone, idiomatic in common languages
NVM Libraries: pmem.io
C/C++ on Linux and Windows

- Open Source
  - [http://pmem.io](http://pmem.io)
- libpmem
- libpmemobj
- libpmemblk
- libpmemlog
- libvmem

NVM Libraries
libpmemobj

Application

Load/Store

API

API

API

atomic

transactions

operations

locks

lists

allocator

libpmemobj

libpmem

memory-mapped pmem
## State of Ecosystem Today

<table>
<thead>
<tr>
<th>OS Detection of NVDIMMs</th>
<th>ACPI 6.0+</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>OS Exposes pmem to apps</strong></td>
<td><strong>DAX provides SNIA Programming Model</strong></td>
</tr>
<tr>
<td></td>
<td>Fully supported:</td>
</tr>
<tr>
<td></td>
<td>• Linux (ext4, XFS)</td>
</tr>
<tr>
<td></td>
<td>• Windows (NTFS)</td>
</tr>
<tr>
<td><strong>OS Supports Optimized Flush</strong></td>
<td>Specified, but evolving (ask when safe)</td>
</tr>
<tr>
<td></td>
<td>• Linux: <strong>unsafe</strong> except Device DAX</td>
</tr>
<tr>
<td></td>
<td>• (and new file systems like <strong>NOVA</strong>)</td>
</tr>
<tr>
<td></td>
<td>• Windows: <strong>safe</strong></td>
</tr>
<tr>
<td><strong>Remote Flush</strong></td>
<td>Proposals under discussion</td>
</tr>
<tr>
<td></td>
<td>(works today with extra round trip)</td>
</tr>
<tr>
<td><strong>Deep Flush</strong></td>
<td>Upcoming Specification</td>
</tr>
<tr>
<td><strong>Transactions, Allocators</strong></td>
<td>Built on above via libraries and languages:</td>
</tr>
<tr>
<td></td>
<td>• <a href="http://pmem.io">http://pmem.io</a></td>
</tr>
<tr>
<td></td>
<td><strong>Much more language support to do</strong></td>
</tr>
<tr>
<td><strong>Virtualization</strong></td>
<td>All VMMs planning to support PM in guest</td>
</tr>
<tr>
<td></td>
<td>(KVM changes upstream, Xen coming, others too…)</td>
</tr>
</tbody>
</table>
NVM Programming Model Resources

www.snia.org/PM

SNIA Standards Portfolio
- NVM Programming Model v1.2a – Draft for public review
- NVM Programming Model v1.1 - SNIA Technical Position
- NVM Programming Model v1.0 - SNIA Technical Position

SNIA Technical White Papers
- NVM PM Remote Access for High Availability
- Persistent Memory Atomics and Transactions

SNIA Videos and Presentations
- The SNIA NVM Programming Model – Latest Developments and Challenges
- Persistent Memory Summit 2017
NVM Libraries
http://pmem.io

C/C++
- C++ bindings: http://pmem.io/nvml/cpp_obj
- libpmemobj page: http://pmem.io/nvml/libpmemobj
- Upstream in some distros already, Windows preview available

NVML Source Tree
- https://github.com/pmem/nvml

Persistent Collections for Java (experimental)
- https://github.com/pmem/pcj

Enhanced valgrind for Persistent Memory
- https://github.com/pmem/valgrind