Walking the PMEM Talk

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

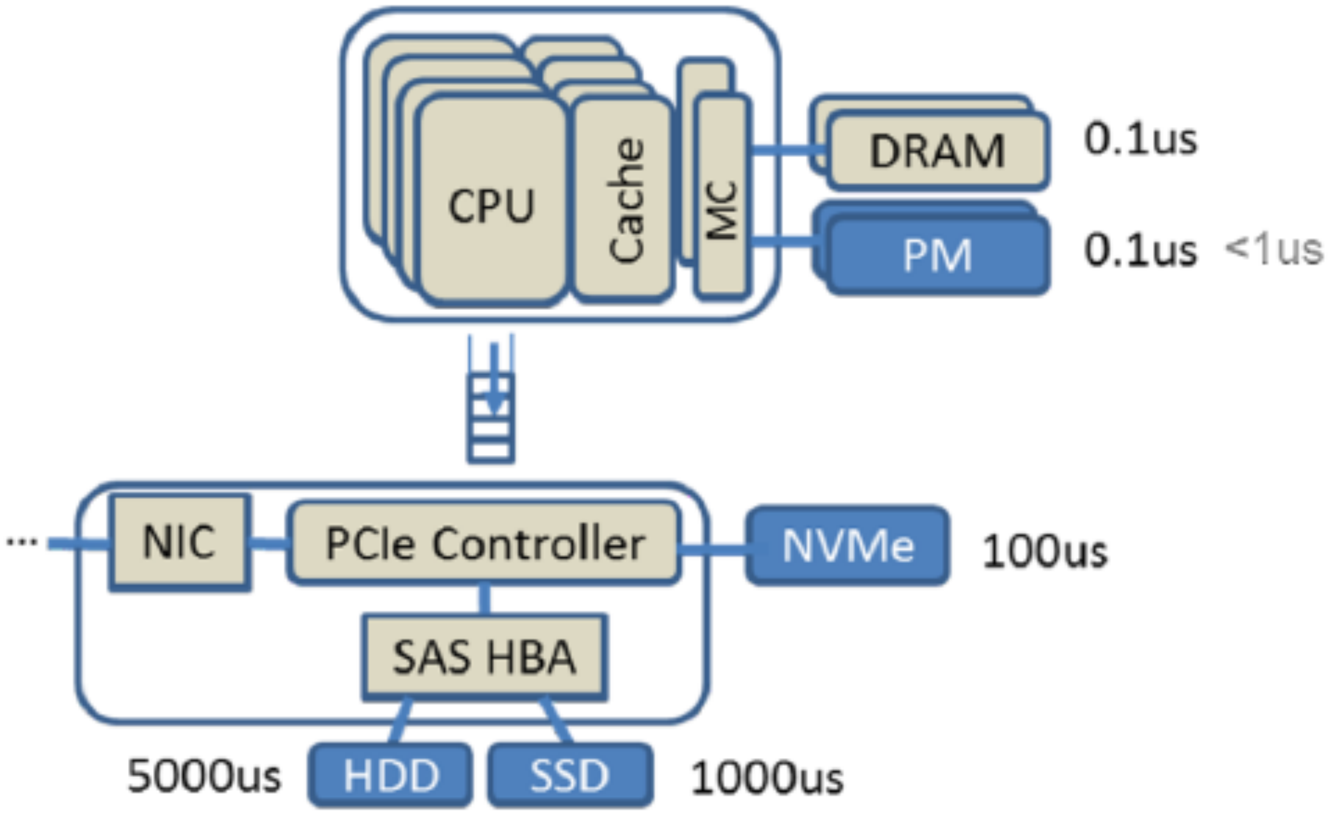
Agenda Slide

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

NetApp

Persistent Memory/Non-Volatile Memory

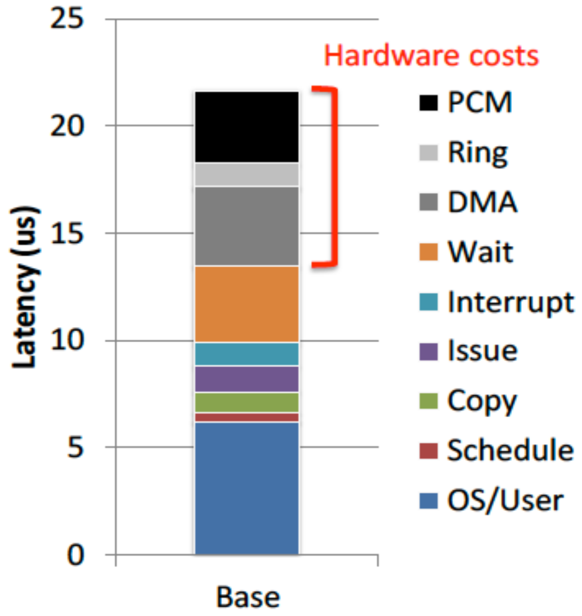
Byte Addressable device at near-memory speeds



NetApp

Implications to Software

