An Evaluation of DAOS for Simulation and Deep Learning HPC Workloads

Luke Logan, Jay Lofstead*, Xian-He Sun, Anthony Kougkas
llogan@hawk.iit.edu, gflofst@sandia.gov, sun@iit.edu, akougkas@iit.edu

CHEOPS@EuroSys'23

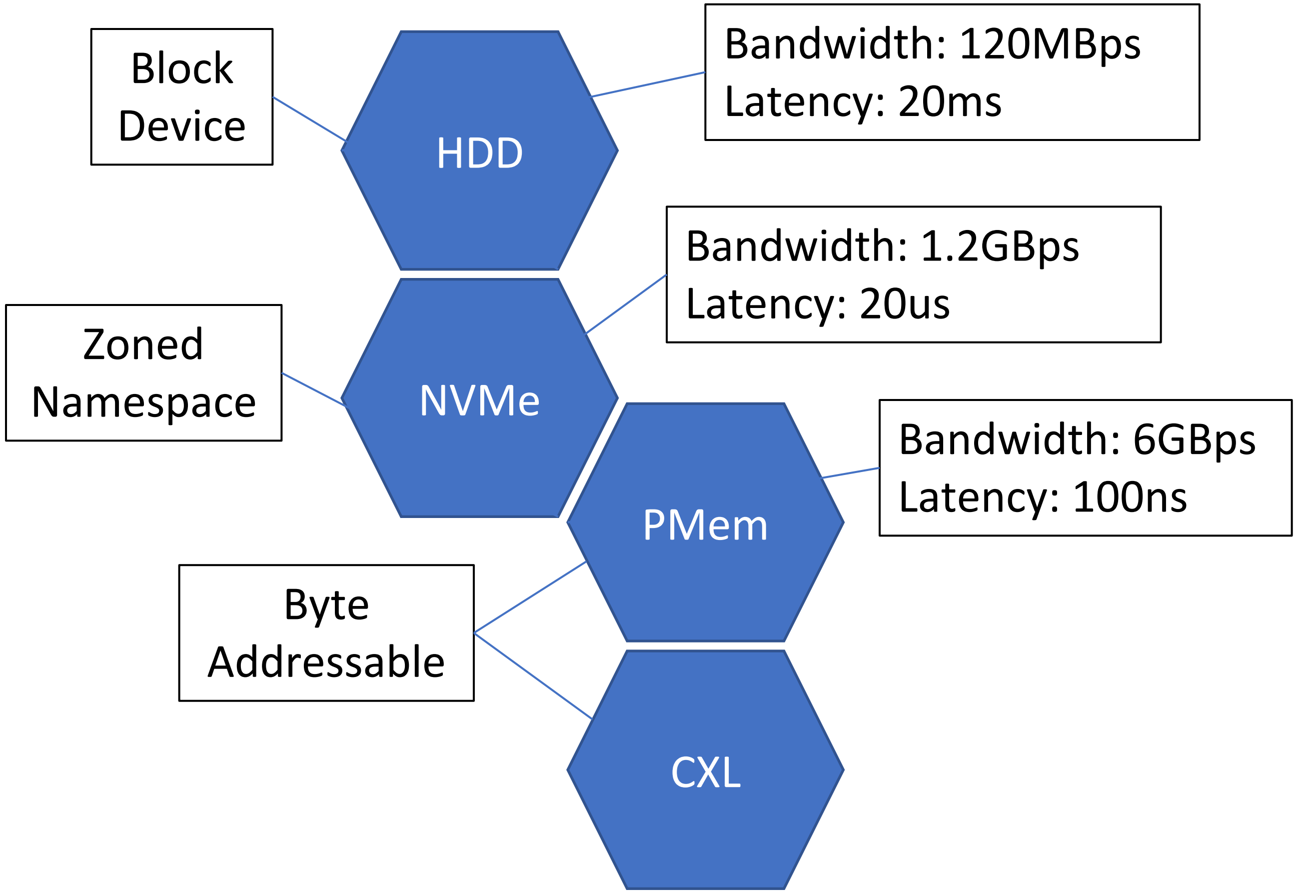
ILLINOIS INSTITUTE OF TECHNOLOGY

Scalable Computing Software Laboratory

Sandia National Laboratories

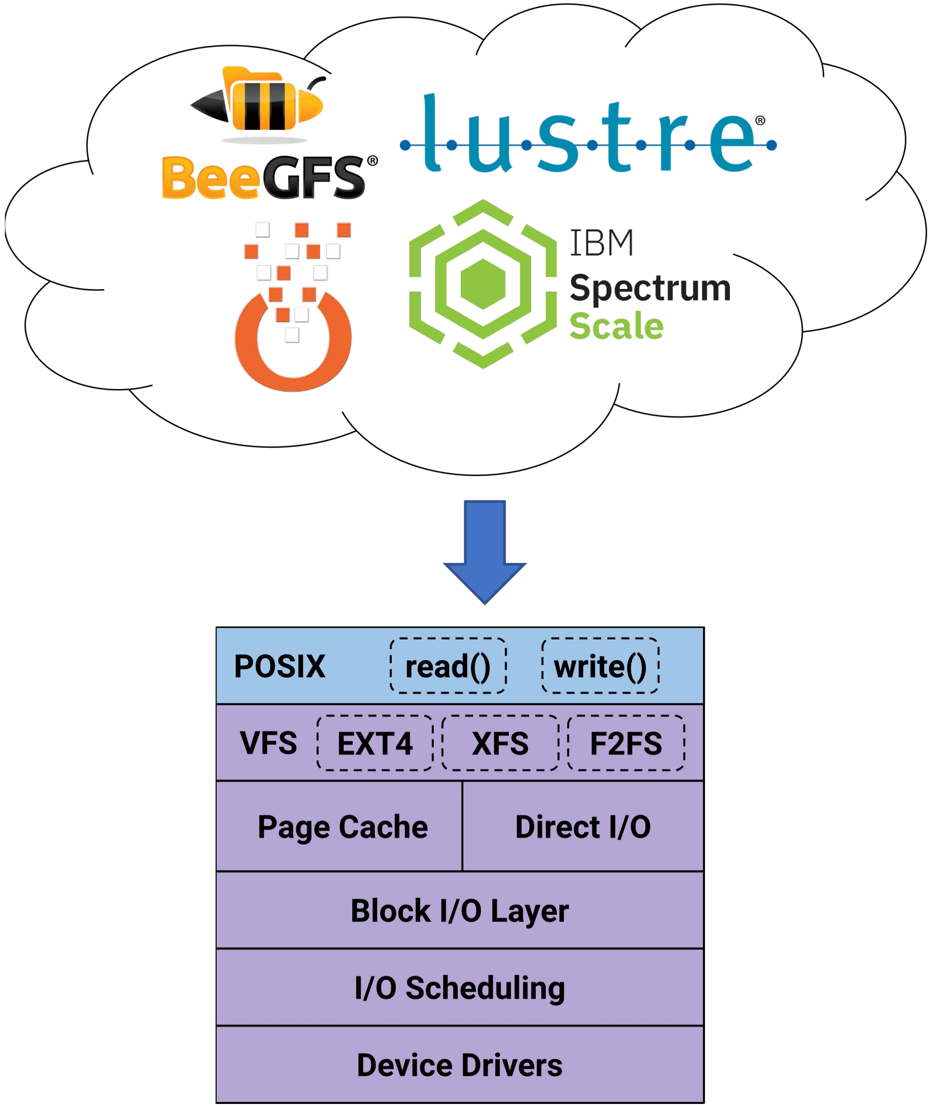
Rapid Evolution of Storage Hardware

- Order of magnitude performance improvements per generation
- New interfaces being exposed to enable hardware-specific optimization



