Data-Intensive Systems in the Microsecond Era



Pınar Tözün[1], Ivan Luiz Picoli[1,2], Heiner Litz[3], Philippe Bonnet[1]

[1]IT University of Copenhagen

[2]Samsung Semiconductor Denmark Research (SSDR)

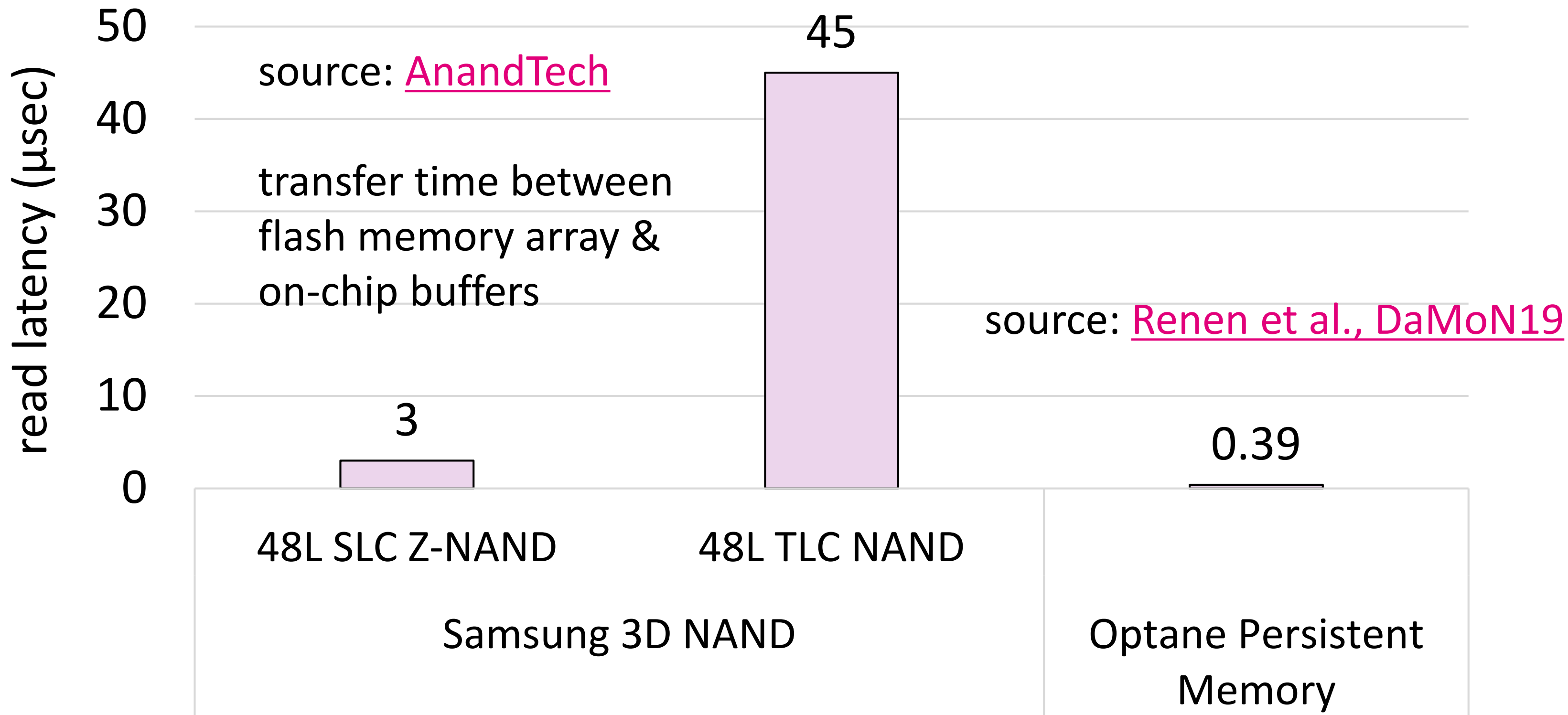
[3]UC Santa Cruz

HPTS

06/11/2019

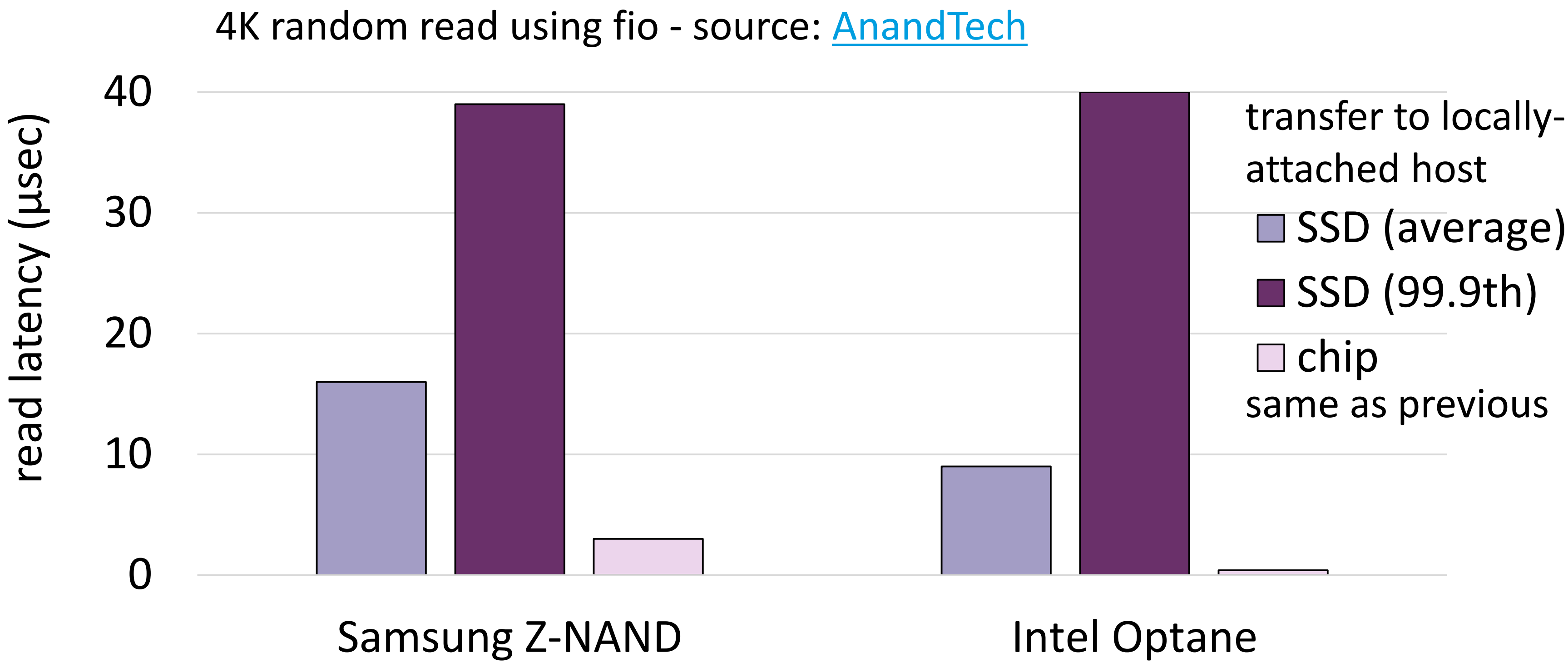
state of the world we live in today

storage chips in the µsec era



