PMEM File-System in User-Space

Shachar Sharon
shachar.sharon@netapp.com

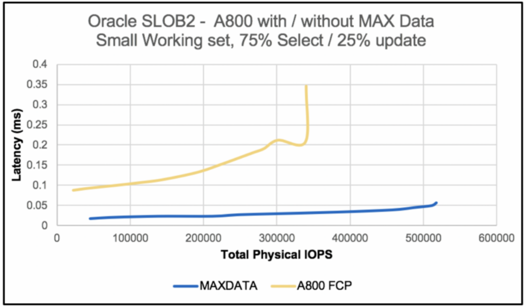
SNIA EMEA, Storage Developer Conference 2020, Tel-Aviv, Israel

In This Talk:

- Lessons learned at NetApp from developing PMEM-based file-system
- Discuss how PMEM technologies are a game-changer in file-system design
- Use case: **PMFS2** over **ZUFS**

NetApp's MAX-Data is a PMEM-based Solution

"Get MAX power with Intel Optane and NetApp Memory Accelerated Data"



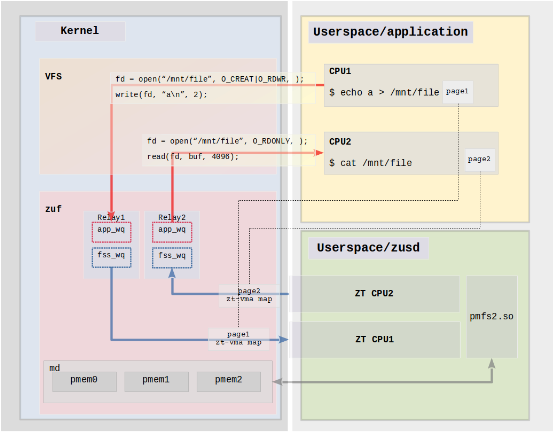
MAX-Data Overview

- Enterprise grade storage solution
- PMEM-based file-system
- Low-latency (sub 10us), high throughput (millions of IOPS)
- Snapshots, crash consistency, POSIX compliant
- Written in user-space (via **ZUFS**)

ZUFS Overview

A framework for developing PMEM-based file-system in user-space

- Optimized for PMEM devices
- Zero-copy between kernel and user-space
- Designed for modern multi-cores machines



NetApp

NetApp Open-Sourced ZUFS

- **ZUFS** is a framework for developing PMEM-based file-system in user-space
- **PMFS2** is an educational file-system over ZUFS
- Open-source:
  https://github.com/NetApp/zufs-zuf
  https://github.com/NetApp/zufs-zus
  https://github.com/sagimnl/pmfs2

Linux PMEM-based File-Systems

- **DAX** based file-system (*xfs*, *ext4*)
- **PMFS** was an attempt to implement an in-kernel PMEM file-system (discontinued)
- **NOVA** is still under development
- **PMFS2** is a re-implementation of PMFS in user-space using **ZUFS**

PMFS2 Overview

- An educational file-system
- Fundamental block/page size: 4K
- Data structures at sub-block granularity
- Sync operations at cache-lines granularity
- File mapping using radix-tree
- Space management via free-queue
- Recon/fsck upon mount (no journal)

Performance of User-Space File-System

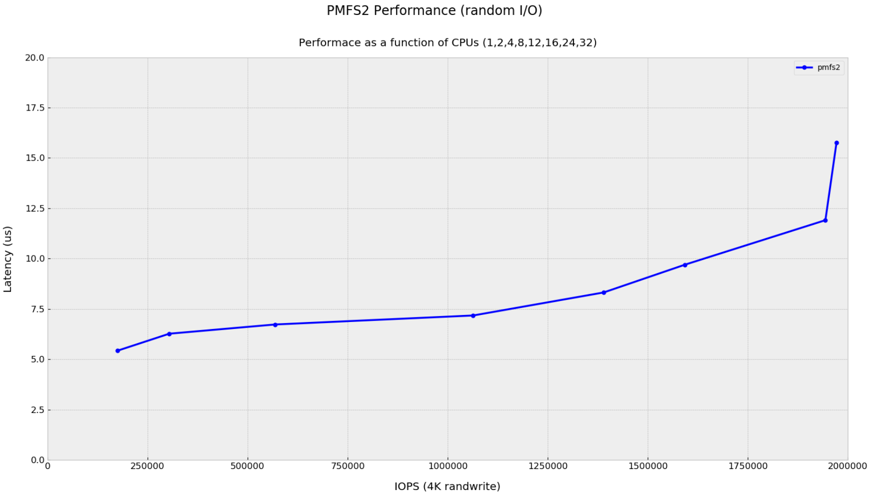
ZUFS + PMFS2

Can a user-space PMEM file-system be as good as in-kernel?

