The Persistent Memory Programming Model

Andy Rudoff, Principal Engineer, Intel
Founding Member, SNIA NVM Programming TWG
The Backstory
Eight Years Summarized in Ten Minutes
SNIA Technical Workgroup Involvement

- 2012 (June)
  - NVM Programming TWG Formed
  - Immediate Participation from key OSVs, IHVs, ISVs
- 2013 (December)
  - SNIA Publishes the NVM Programming Model version 1.0
- 2015 (March)
  - SNIA Publishes version 1.1
- 2017 (June)
  - SNIA Publishes version 1.2
SNIA Starts the Summit

• 2012 (June)
  • NVM Programming TWG Formed
  • Immediate Participation from key OSVs, IHVs, ISVs

• 2013 (December)
  • SNIA Publishes the NVM Programming Model version 1.0

• 2015 (March)
  • SNIA Publishes version 1.1

• 2017 (June)
  • SNIA Publishes version 1.2

2013 (January)
First PM Summit ("NVM Summit")
A Proposed Programming Model
Covering All Three Paths

1. Standard Access
   - File System
   - Kernel

2. NVM regions exposed as files
   - NVM Management API
   - Customer Kernel Modules
   - Open NVM Kernel APIs

3. NVM API
   - NVM User-space APIs
   - Kernel
   - User

NVM Summit

© 2020 SNIA Persistent Memory Summit. All Rights Reserved.
NVM Summit 2014
Two Levels of Flushing Writes

- CLFLUSH, CLFLUSHOPT, CLWB
- PCOMMIT
NVM Library: pmem.io
64-bit Linux Initially

- Open Source
  - http://pmem.io
- Libpmem
- Libpmemobj
- Libpmemblk
- Libpmemlog
- Libvmem

Transactionals

Application

Standard File API

Load/Store

User Space

Library

PM-Aware File System

Kernel Space

NVQIIM
Active Work

- More details on flushing to persistence
  - Includes flushing to remote persistence
- Continue to refine the error model
- Transactions
- APIs?
## Types of Store Barriers

<table>
<thead>
<tr>
<th>Barrier Type</th>
<th>Current Status</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Standard API</strong></td>
<td>Fully specified&lt;br&gt;Full supported:&lt;br&gt;- Linux (xfs, XFS)&lt;br&gt;- Windows (NTFS)</td>
</tr>
<tr>
<td><strong>Optimized Flush</strong></td>
<td>Specified, but evolving (ask when safe)&lt;br&gt;- Linux: unsafe except Device DAX&lt;br&gt;- Windows: safe</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>Upcoming Specification</td>
</tr>
<tr>
<td><strong>Transactions</strong></td>
<td>Built on above via libraries and languages&lt;br&gt;Much more language support to do</td>
</tr>
</tbody>
</table>
TWG Ongoing Work

- **Security**
  - PM Hardware Security Threat Model (balloting)

- **Remote persistent memory (via RDMA)**
  - Ongoing – optimizations for RDMA worked in multiple forums
  - Remote asynchronous flush (under discussion)

- **Higher-level Semantics**
  - As we learn more..
Optimized Flush: Flushing from Userspace

The programming model includes the storage APIs!

Use PM Like an SSD

Optimized flush
Focus of Many Talks Today

Application

Standard File API

Load/Store

User Space

pmem-Aware File System

MMU Mappings

Kernel Space

Persistent Memory

DAX
Where the Model Worked

Hard Earned Successes
Building on the Model
Support for volatile memory usage

- memkind
- vmemcache

Low level support for local persistent memory

- libpmem

Low level support for remote access to persistent memory

- librpmem

Low-level support

- Interface to create a persistent memory resident log file
  - libpmemlog

- Interface for persistent memory allocation, transactions and general facilities
  - libpmemobj

- Interface to create arrays of pmem-resident blocks, of same size, atomically updated
  - libpmembk

In Development

NVDIMM

In Development

PCJ – Persistent Collection for Java

C++

C

PCJ/LLPL

Python

pmemkv

Application

Load/Store

User Space

PMDK

Kernel Space

Standard File API

pmm-Aware File System

MMU Mappings

NVDIMM

Transaction Support

Support for
volatile
memory
usage

Application
2019 – A Very Exciting Year!