Z-NAND storage chips are 8x slower than Optane memory, and 15x faster than existing NAND chips.

SSDs in the μsec era



SSDs equipped with Z-NAND & Optane deliver at best 5x & 20x the read latency of the underlying storage chip, respectively.

interconnects in the µsec era

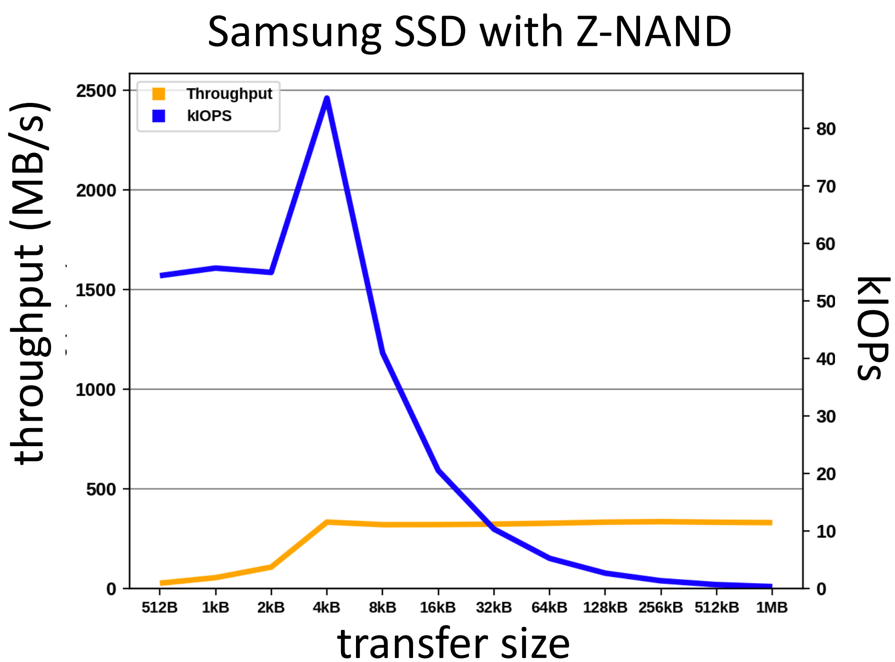
Not going to show numbers here.

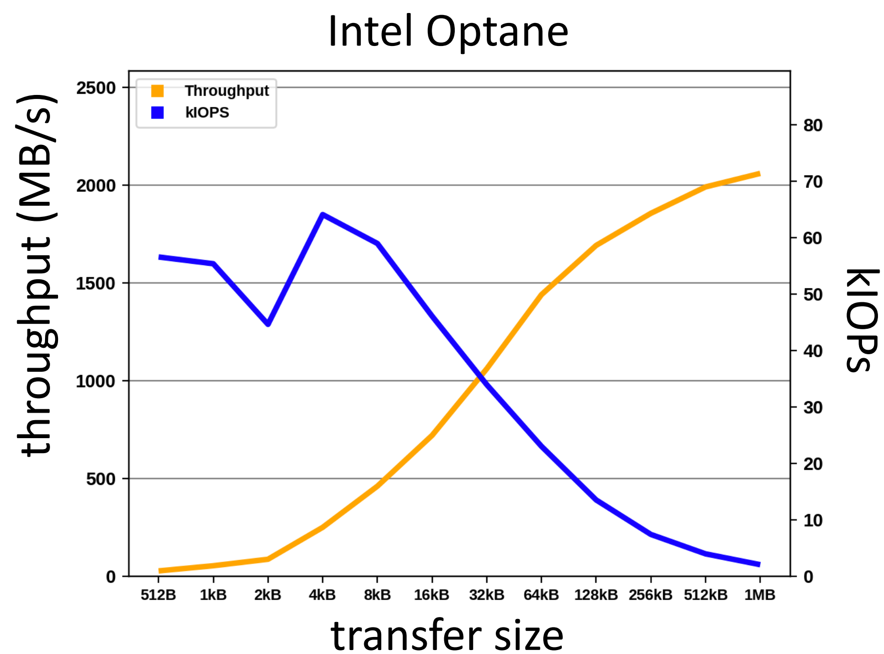
Latency is of the same order as storage chip latency for fast interconnects.