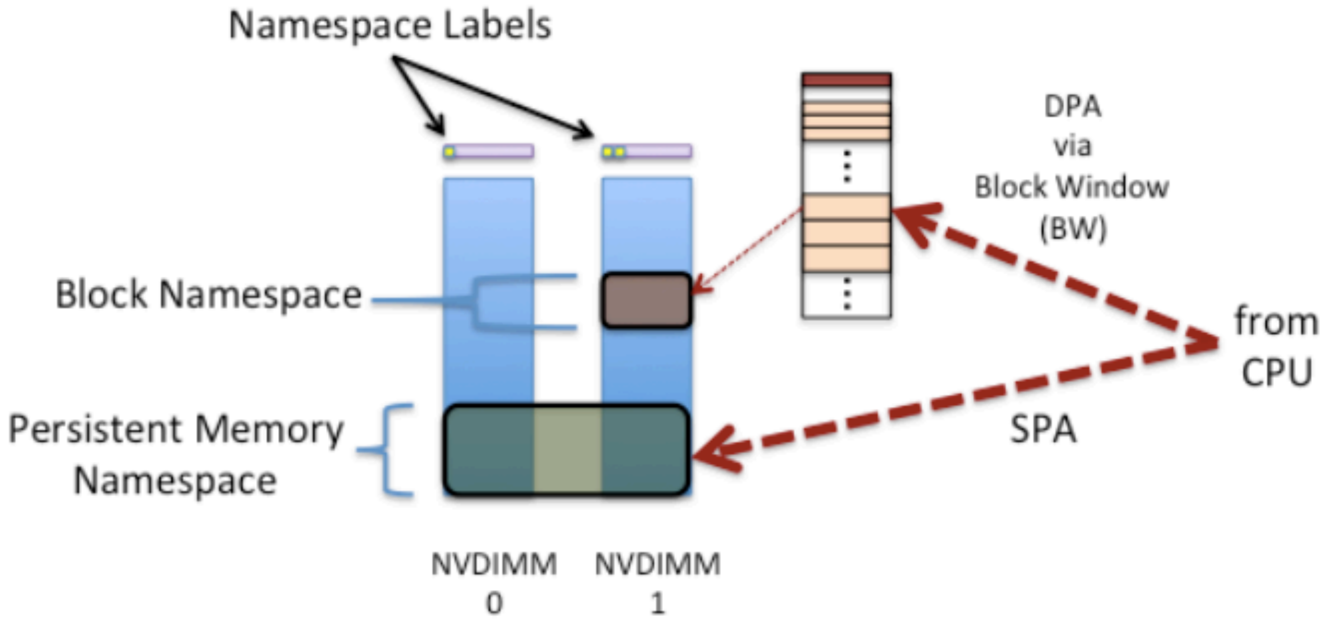
- Persistence: Volatility a Virtue!
  - Application or OS panics because of an illegal/wrong persistent memory address
  - Ensure durability
  - Ensure ordering
- Fast Access Speed
  - Software Stack overhead
- Byte Addressable
  - Block-oriented software
  - Can leverage Load/Store



[Caulfield, MICRO '10]