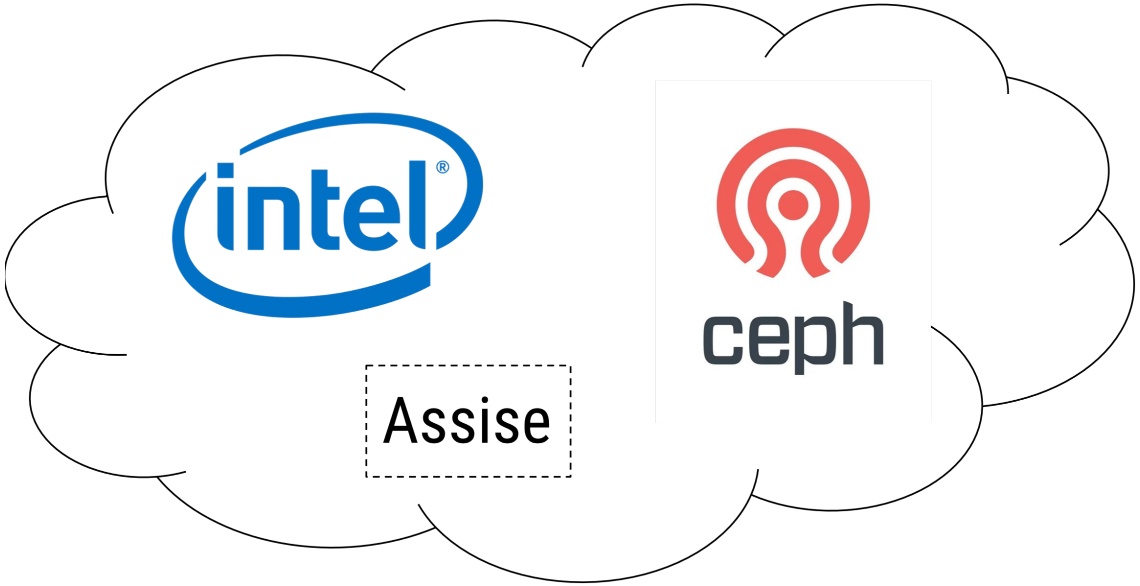
Traditional Parallel Filesystems

- Used in majority of HPC sites
- Designed for hard drives
- Rely on monolithic kernel I/O stack
    - Context switch, interrupt
- **Do not take full advantage of new hardware interfaces!**



Rebuilding The Storage Stack

- New storage stacks emerging which utilize hardware-specific APIs
  - E.g., DAOS, CephFS
- Optimization for (meta)data storage
  - Bypass the kernel
  - Reduce context switching & interrupting



But, what's the impact?

The value of optimizing storage stacks for modern hardware is not well-understood

- Many single-node evaluations (not distributed)
- Many emulate PMEM using DRAM
- Old software versions (e.g., DAOS 1.0)
- Synthetic workloads

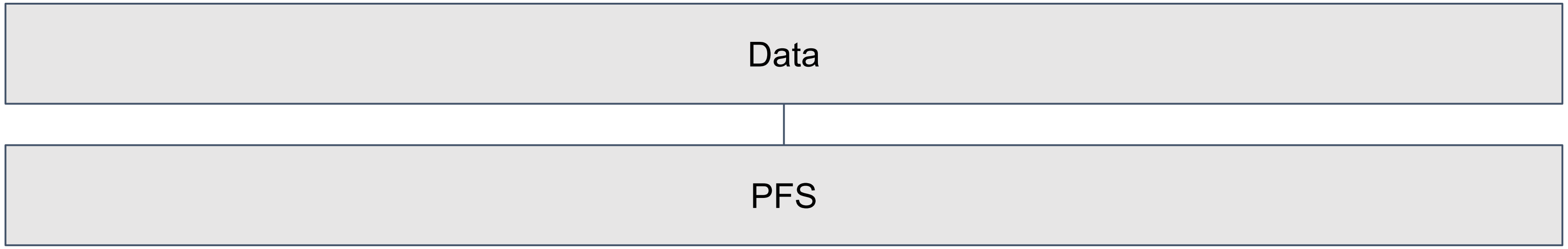
Our Goal

We quantify the performance benefit of utilizing hardware-optimized vs traditional storage stacks for real deep learning and simulation workloads

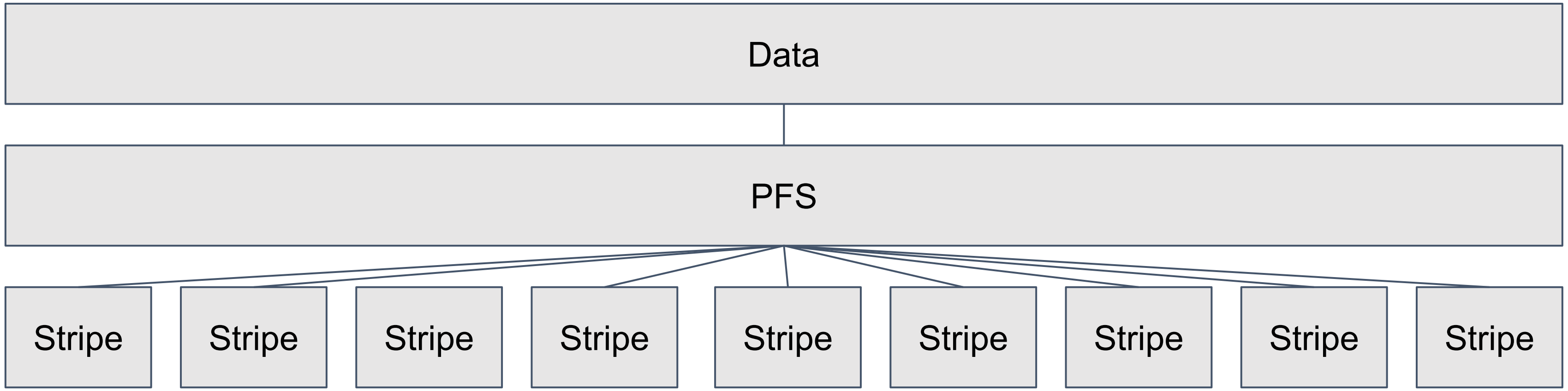
Hardware-Optimized	Traditional
DAOS	OrangeFS
	BeeGFS

