Karlsruher Institut für Technologie

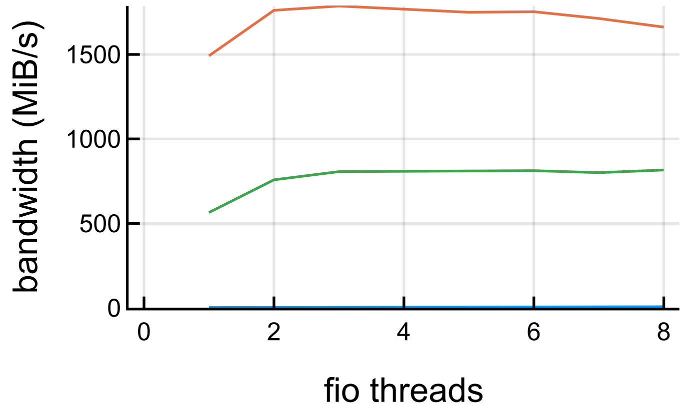
Towards Less CPU-Intensive PMEM File Systems

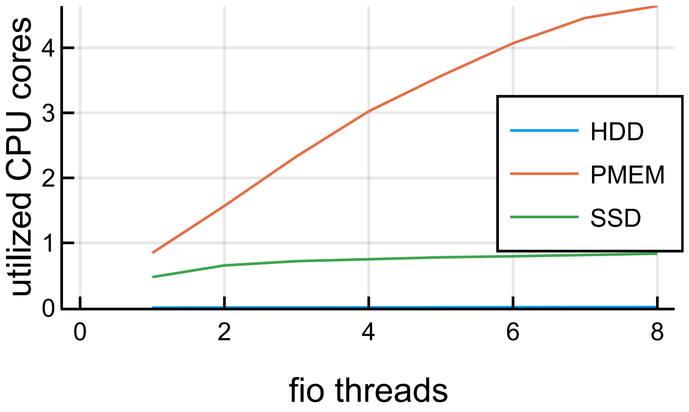
Lukas Werling, Christian Schwarz, Frank Bellosa