Performance Benchmarks with FIO (4K-random)

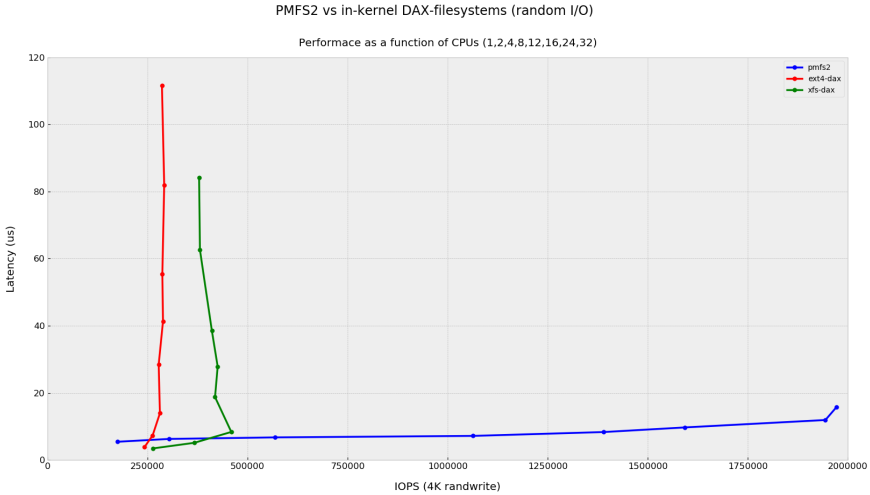
```
$ fio --name=pmfs2-bs4-jobs32
  --filename=/mnt/pmfs2/pmfs2-bs4-jobs32
  --numjobs=32 --bs=4096 --size=33554432
  --fallocate=none --rw=randwrite --ioengine=psync
  --sync=1 --direct=1 --time_based --runtime=30
  --thinktime=0 --norandommap --group_reporting
  --randrepeat=1 --unlink=1 --fsync_on_close=1
  --minimal
```

PMFS2 Performance (4K-random)

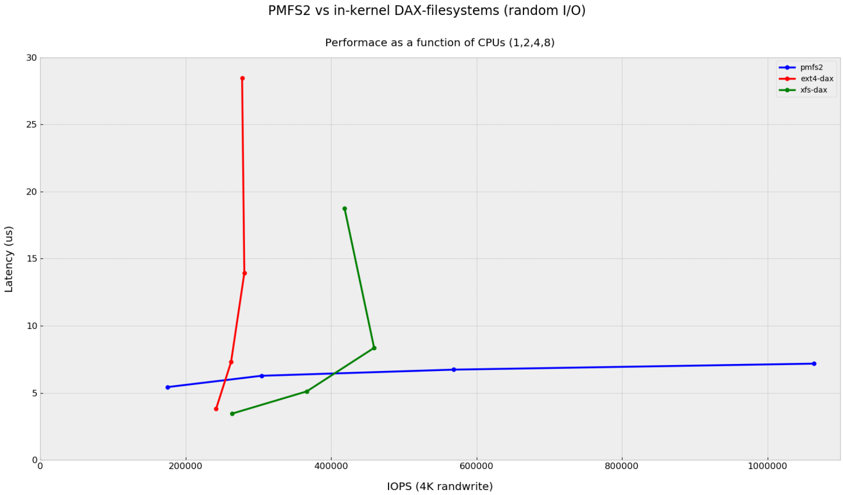


NetApp

PMFS2 vs kernel DAX-filesystems (4K-random)



PMFS2 vs in-kernel DAX-filesystems (Zoom-in)



NetApp

Key Observations on PMEM technology

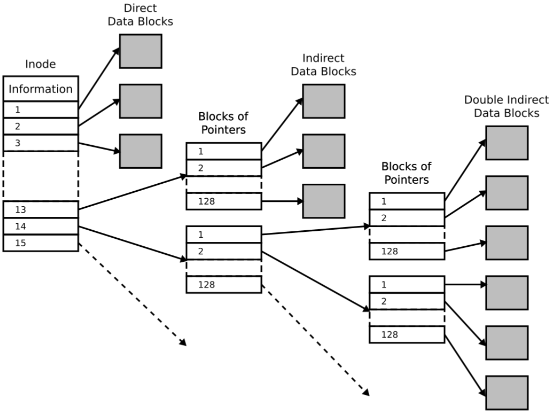
PMEM is a paradigm shift in file-system design!