Background

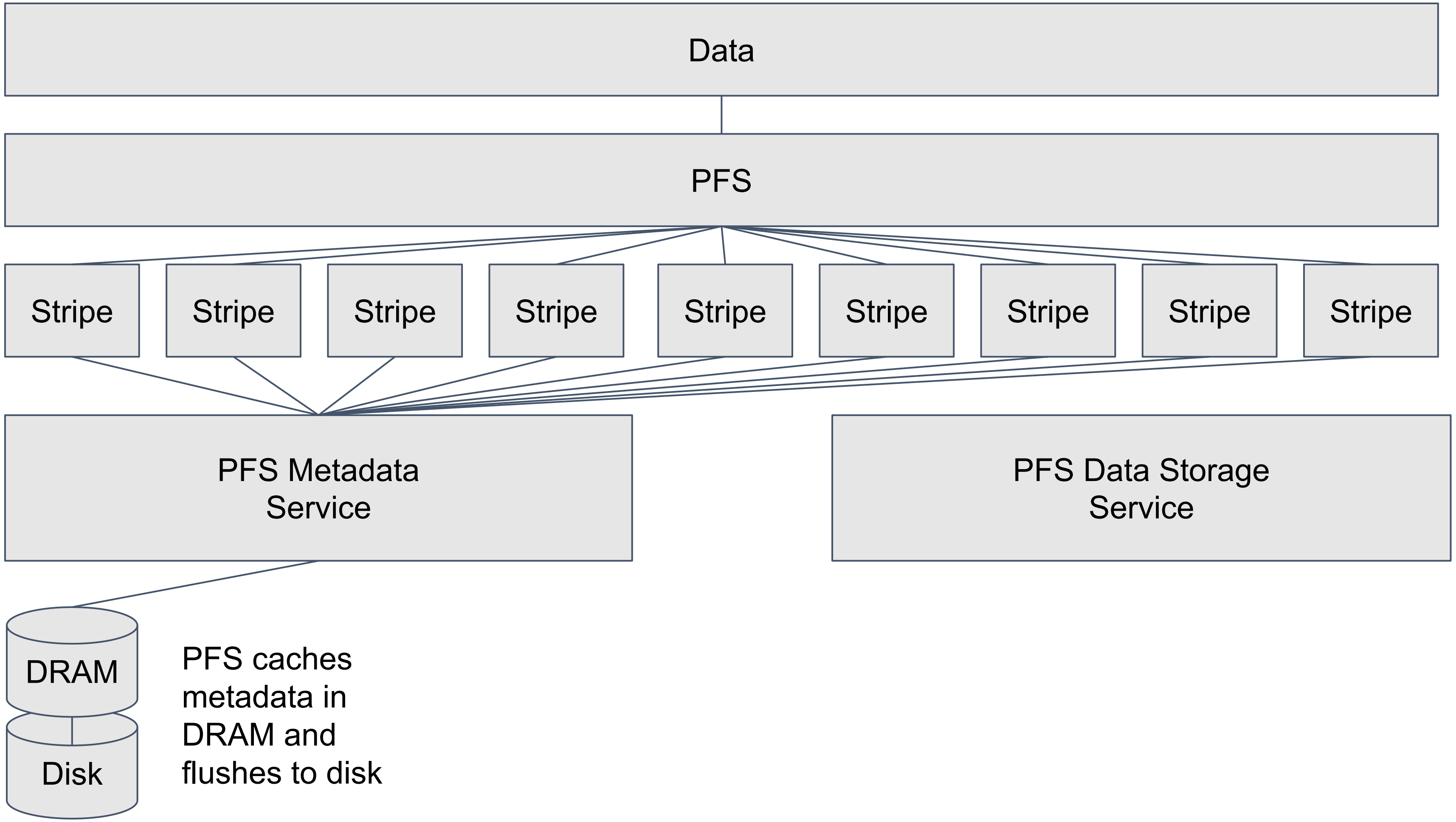
PFS (1): Data Arrives at PFS



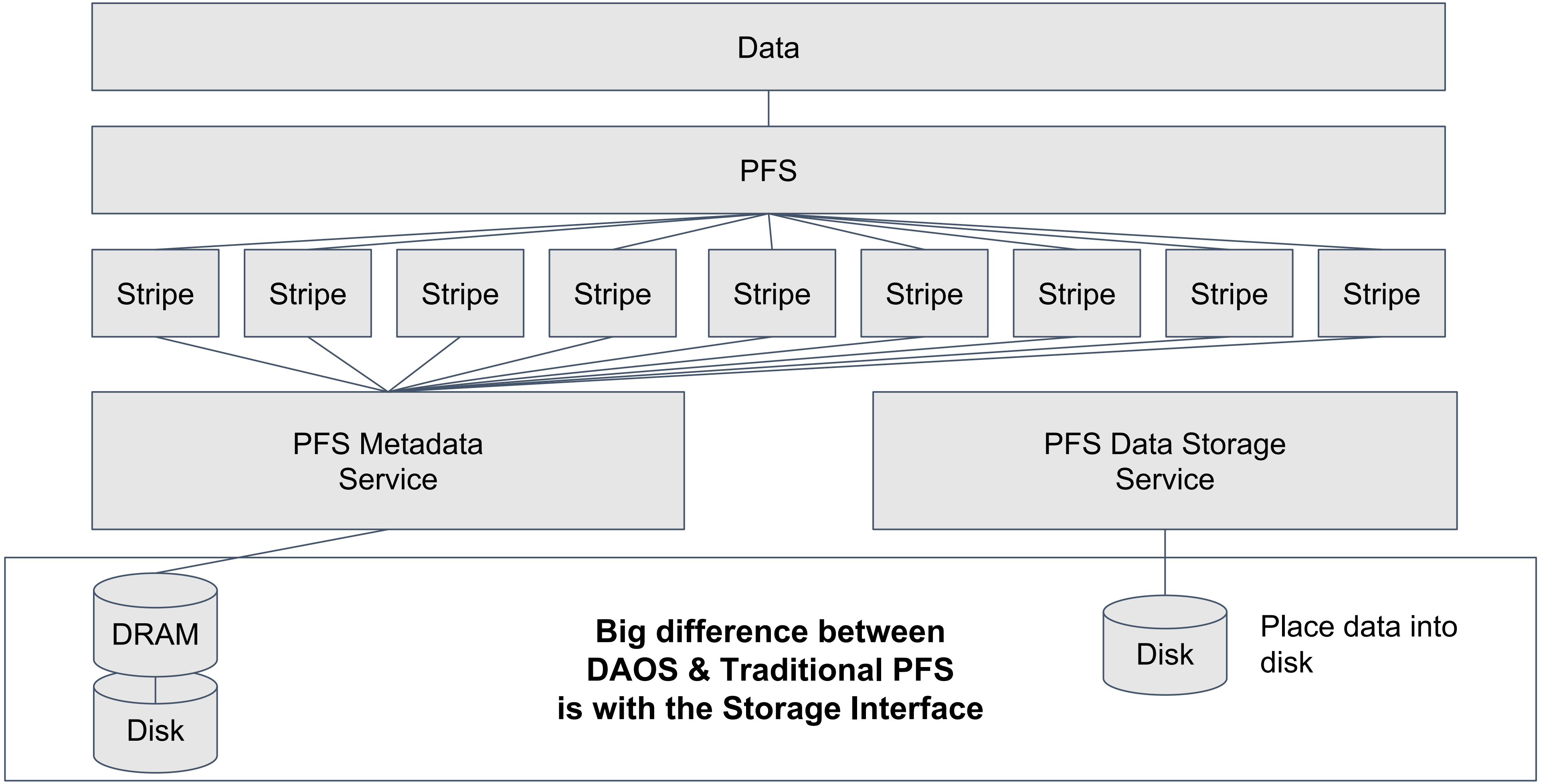
PFS (2): PFS Divides into Stripes



PFS (3): PFS Registers Stripes with MD



PFS (4): PFS Stores Data



Big difference between DAOS & Traditional PFS is with the Storage Interface

Storage Interfaces

- Filesystem (e.g., EXT4, F2F2)
  - System calls + interrupts are used to interact with a filesystem
  - Work for any device type
  - Used by traditional PFS to store data
- Storage Performance Development Kit (SPDK)
  - A driver for interacting with NVMes without going through kernel
  - Can be used by DAOS to persist data to NVMe
- Memory Mapping + Direct Access (DAX)
  - Treat a storage device as if it were memory
  - For PMEM and CXL devices, can interact directly using CPU load/store operations, without going through kernel
  - Can be used by DAOS to persist data to PMEM

