Lustre on Flash

Flash is different.

- Pros (vs spinning disk at same cost):
  - Greater bandwidth
  - Lower latency (~100x)
    - ~100 microseconds vs ~10 milliseconds
- Cons:
  - Lower capacity
  - Lower lifetimes/endurance
  - Logical vs physical block size issues (read-modify-write, trim, etc)

Implications for filesystems

- What do you mean by filesystems?
- Significant implications for on disk filesystems and block I/O subsystems
- Trim support and other issues are important
- Controller and flash improvements mitigate some of this
- Lots of work done upstream, we benefit

What about for Lustre?

- Not much. It's just another block device, really.
- It's fast! Fast is nice.
- Lustre already handles high bandwidth OSTs
- It turns out approach used for “Box full of spinning disks” works well for flash
- Servers are already network limited more than disk limited

A little more...

- Lustre is great at extracting all of the bandwidth from high speed flash arrays
- Minimal overhead: Ldiskfs + LVM gets > 95% of raw performance
- ZFS also good (or so I hear)
- Issues are much more around building hardware that can move the data

What about latency...?

- Flash has much better latency for small I/O (Large I/O is bandwidth limited)
- ~100x faster
- Good for small random I/O
- 'Chatty' workloads like (some) big data jobs
- Lustre is poor at exposing this: 4k read latency of 500 microseconds on Cray hardware, 80 microseconds is flash (network latency ~1-5 microseconds)

Latency → Small I/O

- Latency is only relevant for small I/O
- Small I/O is terrible on spinning disk
- But still pretty bad on flash – Flash can't hit top end bandwidth with small I/O
- Small I/O creates lots of network traffic
- Classic spinning disk solution:
  Don't do small I/O

Solution: The Page Cache

- Sequential small I/O doesn't have to be small to disk
- Readahead for reads
- Write aggregation for writes
- Lustre doesn't do small I/O to disk (or over network) unless forced (direct I/O, random reads)
- Works well for flash – Much better than direct I/O (except for random reads)

Small I/O Improvements

- Small I/O is still tough, but it's also important
- High per I/O overhead make it slow even to page cache
- Previous work:
  Fast reads from Intel (~10x improvement for 8 byte reads, helps at all sizes)
- Current/future work:
  Tiny writes
  Immediate short I/O
  Write containers
- See my LAD Developer Summit talk for details

But... Latency matters!

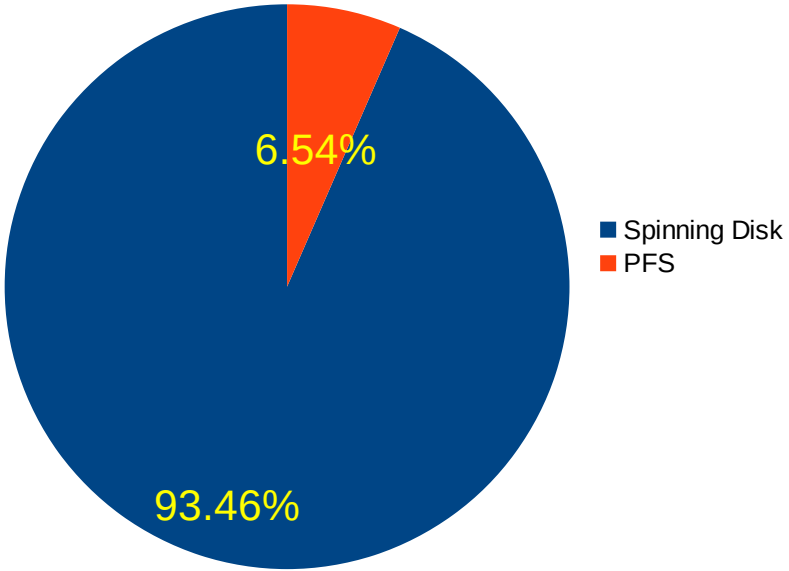
- Excitement over persistent memory tech is all around low latency
- Major investments in this area, DAOS-M from Intel, various related efforts
- And flash latency is 100x better than spinning disk. Shouldn't we try to unlock that?
- Yes: But we already do all right at that.

Latency Realms

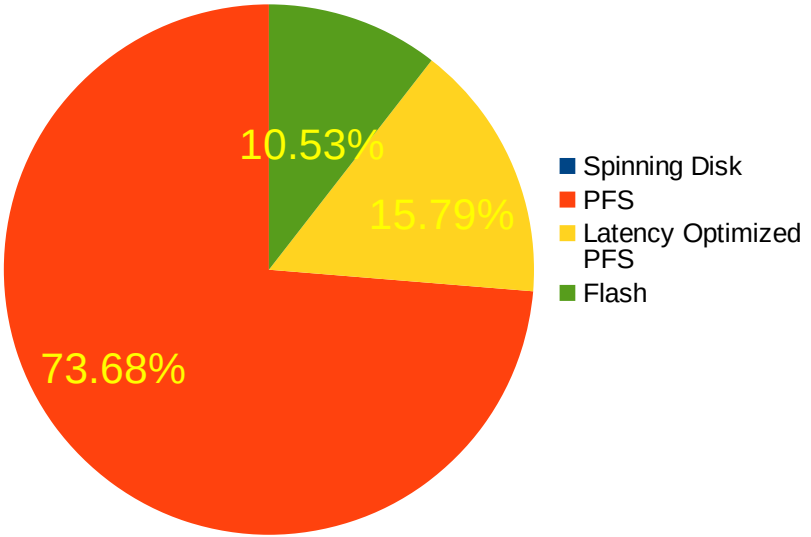
- Let's talk orders of magnitude.
- Spinning disk: ~10 ms (1*10^-2)
- PFS ~700 ms (7*10^-4)
- Flash ~100 µs (1*10^-4)
- Persistent memory ~1 µs (1*10^-6)
- MPI communication (Aries) ~1 µs (1*10^-6)
- 1000 CPU cycles (4 Ghz CPU) ~0.25 µs (0.25*10^-6)