Fast network-attached storage (RDMA-based) just adds to the latency of direct-attached storage (PCIe).

ftls in the µsec era …

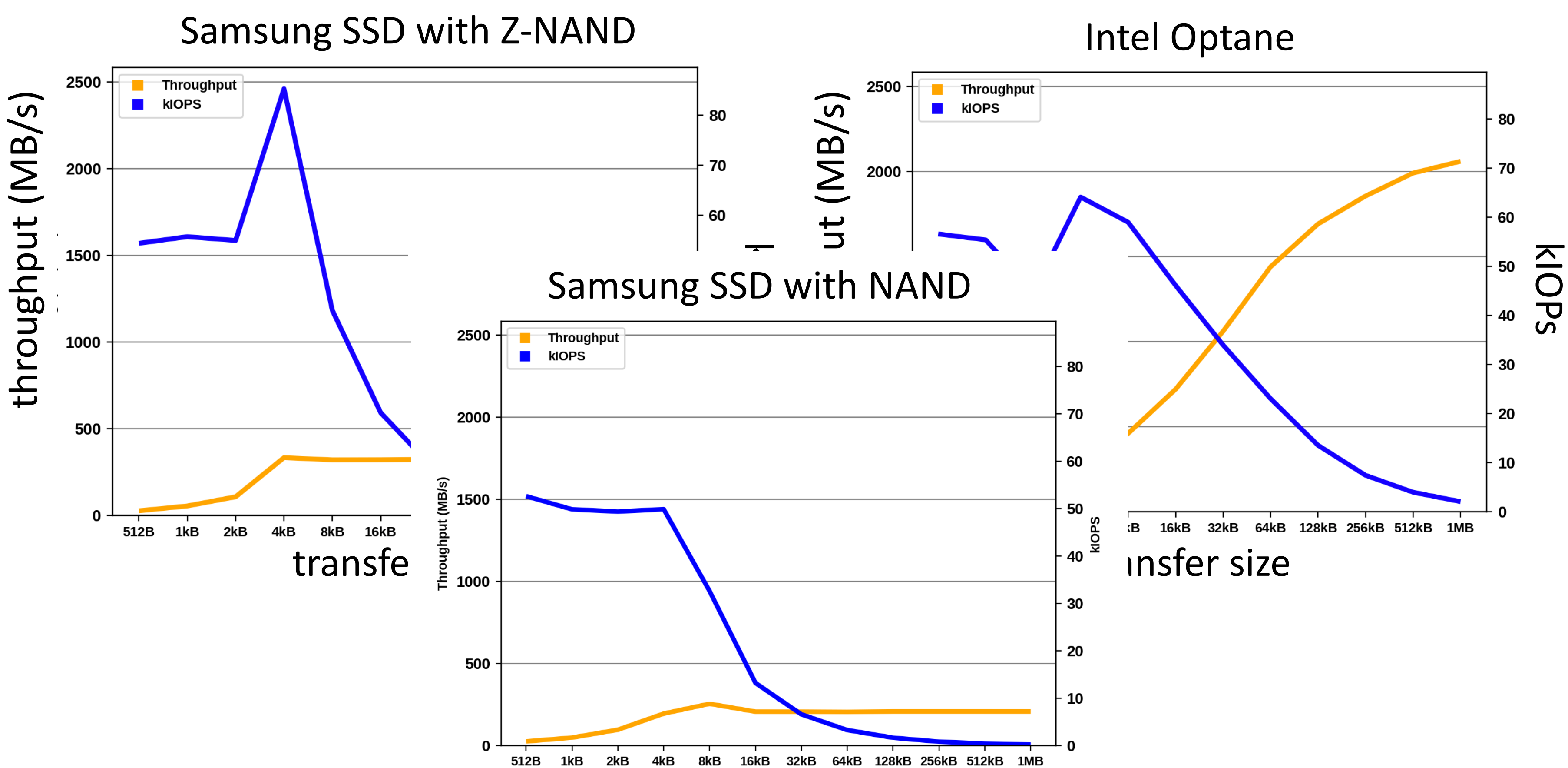
random writes- source: AnandTech





ftls in the µsec era …

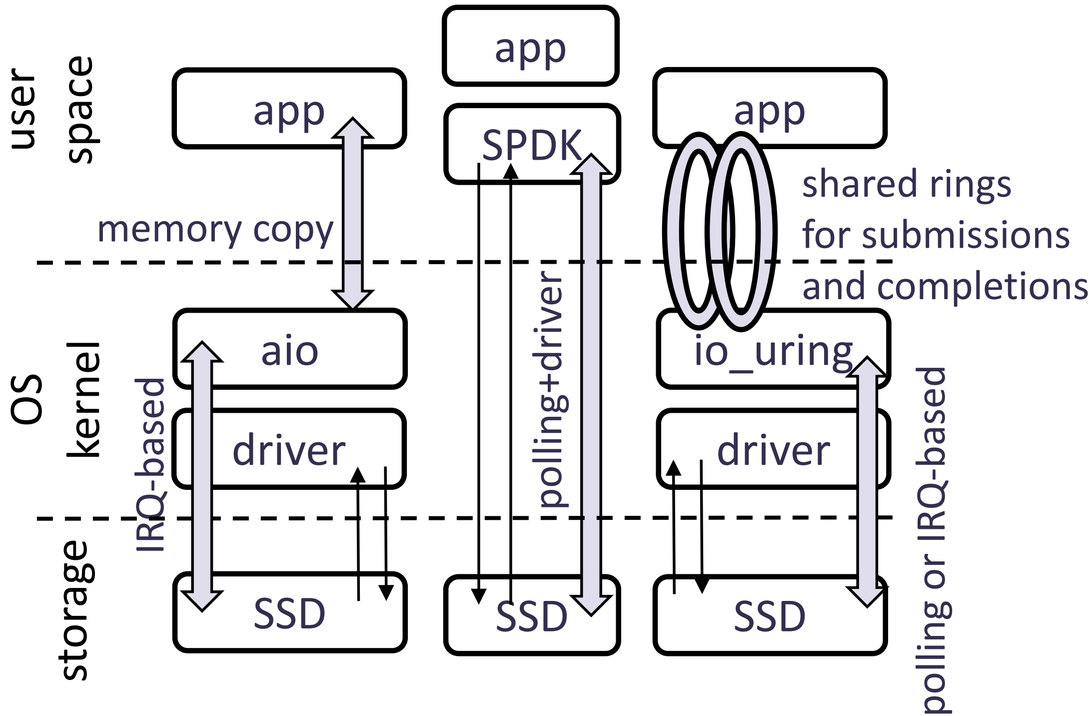
random writes- source: [AnandTech]()

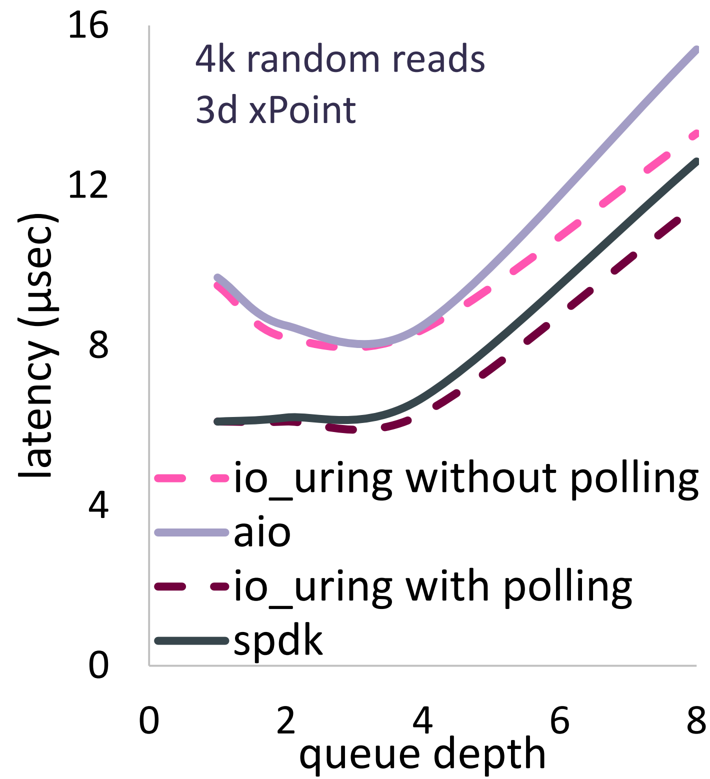


… have even more drastic impact on throughput!

linux IOs in the µsec era

sources: Faster IO through io_uring & Efficient I/O with io_uring & J.Axboe





separation of control & data plane in linux now, POSIX out
zero copy & minimized synchronization overhead

the benefits of fast storage wasted by
- data movement overheads
(from device to host & across network)
- black-box generic flash-translation layers
- multitude of software layers

how do we prevent these?

back when I was 10 …

Put Everything in Future (Disk) Controllers (it's not "if", it's "when?")

Jim Gray

http://www.research.Microsoft.com/~Gray

Acknowledgements:
Dave Patterson explained this to me a year ago
Kim Keeton
Erik Riedel
Catharine Van Ingen
Helped me sharpen these arguments