Evaluation

Testbed

- 4 nodes
- **CPU:** 96 cores / 192 threads
  - 2x Intel(R) Xeon(R) Gold 6342 CPU@2.80GHz
- **Network:** 100GBe, IPoIB
- **PMEM:** 2TB
  - 8x Intel Optane DC Persistent Memory (256GB per module)
- **NVMe:** 64TB
  - 16x 4TB NVMe

Software

- **OS:** Centos8
  - Kernel 4.18
- **DAOS:** 2.1.104-tb
- **OrangeFS:** 2.9.8
- **BeegFS:** 3.7.1
- **Io500:** isc’22
- **MPI:** mpich 3.3.2

Synthetic Workloads (IO500)

Experimental Setup (1)

OrangeFS + BeeGFS

- Default config of OrangeFS
- Stripe Size: 64KB
- Filesystem: EXT4
- Co-locate metadata + data servers
    - Not typical in HPC, but hardware limits

DAOS (NVMe)

- Cache: 50GB PMEM
    - As low as it would go
- 5TB NVMe
- 1 dedicated core

DAOS (PMEM)

- 6TB PMEM

DAOS (DRAM)

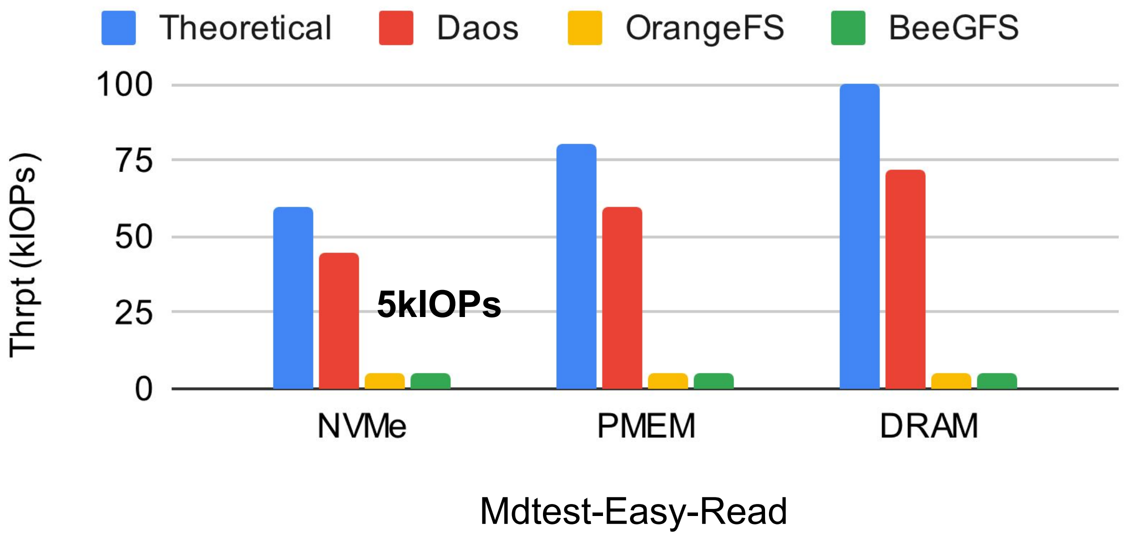
- 3TB DRAM

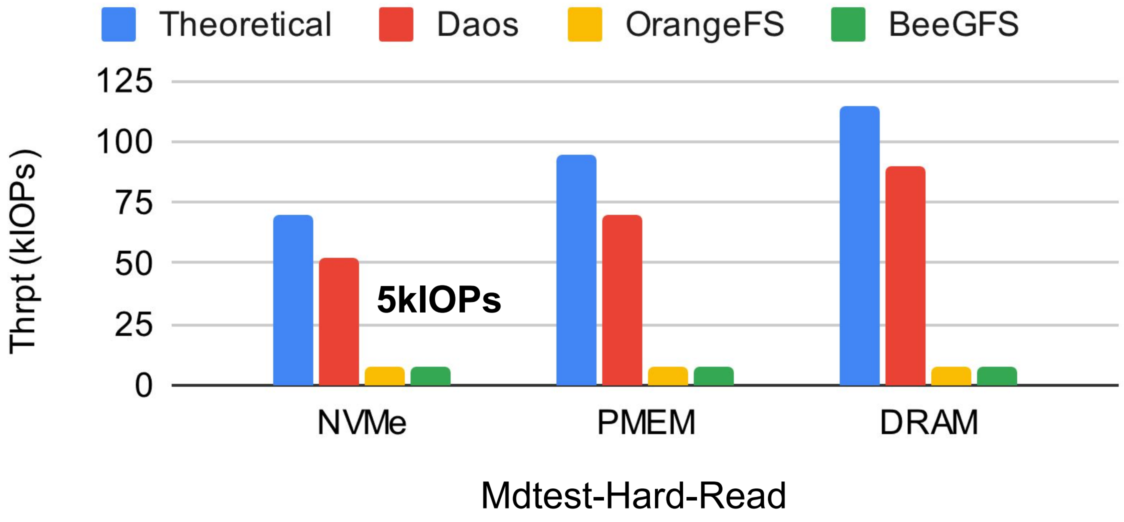
Experimental Setup (2)

- IO500 was executed with 512 total processes
  - (128 processes per node)
- We measure the theoretical throughput of the storage hardware using dd on the device file for NVMe & PMEM
- We run each workload for 5 minutes

IO500 (Mdtest, 1)