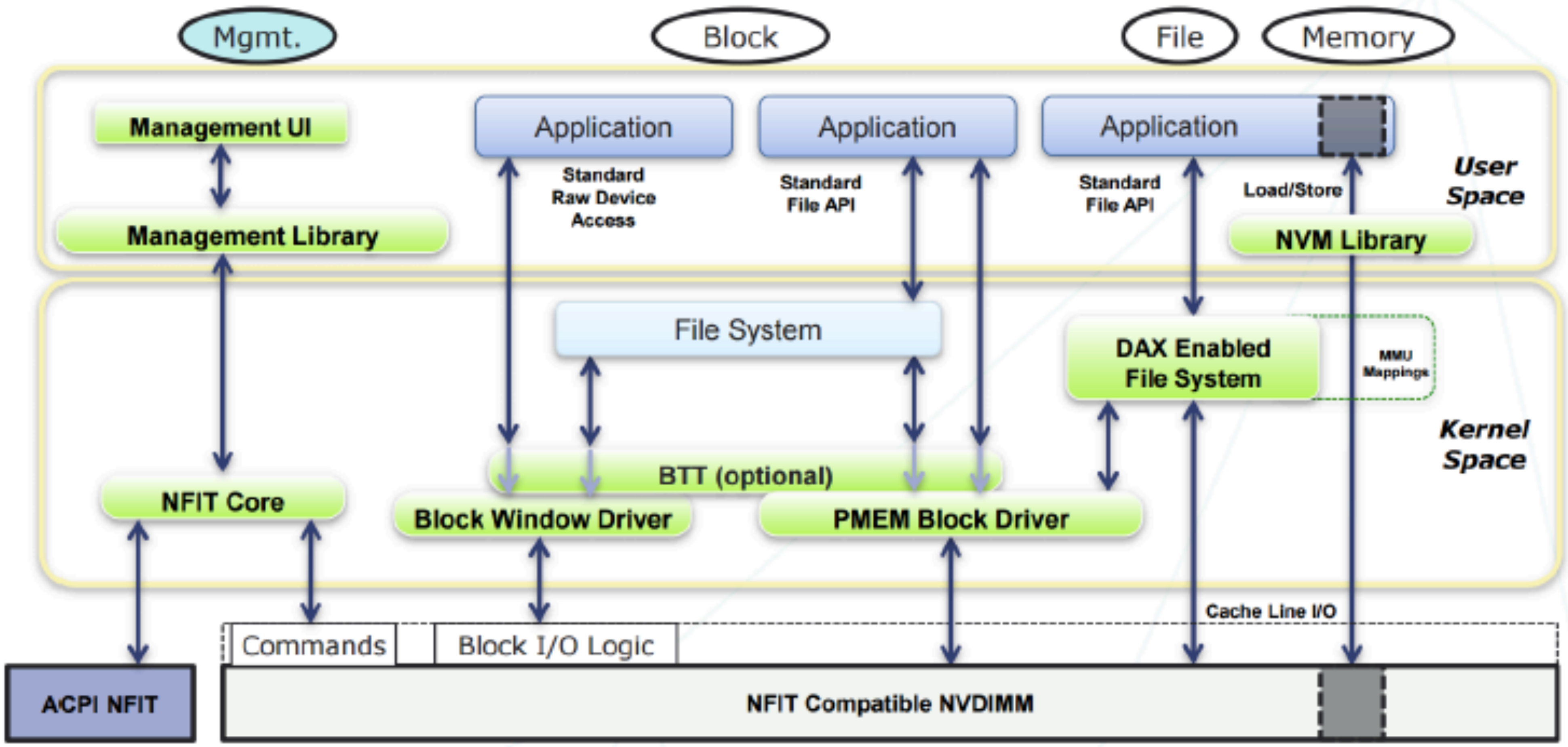
NVDIMM Namespaces

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



NetApp

NVDIMM Software Architecture



http://pmem.io/documents/NVDIMM_Namespace_Spec.pdf

NetApp

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

NetApp

ZUFS

Zero copy **U**ser mode **F**ile **S**ystem

NetApp

Userspace vs. Kernel File Systems

Kernel	Userspace
Fast	Portable
	Resilient
	Simple to add functionality and debug
	Fewer licensing restrictions

