Walking the PMEM Talk

Priya Sehgal (priya.sehgal@netapp.com)
Senior Engineer, NetApp
24th May 2018
Agenda Slide

1) Introduction
2) Persistent Memory in Linux Kernel
3) ZUFS – FUSE for NVM
4) Userspace PMEM Libraries
Persistent Memory/Non-Volatile Memory

Byte Addressable device at near-memory speeds
Implications to Software

- **Persistence: Volatility a Virtue!**
  - Application or OS panics because of an illegal/wrong persistent memory address
  - Ensure durability
  - Ensure ordering

- **Fast Access Speed**
  - Software Stack overhead

- **Byte Addressable**
  - Block-oriented software
  - Can leverage Load/Store

[Caulfield, MICRO ‘10]
NVDIMM Namespaces

- Similar to SCSI LUNs OR NVMe Namespaces
- Persistent Memory Namespace
  - Associated with Interleaved DIMMs
- Block Mode Namespace
  - Associated with specific DIMM
NVDIMM Software Architecture

Persistent Memory in Linux Kernel

Drivers

- **PMEM**
  - Drives a system physical address range, which may be interleaved across multiple DIMMs
  - Provides direct byte-addressable load/store access to NVDIMM Storage
  - 'direct_access' operation – Translate sector number to Page Frame Number
  - You can have any DAX-FS on top of it

- **Block Window Driver (single DIMM)**
  - Enables DIMM-bounded failure modes (e.g., RAID)

- **Block Translation Table (BTT)**
  - Sector-sized old device → block translation table providing atomic and powerfail block update
  - On top of whole block device OR a partition of a block device emitted by either PMEM or BLK Namespace
  - Makes every write an “allocating write” i.e., every write goes to a free block. → Maintains Flog (Free list + log)

- **Device-DAX**
  - Bypass the file system completely
  - Talks directly to PMEM Namespace
Persistent Memory in Linux Kernel

Core Kernel

- Base Platform
  - X86 CPU Instruction
    - Cache management – CLFLUSHOPT, CLWB
    - Machine check safe memcpy – read from persistent memory but also handle any media error
  - X86 MM
    - Huge Pages – 2MB, GB

- Subsystem
  - Memory Management
    - ZONE DEVICE - Tell some portion in NV and it used in RDMA
  - File System
    - XFS-DAX, EXT4-DAX
      - Bypasses page cache
      - DAX fsync/msync
  - NVMDIMM Core
    - Capacity Provisioning – mapping addresses to NVDIMM
    - Error Handling
ZUFS

Zero copy User mode File System
## Userspace vs. Kernel File Systems

<table>
<thead>
<tr>
<th>Kernel</th>
<th>Userspace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast</td>
<td>Portable</td>
</tr>
<tr>
<td></td>
<td>Resilient</td>
</tr>
<tr>
<td></td>
<td>Simple to add functionality and debug</td>
</tr>
<tr>
<td></td>
<td>Fewer licensing restrictions</td>
</tr>
</tbody>
</table>

**Gap:** Near-memory speed Kernel-to-User bridge
## FUSE for PM

<table>
<thead>
<tr>
<th>Design Assumptions</th>
<th>FUSE</th>
<th>ZUFS</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Typical medias</strong></td>
<td>Built for HDDs &amp; extended to Flash</td>
<td>Built for PM/NVDIMMs and DRAM</td>
</tr>
<tr>
<td><strong>SW Perf. goals</strong></td>
<td>• Secondary (High latency media)</td>
<td>• SW is the bottleneck</td>
</tr>
<tr>
<td></td>
<td>• Async I/O Throughput</td>
<td>• Latency is everything</td>
</tr>
<tr>
<td><strong>SW caching</strong></td>
<td>Slow media -&gt; Rely on OS Page Cache</td>
<td>Near-memory speed media -&gt; Bypass OS Page Cache</td>
</tr>
<tr>
<td><strong>Access method</strong></td>
<td>I/O only</td>
<td>I/O and mmap (DAX)</td>
</tr>
<tr>
<td><strong>Cost of redundant copy / context switch</strong></td>
<td>Negligible</td>
<td>The bottleneck -&gt; Avoid copies, queues &amp; remain on core</td>
</tr>
<tr>
<td><strong>Latency penalty under load</strong></td>
<td>100s of µs</td>
<td>3-4 µs</td>
</tr>
</tbody>
</table>
ZUFS
Zero copy User mode File System

- Kernel-to-User Bridge designed for PM

- Key Features and Architecture
  - Low Latency and Efficient
    - Perform I/O synchronously
      - Core and L1 cache affinity
    - Zero data copy
    - Avoid Page Cache
  - Optimal PMEM access
  - NUMA Aware
  - Page table mapping supports I/O and DAX semantics

Preliminary Results
Random 4KB directIO write access

- Measured on
  - Dual socket
  - XEON 2650 (48HT)
  - DRAM backed PMEM

FUSE Vs ZUFS Penalty (PM, DirectIO)

Persistent Memory Libraries
Persistent Memory Development Kit (PMDK)

- Collection of libraries built on DAX feature
- **Libpmem** - provides low level persistent memory support
- **Libpmemobj** - provides a transactional object store, providing memory allocation, transactions, and general facilities for persistent memory programming
- **Libpmemblk** – Block access to pmem. Atomically updated
- **Libpmemlog** – pmem resident log file (used by databases)
- **Librpmem** - low-level support for remote access to persistent memory utilizing RDMA-capable RNICs
- And many more
Example using libpmem

```c
/* create a pmem file */
if ((fd = open("/pmem-fs/myfile", O_CREAT|O_RDWR, 0666)) < 0) {
  perror("open");
  exit(1);
}
/* allocate the pmem */
posix_fallocate(fd, 0, PMEM_LEN)

/* memory map it */
if ((pmemaddr = pmem_map(fd)) == NULL) {
  perror("pmem_map");
  exit(1);
}
/* store a string to the persistent memory */
strcpy(pmemaddr, "hello, persistent memory.");
/* flush above strcpy to persistence */
pmem_persist(pmemaddr, PMEM_LEN);

strcpy(pmemaddr, "hello again, persistent memory.");
pmem_flush(pmemaddr, PMEM_LEN);
pmem_drain();
```

- Force changes to NVM
- Flush processor caches
- Wait for h/w buffers to drain
Thank You
References

- [Caulfield, MICRO ‘10]: Moneta: A High-performance Storage Array Architecture for Next-generation, Non-volatile Memories, Adrian M. Caulfield, Arup De, Joel Coburn, Todor I. Mollov, Rajesh K. Gupta, and Steven Swanson, MICRO 43

- https://github.com/NetApp/zufs-zuf
- https://github.com/NetApp/zufs-zus