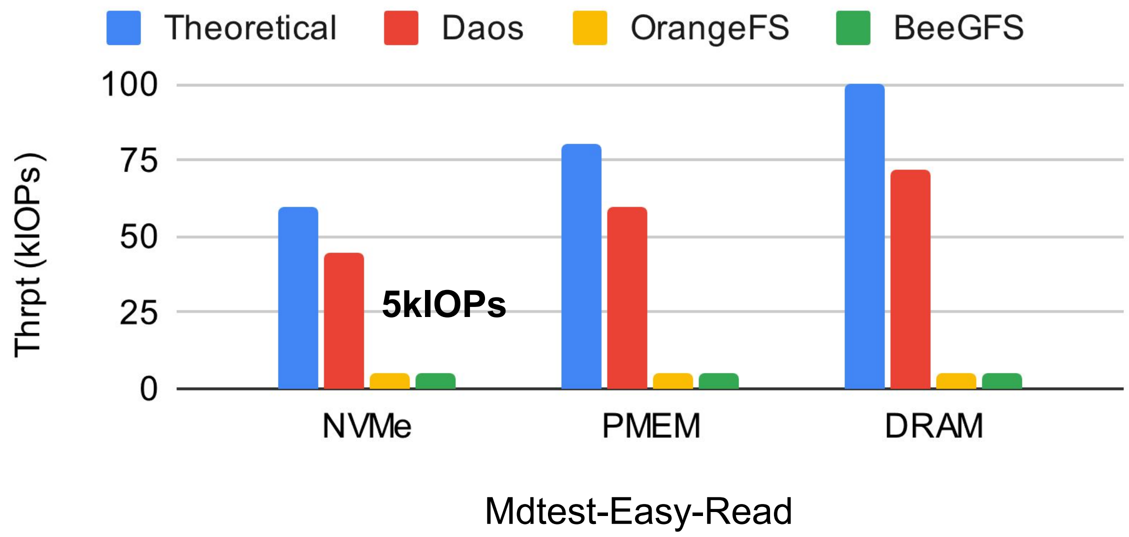
- Stress metadata throughput
  - mdtest-read: open / close for existing files, listing directories, etc.
  - mdtest-write: create new files, remove directories, etc.
- Metadata is cached in memory, but must be updated in storage eventually

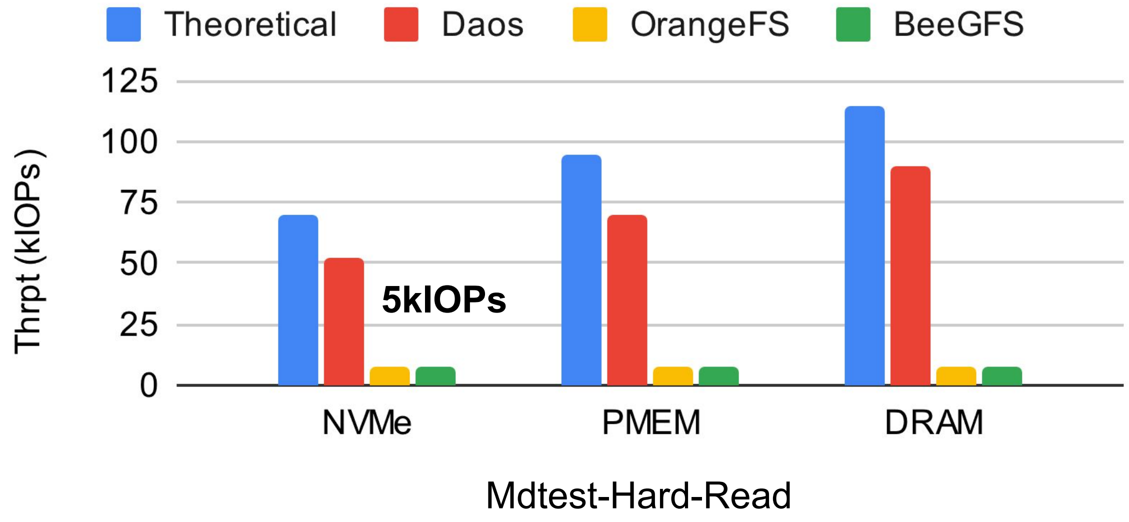




IO500 (Mdtest, 2)

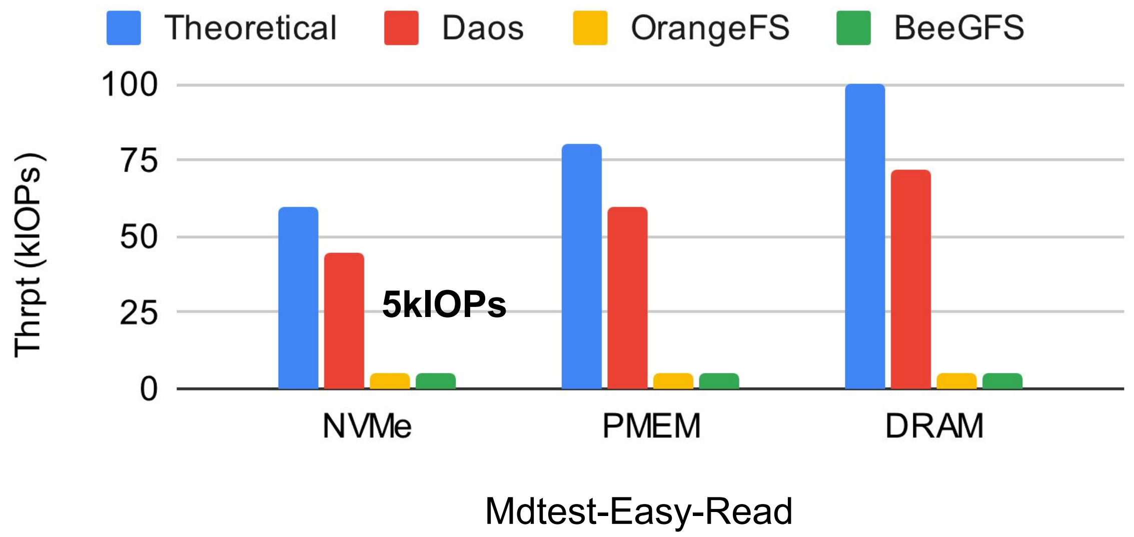
- BeeGFS/OrangeFS are 4-5kIOPs
- DAOS is 40-70kIOPs
- DAOS is **15x** faster, regardless of hardware type
  - BeeGFS / OrangeFS use interrupts + context switching for metadata ops
  - DAOS stores metadata primarily in PMEM or DRAM
  - DAOS uses API interception

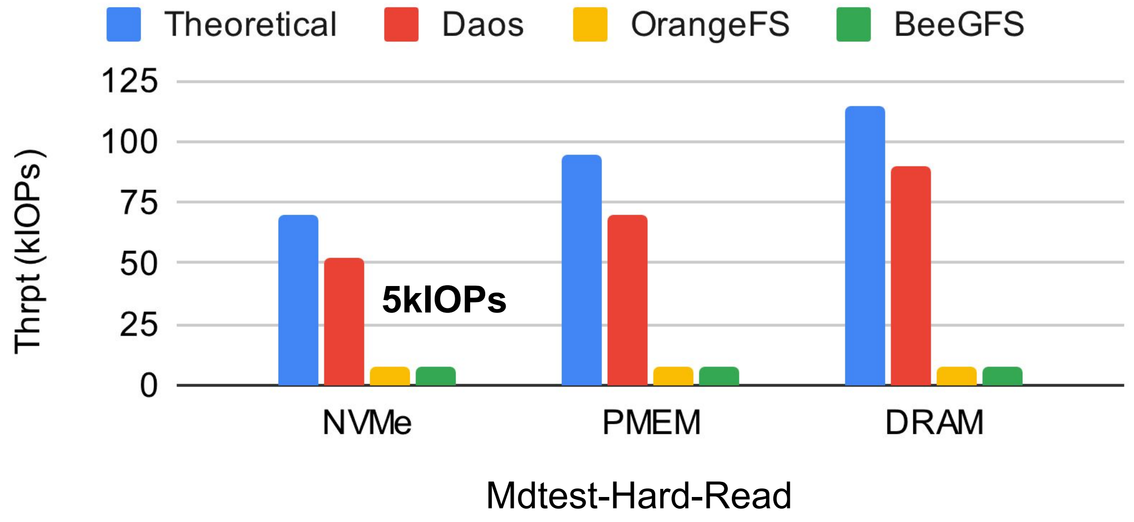




IO500 (Mdtest, 3)