Lessons Learned from ZUFS and PMFS2

- Meta-data sync at cache-line granularity (e.g., inodes, dentries)
- Alternative file-mapping (radix-tree vs traditional B-tree)
- Soft updates are preferred over journal
- Stay on same CPU when possible
- Bypass page-cache, avoid extra copies

Example 1: Regular File Mapping

- A regular file implements mapping from virtual-address to physical address
- *UFS* used indirect inode pointer pointer structure
- Many modern file-systems use variant on *B-tree*
- PMEM allows using alternative data-structure (*radix-tree*)



Regular File Operations

Modern files are complex:

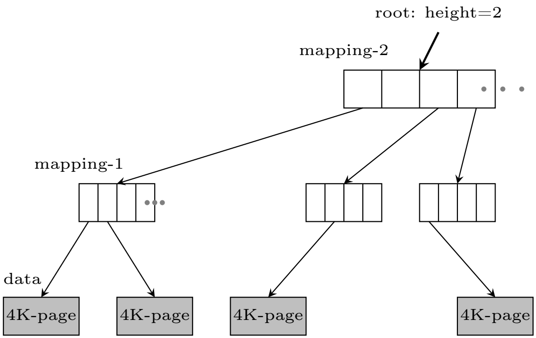
- `read, write, pread, pwrite`
- `truncate, ftruncate`
- `lseek(SEEK_DATA, SEEK_HOLE), fiemap`
- `fallocate (FL_PUNCH_HOLE, FL_COLLAPSE_RANGE)`
- `copy_file_range, ioctl(FICLONE)`
- `stat, fstat, mmap`

NetApp

File Mapping on PMEM

Hierarchical address-mapping with radix-tree

- Data leaves: 4K-page
- Mapping: 512 pointers per page
- Mapping level-1: 2M
- Mapping level-2: 1G
- Root-pointer and height on inode

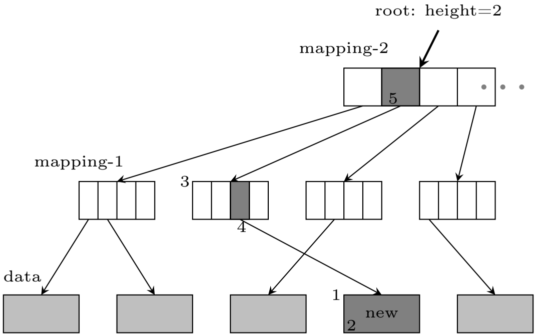


File Write

Write as a sequence of crash consistent operations

```
pwrite(fd,"new",3,0x202000)
```

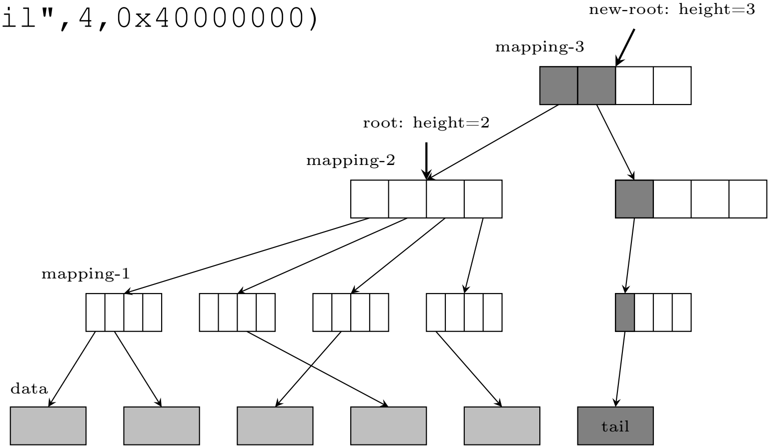
1. Allocate new data block, zeroed
2. Copy data to newly allocated block
3. Allocate new meta block, zeroed
4. Set pointer on mapping level-1
5. Set pointer on mapping level-2
6. Update root



File Write

Increase tree height

```
pwrite(fd,"tail",4,0x40000000)
```



Example 2: Crash Consistency

Problem: how to maintain file system integrity in the event of a crash?

- Common method: journal
- PMEM file-system allows to revive the old method of soft-updates
- Requires careful crafting of meta-data and order of operations

Crash Consistency on PMEM File-Systems

(Not fully implemented on PMFS2, yet)

- PMEM speed is closer to DRAM than SSD
- Fast enough to allow entire file-system re-scan upon mount
- Scan file-system from root, declare unreachable pages as free
- Extra bits for meta-data recovery state
- No need to have a journal

Follow MAX-Data via NetApp’s Blog

https://blog.netapp.com/memory-accelerated-flexpod

Questions?