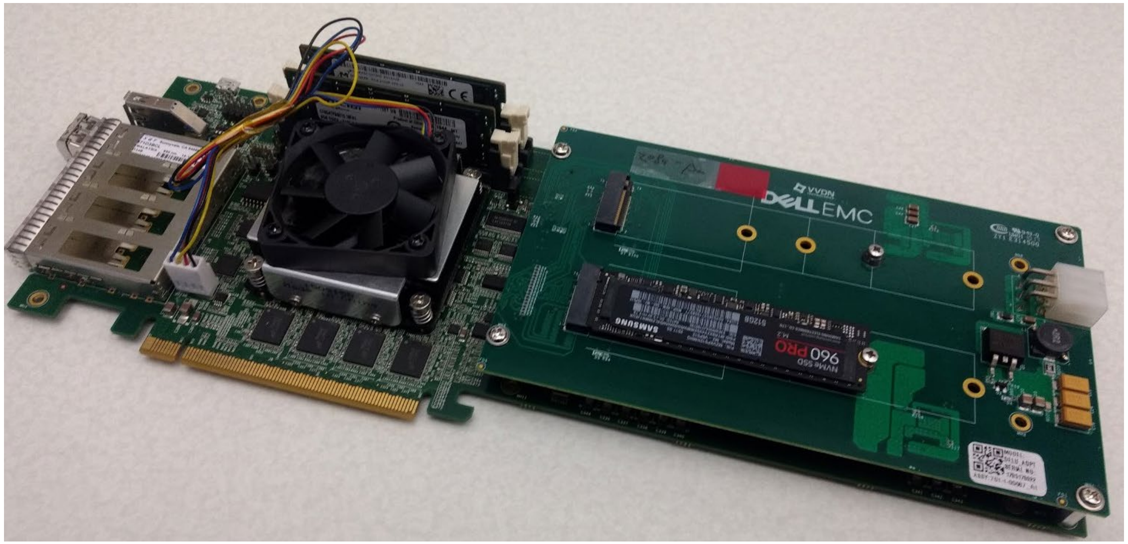
1

Basic Argument for x-Disks

- Future disk controller is a super-computer.
  - 1 bips processor
  - 128 MB dram
  - 100 GB disk plus one arm
- Connects to SAN via high-level protocols
  - RPC, HTTP, DCOM, Kerberos, Directory Services,….
  - Commands are RPCs
  - management, security,….
  - Services file/web/db/… requests
  - Managed by general-purpose OS with good dev environment
- Move apps to disk to save data movement
  - need programming environment in controller