Gap: Near-memory speed Kernel-to-User bridge

NetApp

FUSE for PM

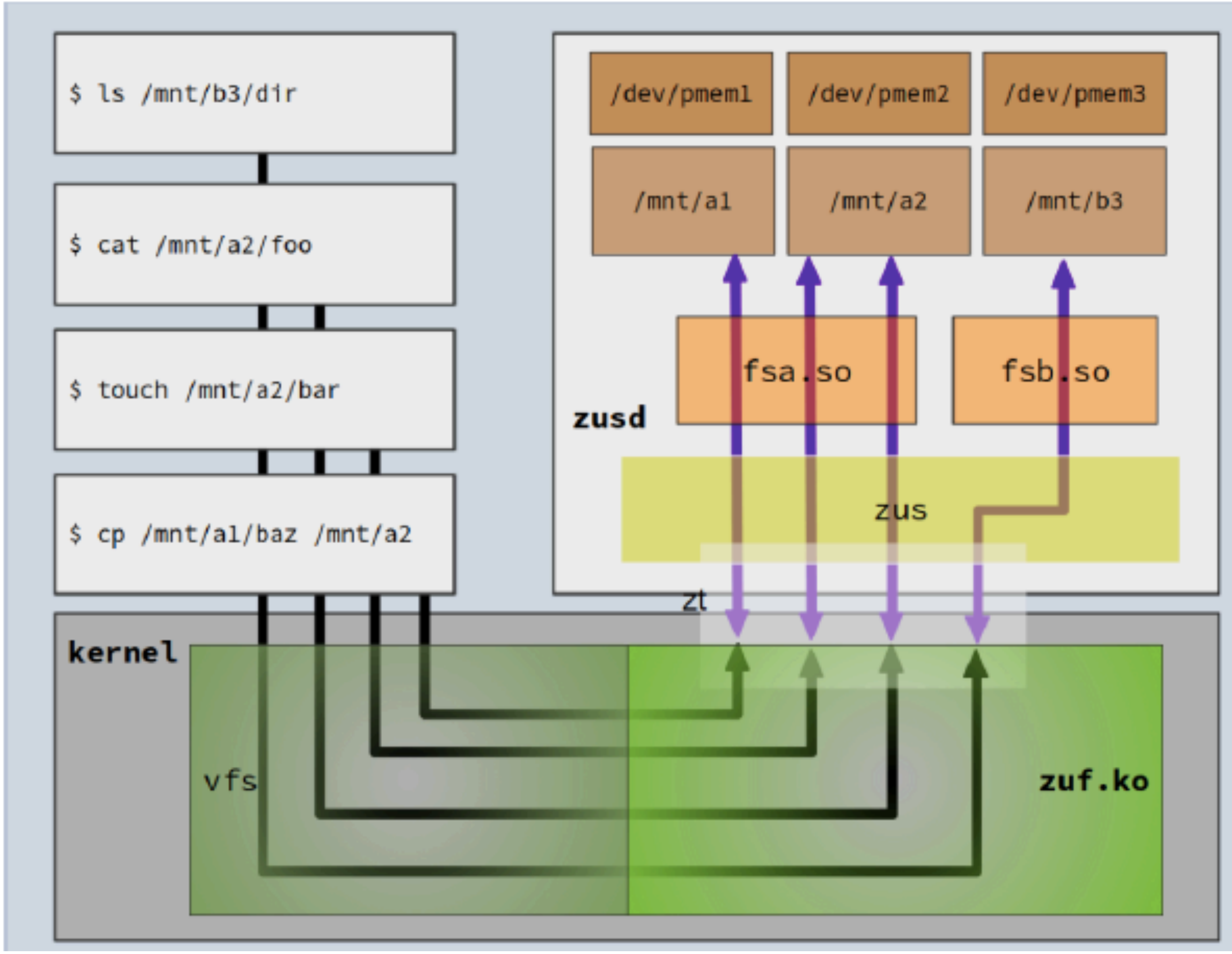
FUSE → ?