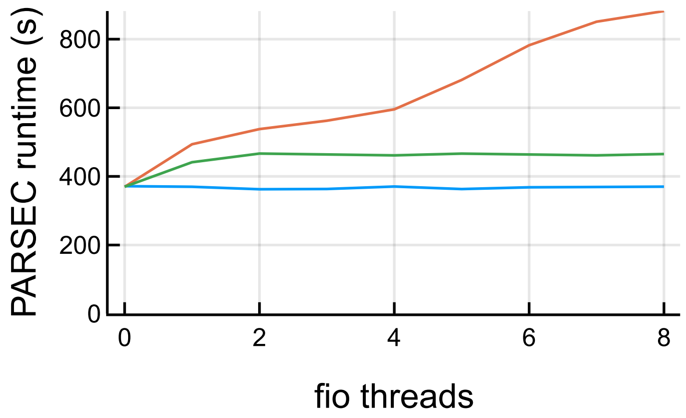
Motivation

I/O: fio random write on ext4, 1-8 threads

CPU: PARSEC, 8 threads





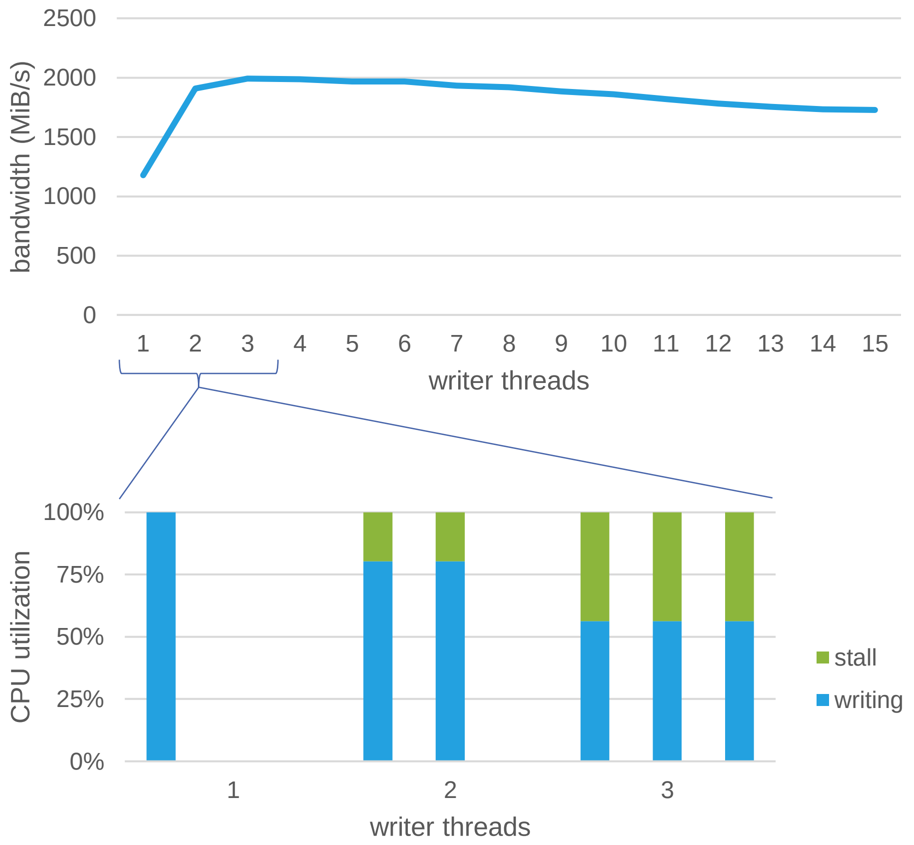


Writes to Optane do not scale well and slow down unrelated processes.

Our solution: Copy-offloading to reduce CPU utilization.

Background: Persistent Memory

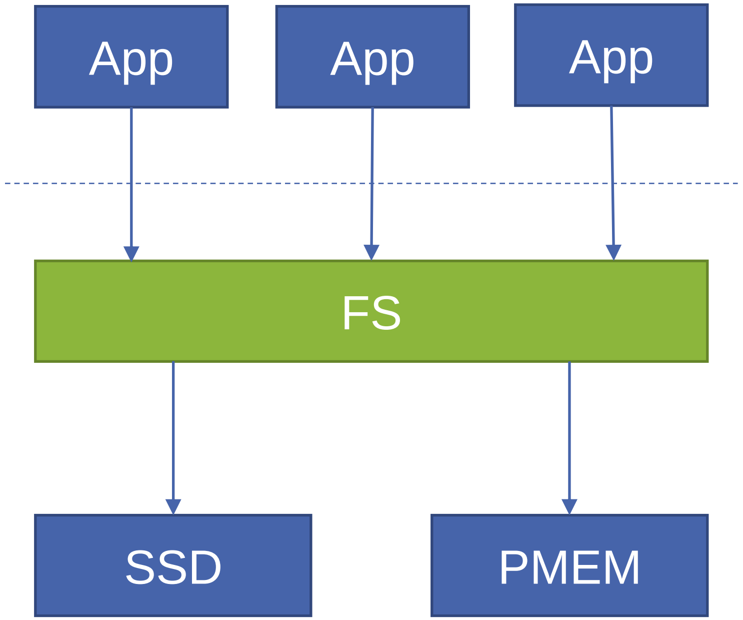
- Limited parallel write bandwidth
- Access via load/store instructions
  - Synchronous access
  - Stall if PMEM is not ready (wasted time)
- PMEM takes away control from the OS
  - I/O scheduling by microarchitecture!
  - Cannot switch task while waiting for I/O



I/O bound processes on PMEM are always also CPU bound.

File System Expectations

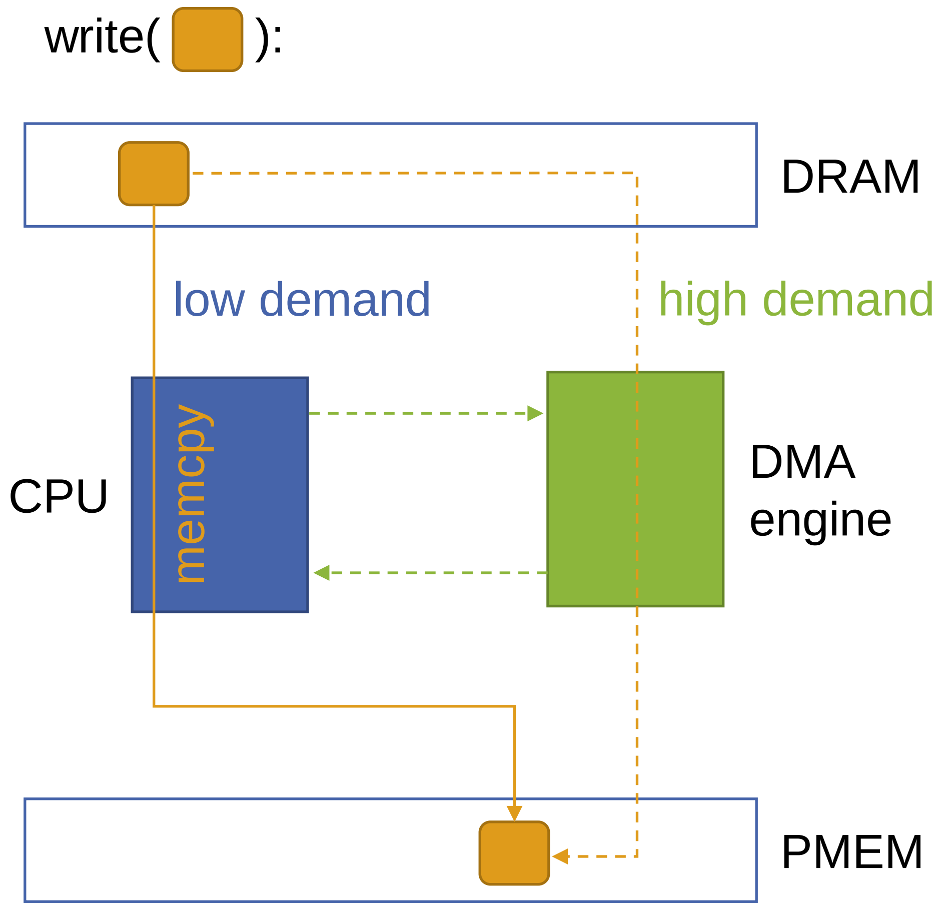
- Best Practices for Optane (Yang et al.)
  - “Limit the number of concurrent threads accessing an Optane DIMM.”
- FS: Arbitrary workloads, unrelated applications
- Simple solution: Semaphore
  - High overhead with rising number of threads