Jim Gray, NASD Talk, 6/8/98
http://jimgray.azurewebsites.net/jimgraytalks.htm

today …



8-core ARMv8 processor

32GB DRAM

2TB+ of NVM via M.2 slots

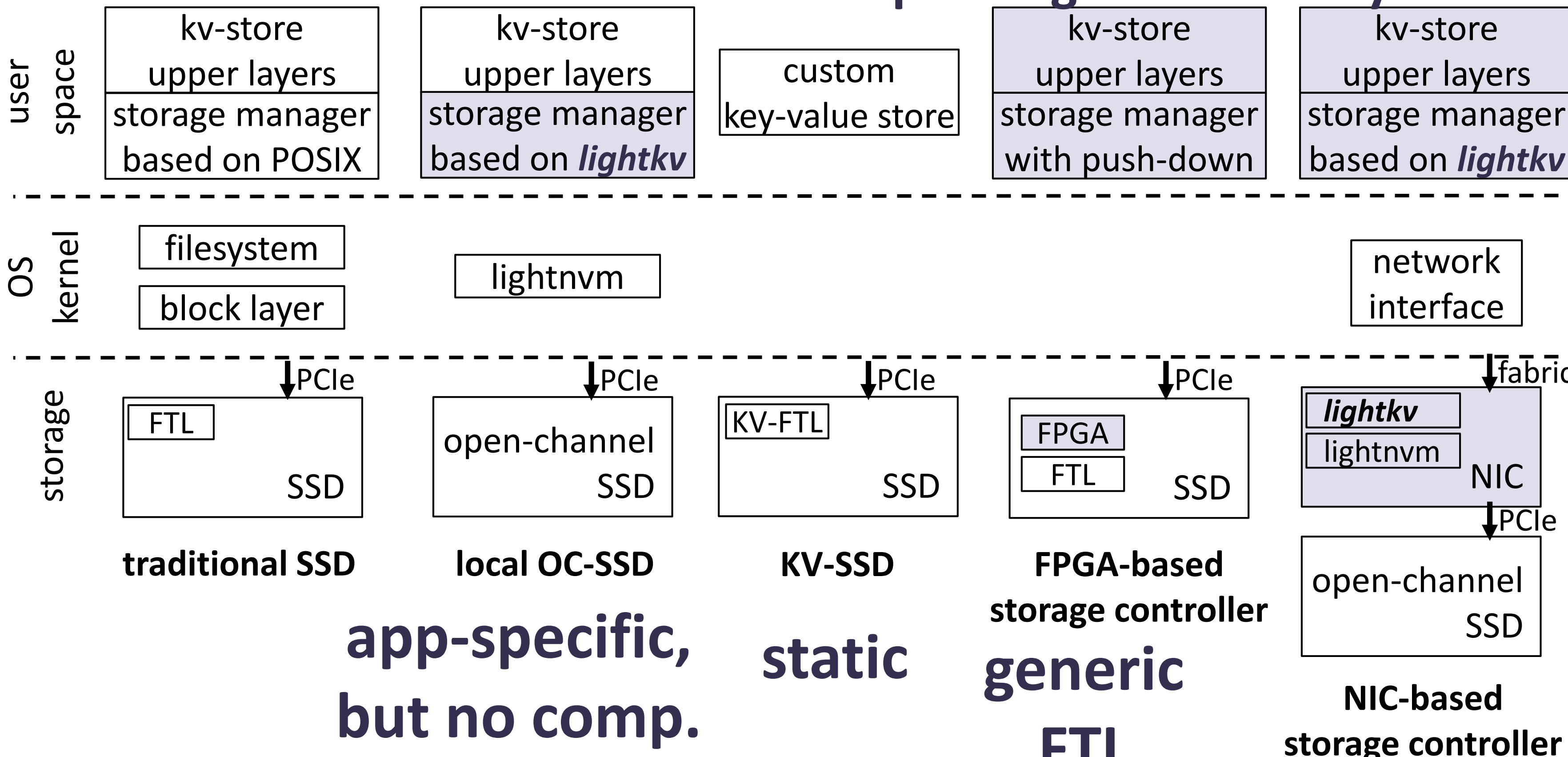
4x 10Gb Ethernet

Dragon Fire Card (DFC)
https://github.com/DFC-OpenSource/

- Future disk controller is a super-computer.
  - »1 bips processor
  - »128 MB dram
  - »100 GB disk plus one arm

SSD landscape

kv-store needs to change when you start pushing functionality down!



app-specific, but no comp. storage

static

generic FTL

app-specific FTL

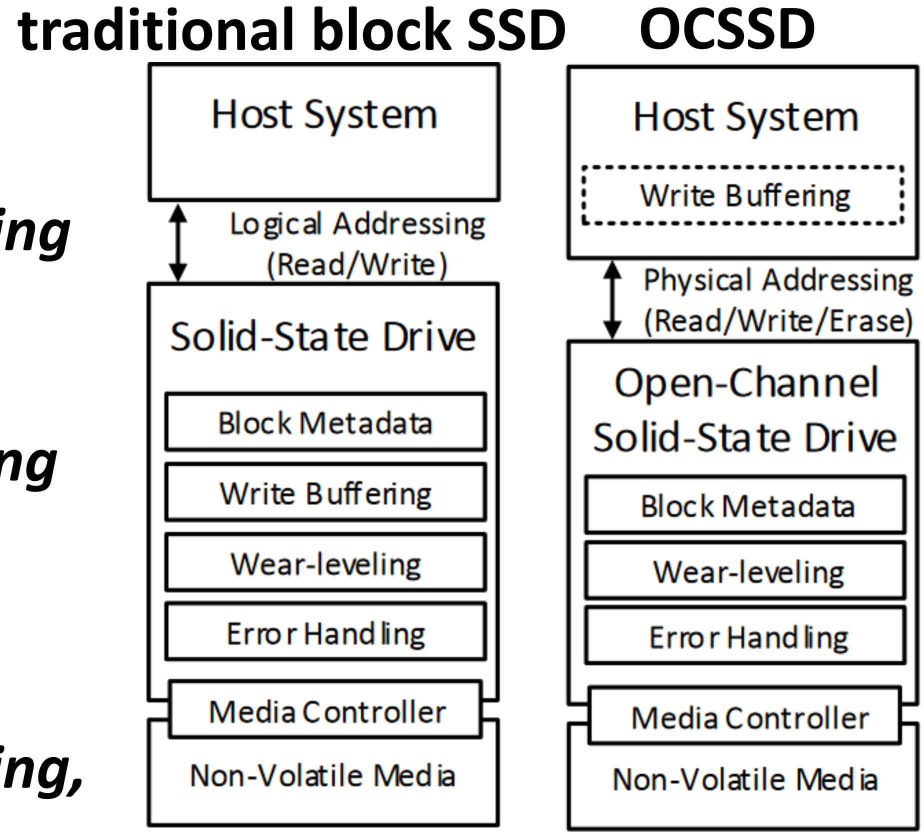
open-channel SSDs

physical address space exposed

- host can make decisions about ***data placement*** & ***I/O scheduling***

SSD management split between

- back-end (embedded on SSD) ***block metadata*** & ***wear levelling*** (for warrantee)
- front-end (host-based) FTL ***mapping of logical to physical address spaces, overprovisioning,*** & ***garbage collection***



separates (*application-customizable*) front-end from (*media-specific*) back-end

potential impact

WHAT?

I/O isolation

- host management of device internal resources for contention avoidance
- control latency predictability – beyond NVMe IO determinism