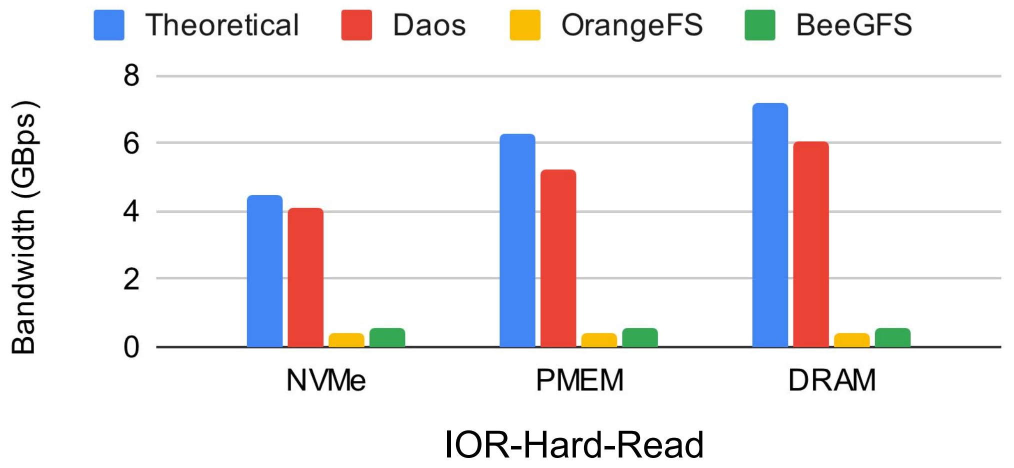
- DAOS is within 20-30% of the theoretical hardware throughput for NVMe, PMEM, and DRAM
  - Software overhead low considering hardware speed
- mdtest-write experiments had similar results

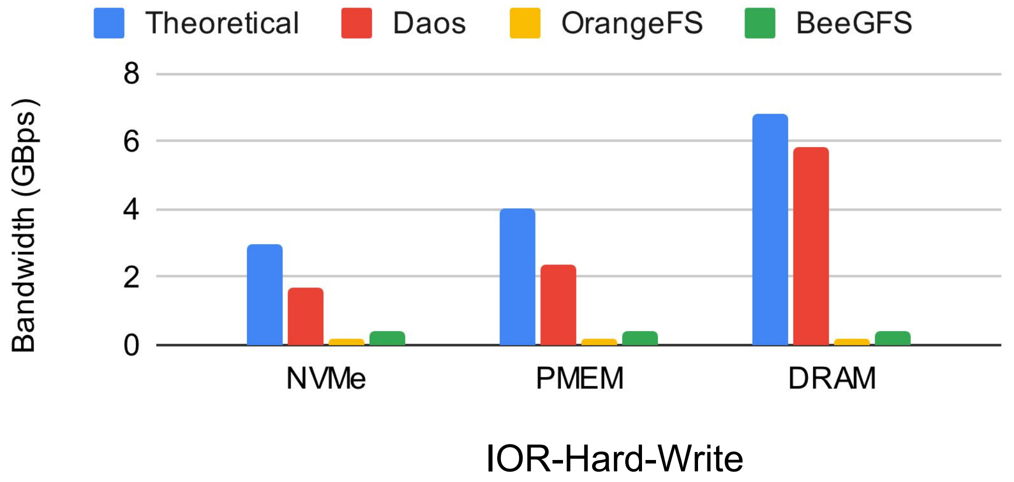




IO500 (IOR-Hard)

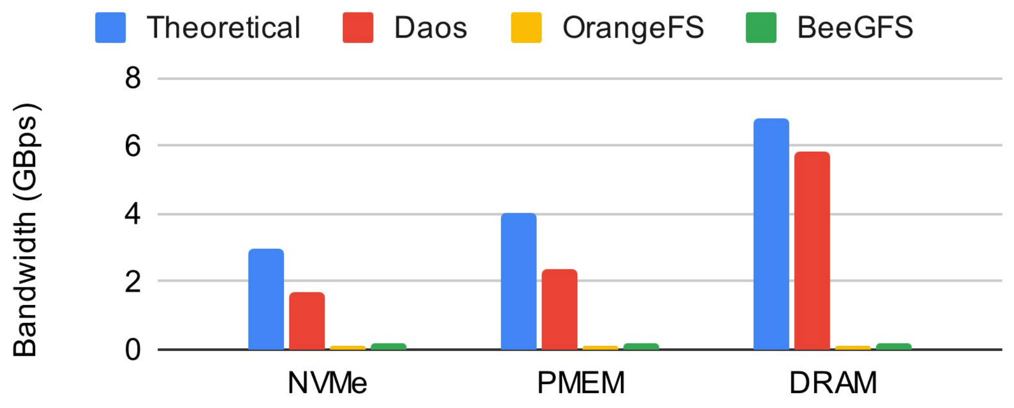
- Worst case for PFS
- Small, unaligned I/O
- DAOS was 8x faster
    - Small I/O creates a lot of metadata
    - Small I/O creates a lot of I/O requests
    - On BeeGFS / OrangeFS, lots of interrupting due to reliance on EXT4
    -

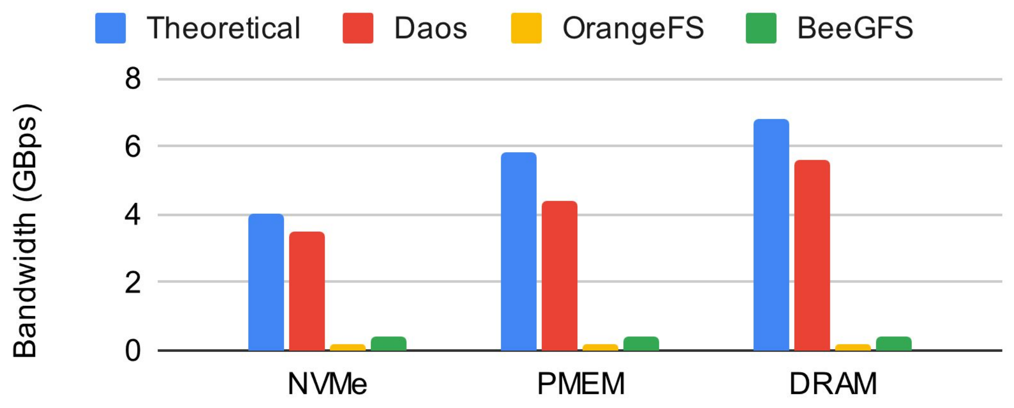




IO500 (IOR-Easy)

- Large-sequential I/O
- Best case for a PFS
- Still outperforms traditional stack by 10x
- Significant metadata overhead for managing stripes
  - 1 metadata op per 64KB
  - From mdtest, we saw 15x slower metadata performance