Latency Pies

High Latency: Spinning Disk + PFS (10 milliseconds)

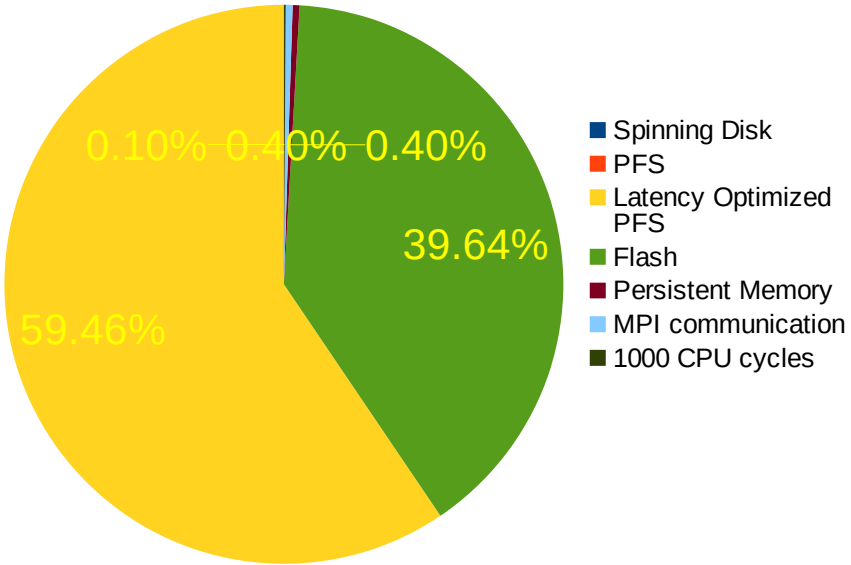


Medium Latency: PFS + Flash (~1 millisecond)

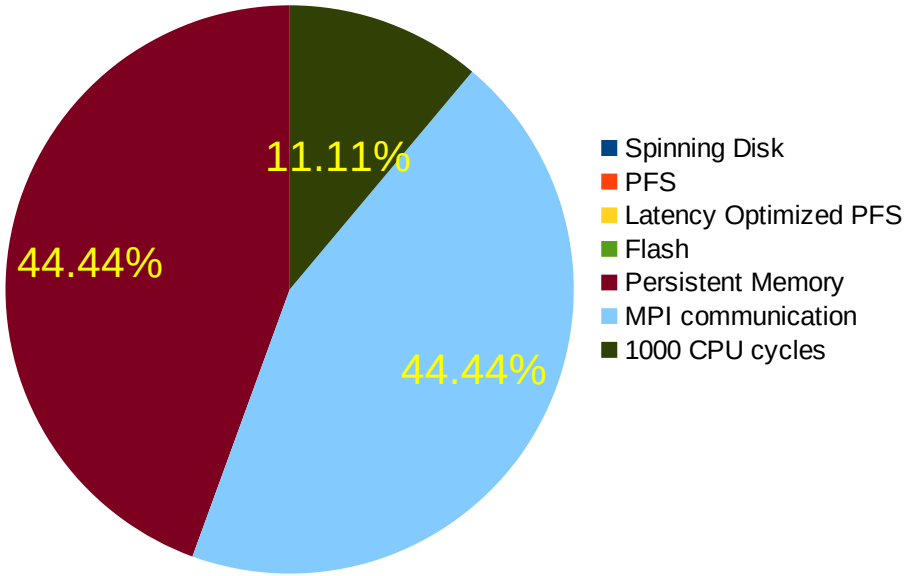


Latency Pies

Low Latency: Flash + Optimized PFS (~250 µs)



True Low Latency: Persistent Memory, MPI, CPU (~2.5 µs)



Spinning Disks and Application Design

- Application design reflects latency realms
- MPI & compute broadly comparable (~1 µs)
- I/O incredibly slow (Spinning disk ~10,000 µs)
- Interleave compute & communication, but wait and do I/O in large chunks
- Compute, MPI, compute, MPI… (repeat)
  Do I/O
  Compute, MPI, compute, MPI… etc

Medium latency I/O & application design

- Flash + Lustre best case latency is ~500 µs
- Compare to ~10,000 µs for spinning disk + Lustre
- Better not to do small random I/O, but some applications have no choice (big data)
- Flash is very helpful for this, giving a ~10x improvement with Lustre
- Even though Lustre is now the main source of latency, it's still a huge improvement

Should we redesign for flash?

- Flash latency is ~100x better (100 µs)
- But still 100x slower than MPI (1 µs)
- Still can't interleave I/O with compute + communication
- So… Probably not.
- Persistent memory is different: 1 µs
- Can now interleave:
  Compute, MPI, store, compute, MPI, store...

The Future

- Persistent memory really is different, will enable new application designs
- POSIX compliance isn’t really possible in available time (~1 µs)
- Lustre can't be the enabling tech there, hence projects like DAOS-M
- But Lustre can unlock the potential of flash

The Future (2)

- “The future is seldom the same as the past” - Seymour Cray
- Seldom, but not always… and Lustre is still changing.
- Lustre is still the future of parallel file systems, still the right answer as we move to flash as primary
- DAOS-M and similar projects are something new (Post-POSIX)