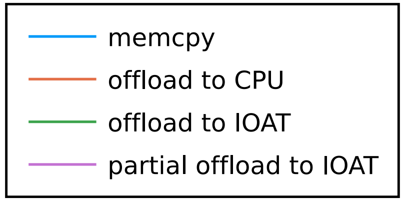
Applications expect efficient FS independent from underlying storage.

Our Solution: Copy Offloading

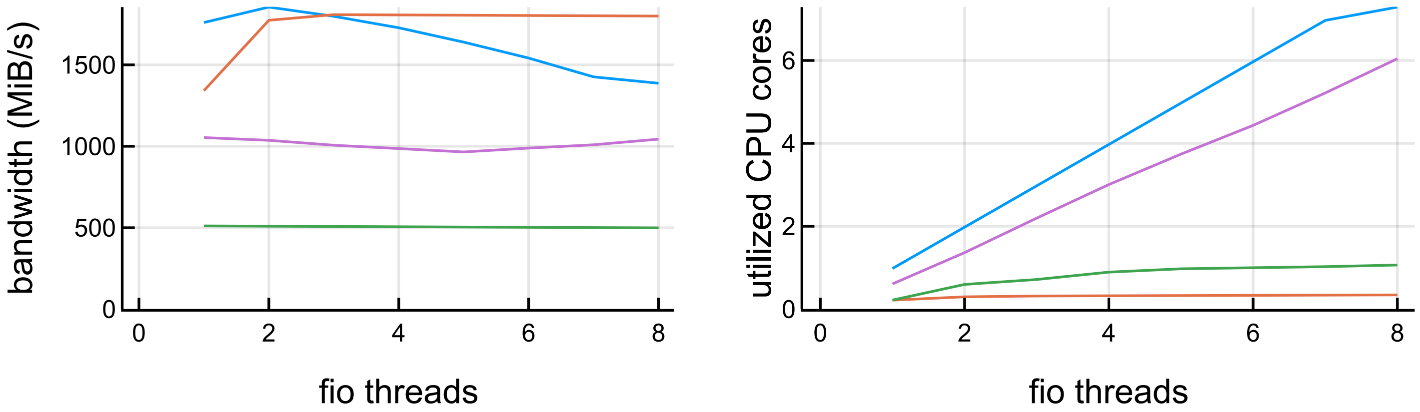
- PMEM FS: DRAM-to-PMEM memcpy
- Our idea: perform off-CPU copy
  - DMA engine (e.g., Intel IOAT)
  - Prototype: isolated CPU core
- Write-bandwidth accounting
  - Preserve low latency if possible
  - Switch to asynchronous copy when reaching bandwidth limit
  - Never offload small copies



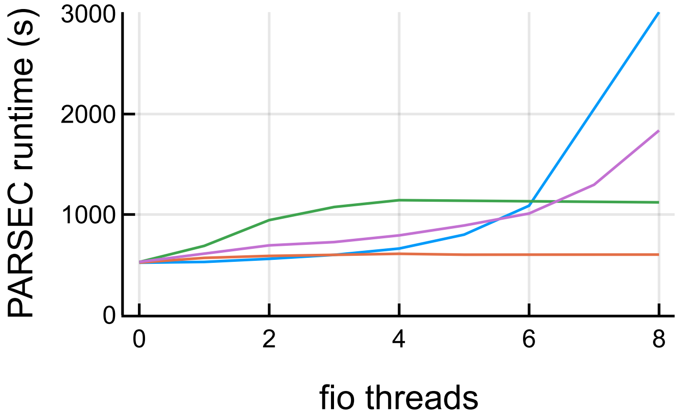
Evaluation



I/O: fio random write on NOVA, 1-8 threads



CPU: PARSEC, 8 threads



Approach looks promising, but needs better hardware support.

Conclusion

- Problem: Limited parallel write bandwidth to PMEM
  - Expensive on-CPU waiting
  - Unacceptable for file systems
- Solution: Copy-offloading
  - Detect high demand
  - Switch to asynchronous copying
- Future work
  - Other DMA devices (e.g., GPU)
  - Accounting for DAX-mmap