Deep Learning Workload

Experimental Setup

Cosmic Tagger

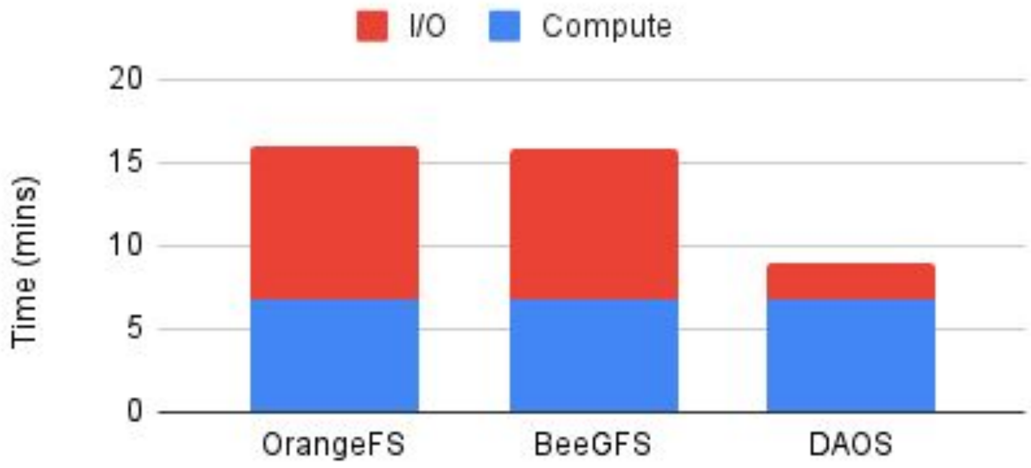
- Convolutional Neural Net for separating neutrino pixels
- 430,000 samples in the dataset
- 450GB size
- 20 – 50KB I/O sizes

DAOS

- 50GB of PMEM
- 5TB of NVMe
- 1 server core
- SPDK as storage backend

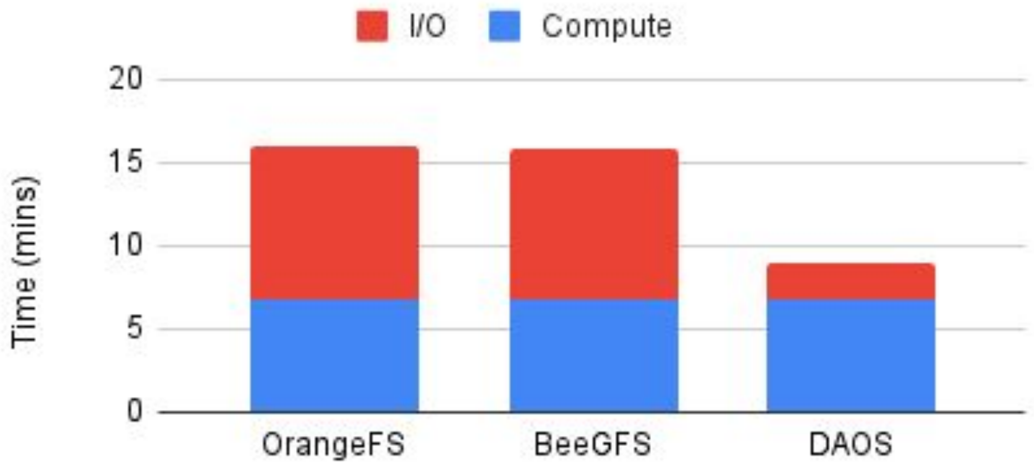
We compare CosmicTagger performance over DAOS, OrangeFS, and BeeGFS using NVMe

Cosmic Tagger



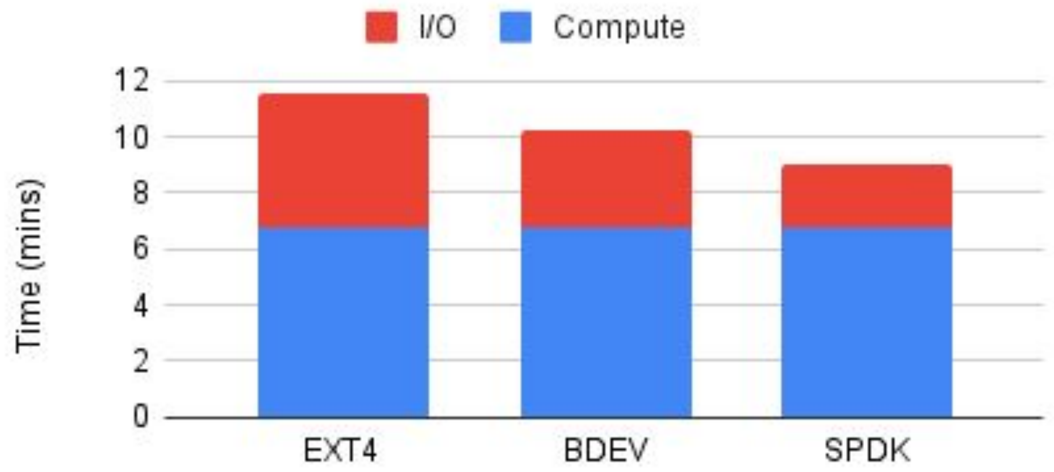
- Roughly half of training time was spent in I/O when using OrangeFS and BeeGFS
- About 20% of training time was spent in I/O when using DAOS

Cosmic Tagger



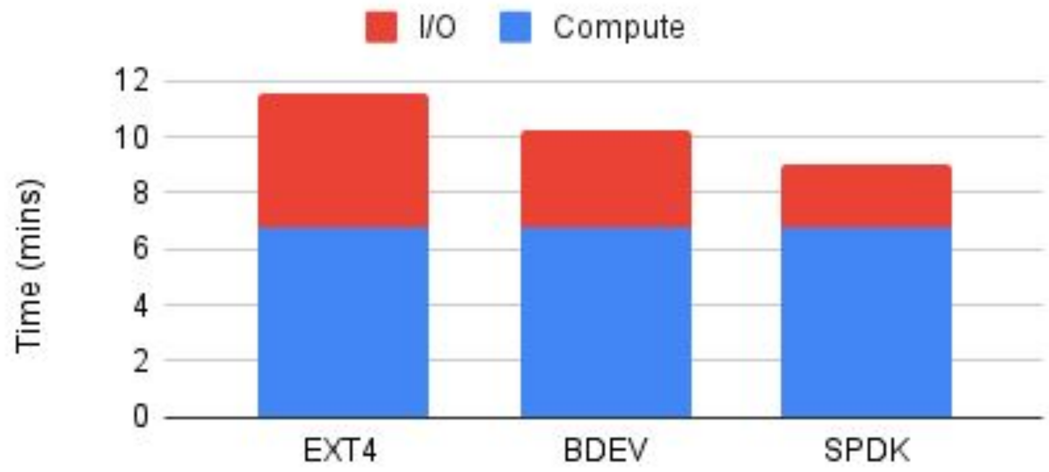
- Nearly a 2x performance improvement in training time
  - I/O time went from 9 min -> 2.5min
- CosmicTagger only overlaps ~6% of compute with I/O
- Where does I/O performance difference come from?