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

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

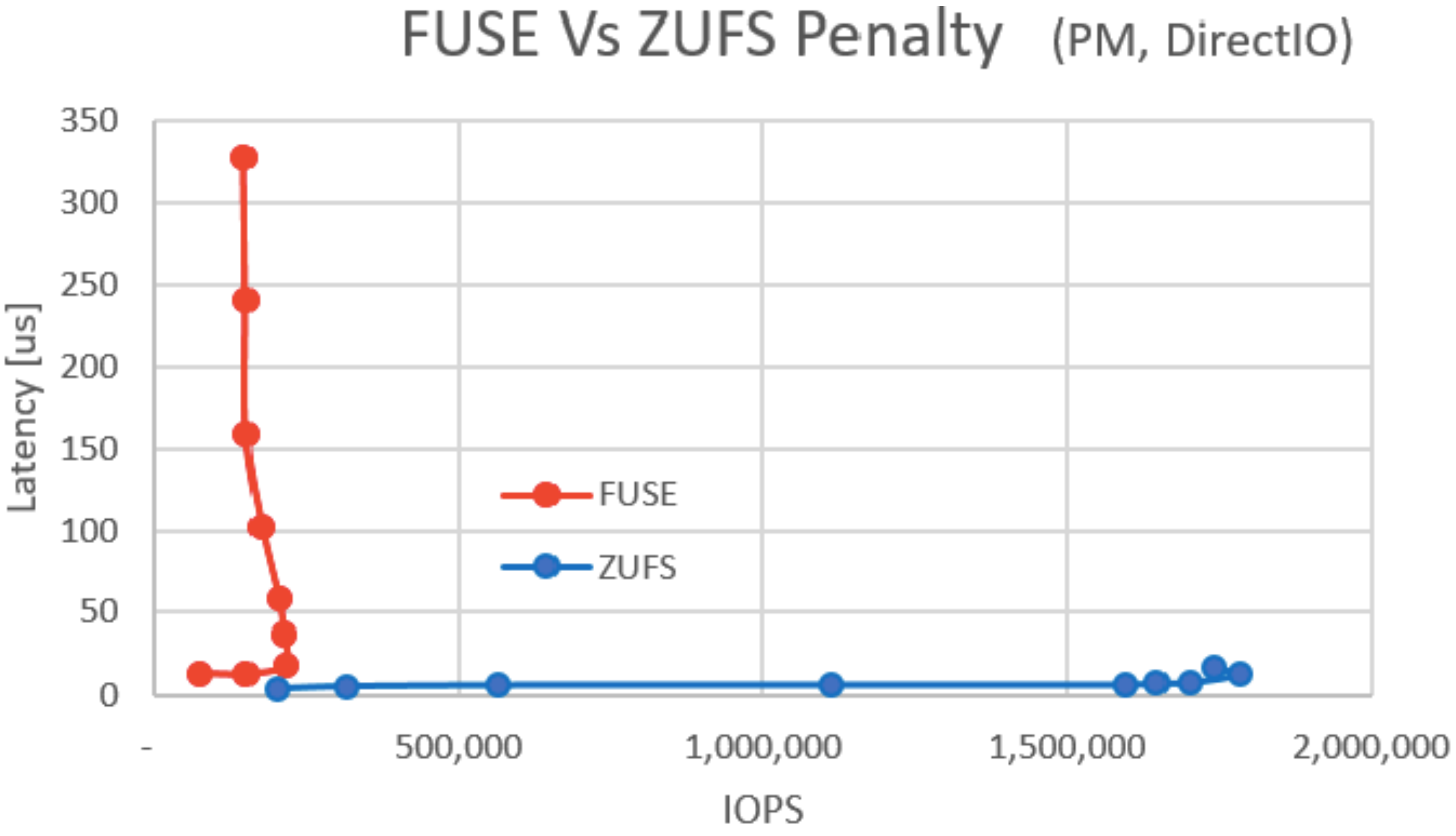


https://www.snia.org/sites/default/files/SDCEMEA/2018/Presentations/5.%20SDC_EMEA_2018_ZUFS_Golander.pdf

Preliminary Results

Random 4KB directIO write access

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



https://www.snia.org/sites/default/files/SDCEMEA/2018/Presentations/5.%20SDC_EMEA_2018_ZUFS_Golander.pdf

NetApp

Persistent Memory Libraries

Persistent Memory Development Kit (PMDK)

pmem.io

- 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

NetApp

Example using libpmem

```
/* 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

NetApp

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
- http://pmem.io/documents/NVDIMM_Namespace_Spec.pdf
- https://www.kernel.org/doc/Documentation/nvdimm/nvdimm.txt
- https://www.kernel.org/doc/Documentation/filesystems/dax.txt
- https://github.com/NetApp/zufs-zuf
- https://github.com/NetApp/zufs-zus

NetApp