• Intel launches Optane DC Persistent Memory
  • Joining multiple NVDIMM-N products in the ecosystem

• Multiple major products move beyond demo to shipping
  • More on this later today

• Number of use cases and SW enablement efforts explodes
  • More on this later today
  • Even includes some start-ups

• Academic activity continues to grow at a steady pace
  • Just two papers of note when our timeline started
  • Now I’ve lost count on how many pmem-related papers are out there
Where the Model Fell Short

Learning from the Ecosystem
The File System

- Application
  - Standard File API
  - Load/Store
    - User Space
      - pmem-Aware File System
        - MMU Mappings
      - Kernel Space
- Persistent Memory
The File System

- Linux ext4
- XFS

Application

Load/Store

User Space

Standard File API

pmmem-Aware File System

MMU Mappings

Kernel Space

Persistent Memory
The File System

Linux
- ext4
- XFS

Windows
- NTFS

Application
- Load/Store
  - User Space

User Space
- Standard File API

Kernel Space
- pmem-Aware File System
  - MMU Mappings

Persistent Memory
The File System

Linux
- ext4
- XFS

Windows
- NTFS

Others
- private or not upstream

Application

Load/Store

User
Space

Kernel
Space

Standard
File API

pmem-Aware
File System

MMU
Mappings

Persistent Memory

Load/Store

pmem-Aware
File System

MMU
Mappings

Persistent Memory

Linux
ext4
XFS

Windows
NTFS

Others
private or not upstream

Standard
File API

Load/Store
The File System

Application

Standard File API

Load/Store

User Space

Optimized Flush: Linux requires MAP_SYNC

Pmem-Aware File System

MMU Mappings

Kernel Space

Persistent Memory

Linux ext4 XFS

Windows NTFS

Others private or not upstream

Linux ext4
XFS

Windows NTFS

Others private or not upstream

pmem-Aware
File System

MMU Mappings

Kernel Space

Persistent Memory

Optimized Flush: Linux requires MAP_SYNC

Linux ext4
XFS

Windows NTFS

Others private or not upstream
The File System

- Linux ext4, XFS
- Windows NTFS
- Others private or not upstream

Uncorrectable in application data: App can handle
The File System

- Linux ext4, XFS
- Windows NTFS
- Others: private or not upstream

Uncorrectable in application data: App can handle

Uncorrectable in FS metadata: Volume lost
The File System

Application

User Space

Load/store

User Space

Kernel Space

Application

Standard File API

pmem-Aware File System

MMU Mappings

Persistent Memory

Possible solutions:
- NOVA
- XFS w/external metadata

Uncorrectable in application data: App can handle

Uncorrectable in FS metadata: Volume lost

Linux ext4 XFS

Windows NTFS

Others private or not upstream
Device DAX on Linux

Legacy Storage API

- Standard Raw Device Access
- Standard File API

Storage API with DAX

- mmap
- Load/Store
- PMDK

Application

- File System
- BTT

Generic NVDIMM Driver

- pmem-Aware File System
- MMU Mappings
- DevDAX

persistent memory

- hardware
- kernel space
- user space

Raw, but powerful (RAS, RDMA, other)
Device DAX on Linux

Legacy Storage API

- Standard Raw Device Access
- Block
- Atomicity
- File System
- Generic NVDIMM Driver

Storage API with DAX

- Standard File API
- Load/Store
- mmap
- PMDK
- pmem-Aware File System
- MMU Mappings
- DevDAX

Application

- user space
- kernel space

Raw, but powerful (RAS, RDMA, other)

Doesn’t follow programming model (POSIX doesn’t work)
Device DAX on Linux

Legacy Storage API

Storage API with DAX

Application

Persistent Memory

File System

Generic NVDIMM Driver

BTT

Default File API

Standard File API

PMDK

mmap

Load/Store

mmap

Raw, but powerful (RAS, RDMA, other)

Doesn't follow programming model (POSIX doesn't work)

The fact Linux invented it shows the model fell short
Ongoing Work

2020 and beyond
The NVM Programming TWG…

• Should probably be called the *PM Programming TWG* now

• Continues to meet, although lower frequency

• Still heavy interest in remote persistent memory (rpmem)

• Still interest in resyncing with ecosystem
  • Publishing a 2.0 PM Programming Specification
Thank you

Please visit www.snia.org/pmsummit for presentations
Master Persistent Memory Programming
Are you ready to begin?

Start Reading