resource utilization

- controlled data placement to reduce write amplification – beyond NVMe streams

streamline data path

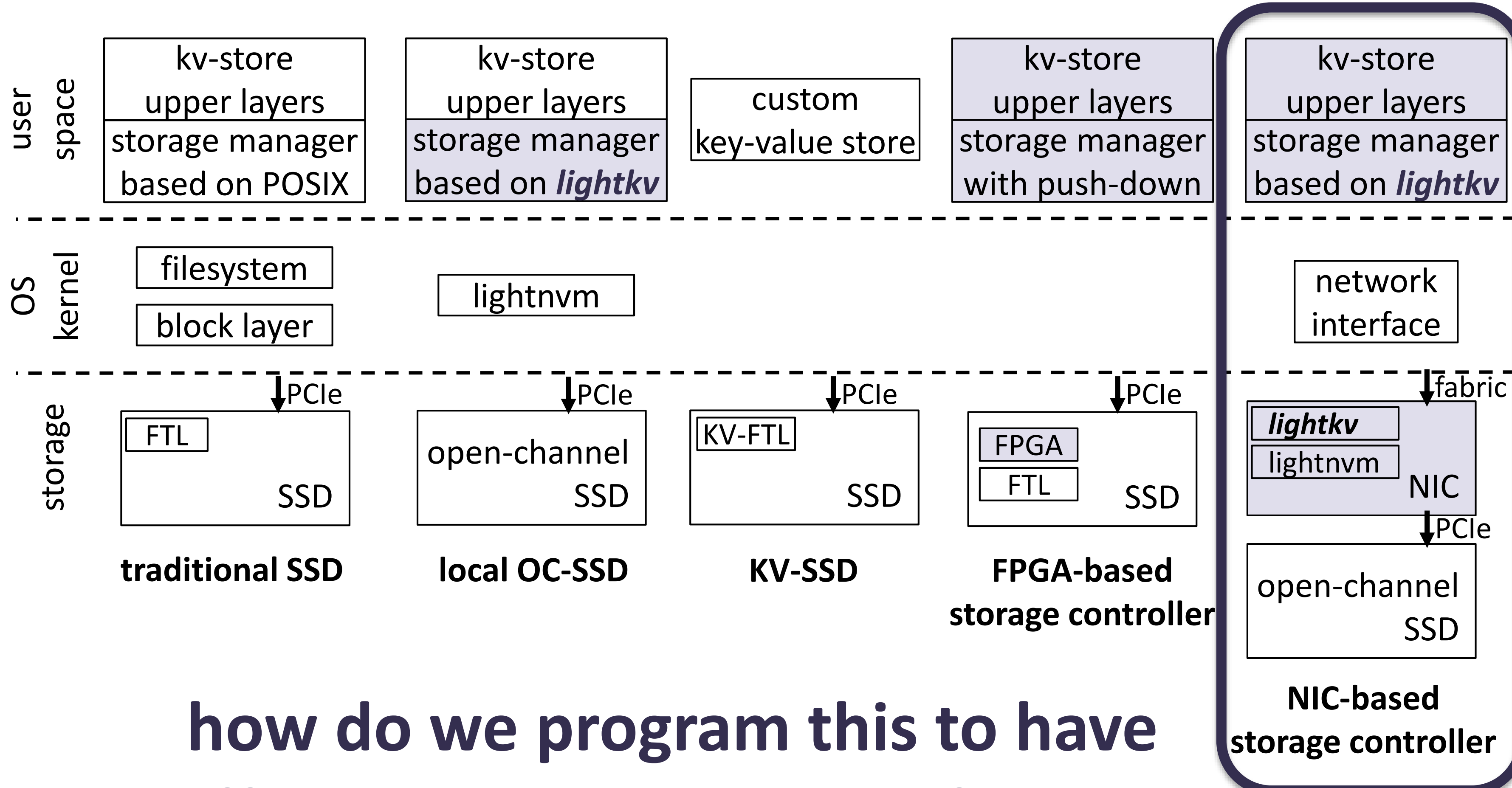
- application-specific FTL

HOW?

computational storage

- offload CPU
- shield host application from complexity of managing the physical space (e.g., flash characteristics)
- co-design of application-specific FTL and OCSSD