Cosmic Tagger: Impact of Storage Backend



- How much impact does the storage backend have on performance?
- We compare EXT4 vs Device File (BDEV) vs SPDK
  - EXT4 is a kernel-level filesystem
  - Device File has no filesystem, but still goes through kernel
  - SPDK is in an NVMe driver which bypasses the kernel completely

Cosmic Tagger: Impact of Storage Backend

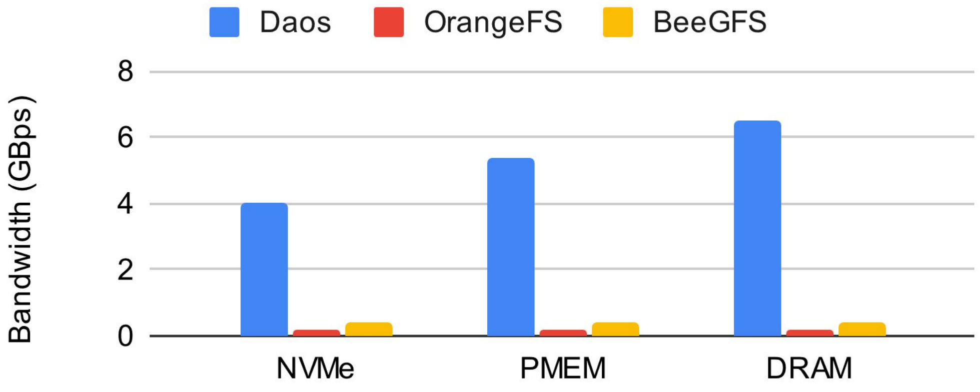


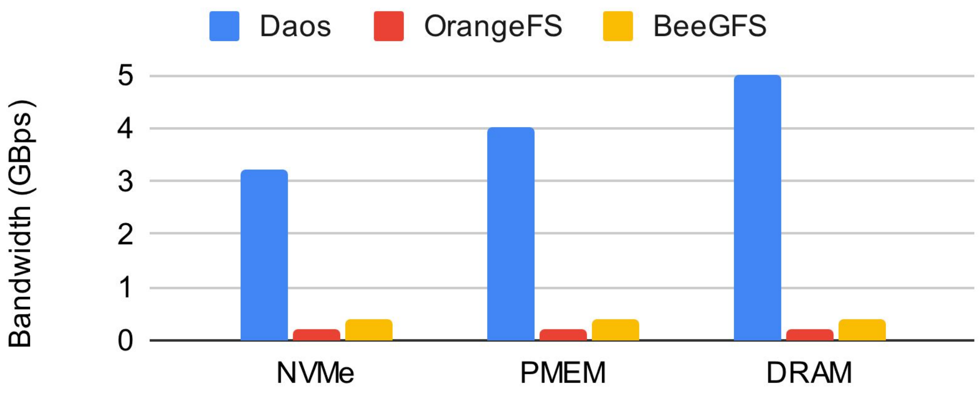
- Using SPDK resulted in 2x improvement in I/O time:
  - 5 min -> 2.5 min
- 95% of I/O touches NVMe due to low PMEM capacity
- SPDK avoids kernel overheads completely
- Remaining differences between DAOS and OrangeFS/BeeGFS is due to metadata management overheads

Simulation Workload

VPIC + BD-CATS

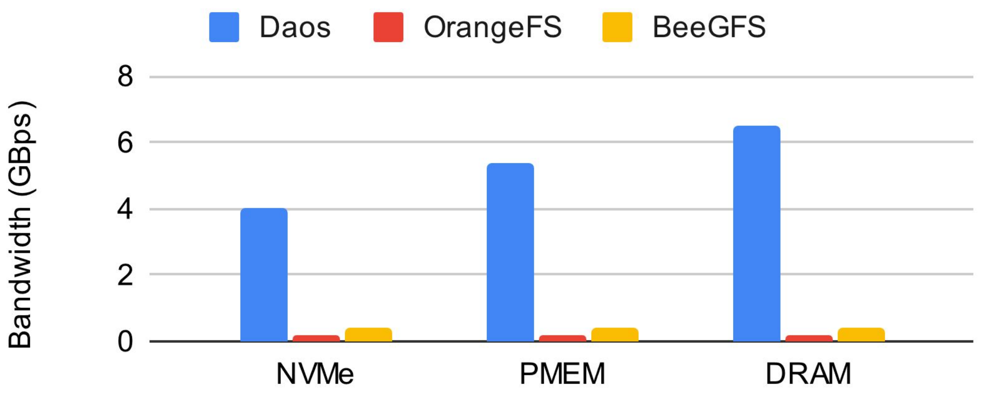
- VPIC
  - Plasma simulation code which simulates particles
  - Checkpoint-restart, write-only
  - 30GB / checkpoint
  - 16 checkpoints (480GB in total)
- BD-Cats
  - Particle clustering code via KMeans
  - Clusters the data that VPIC produces
  - 480GB of I/O in total again

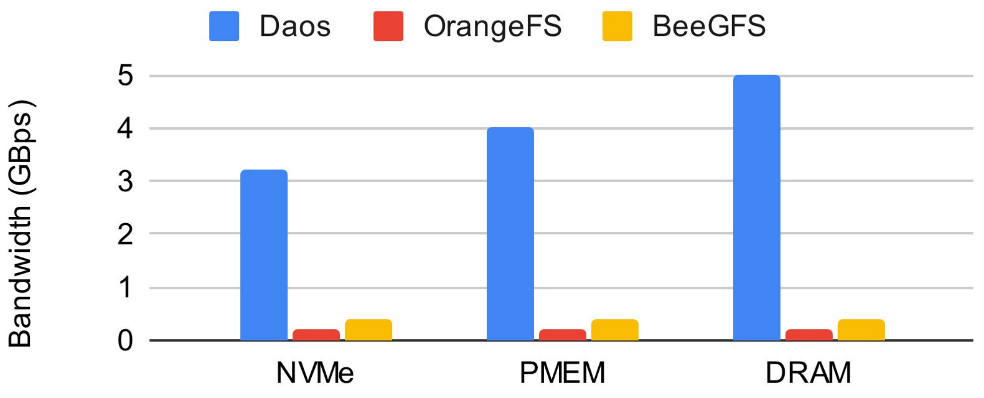




VPIC + BD-CATS

- Overall, 6x faster than others
  - Similar to IOR-Easy
- High metadata cost for stripes
  - 480GB / 64KB = 8 million metadata operations
  - Roughly 4-5kIOPs for each MD operation, 15x slower than DAOS





Discussion

Summary

- Removing kernel stack I/O overheads improves data and metadata ops on modern hardware significantly
- Metadata performance is a significant factor in data-intensive workloads
- Real workloads get observable benefits from the improved I/O & metadata stack

Limitations

- **Small Scale:** Network overheads were minimal
- **Modern Hardware:** Many clusters use modern hardware for temporary storage, not persistent.
    - We didn't test-multi-tiering between PMEM, NVMe, and HDD
    - We didn't test over traditional HDD-only clusters
- **Stripe Size:** Stripe sizes are configurable. We went with the defaults. A 1MB stripe size would improve the IOR easy and VPIC/BD-CATS workloads
- **Storage System Variety:** Many other PFSes used in production, such as Lustre + CephFS. In the future, would be interesting to see how DAOS compares to more systems

Conclusion

Conclusion

- We conducted numerous benchmarks of DAOS over modern hardware in a distributed setting
- DAOS outperforms traditional storage stacks by as much as 15x on modern hardware
- The performance improvement is due to hardware optimization in metadata and data storage


thank
you