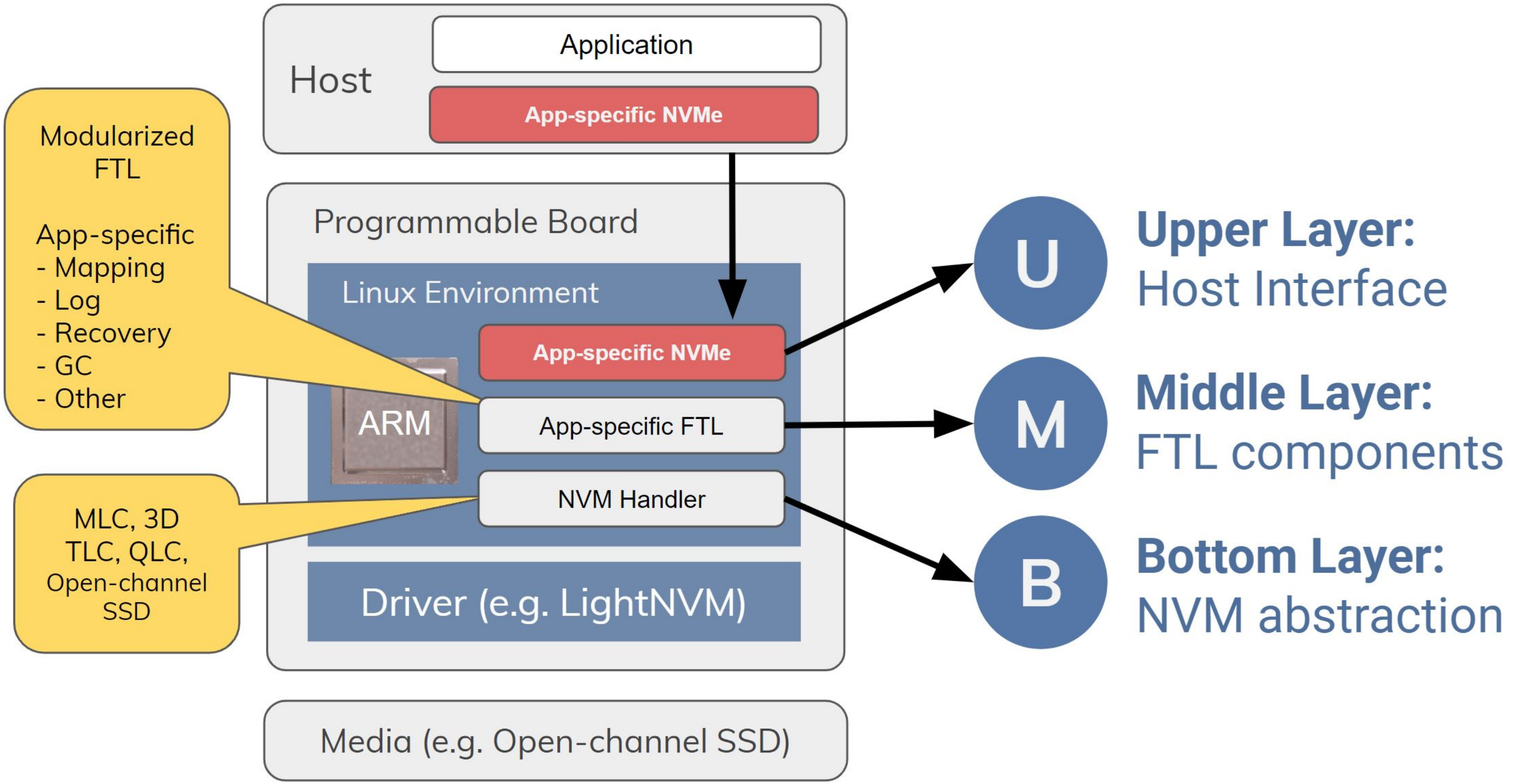
SSD landscape



how do we program this to have different application-specific FTLs?

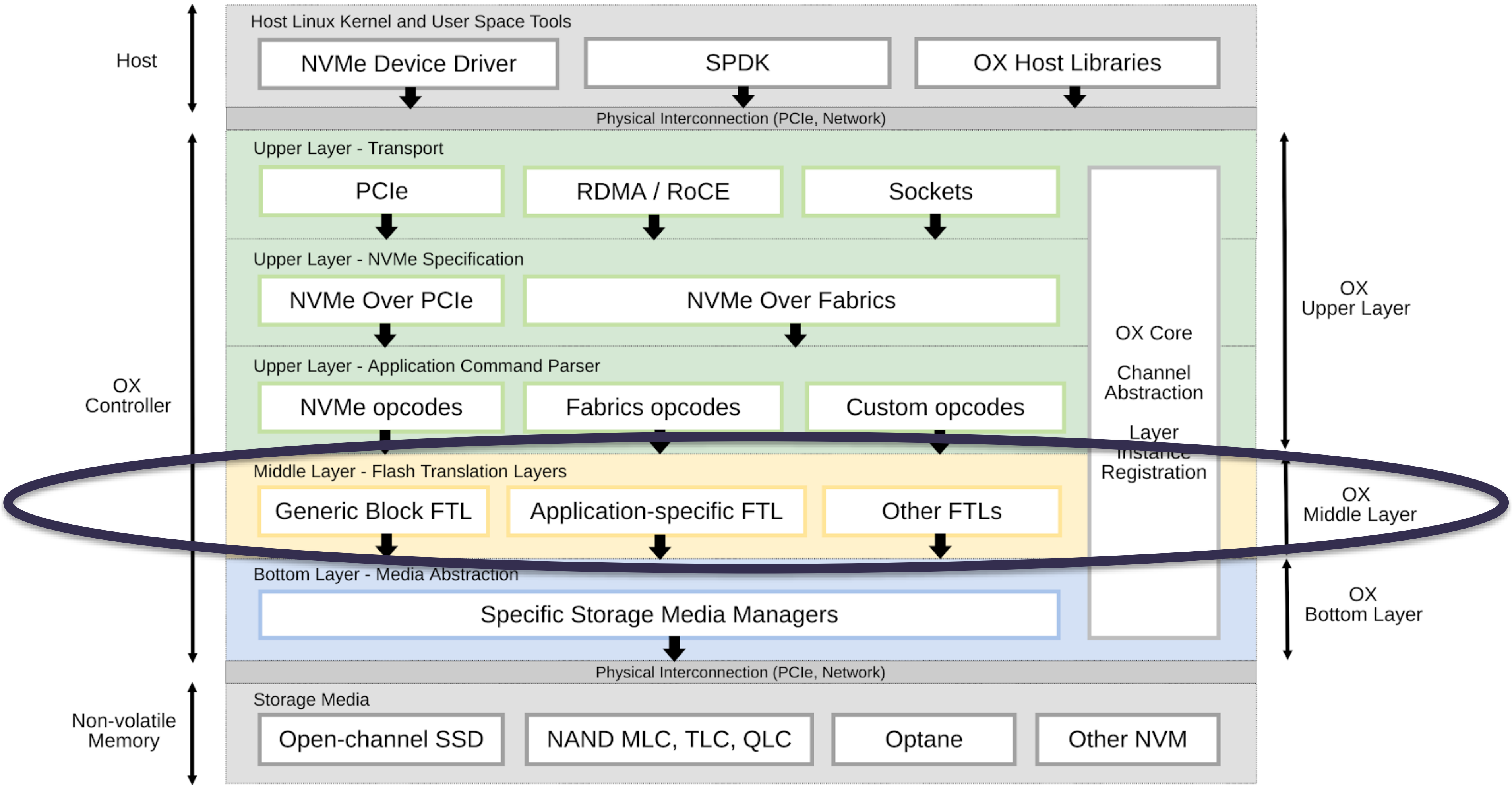
programming interface

core of Ivan’s thesis work

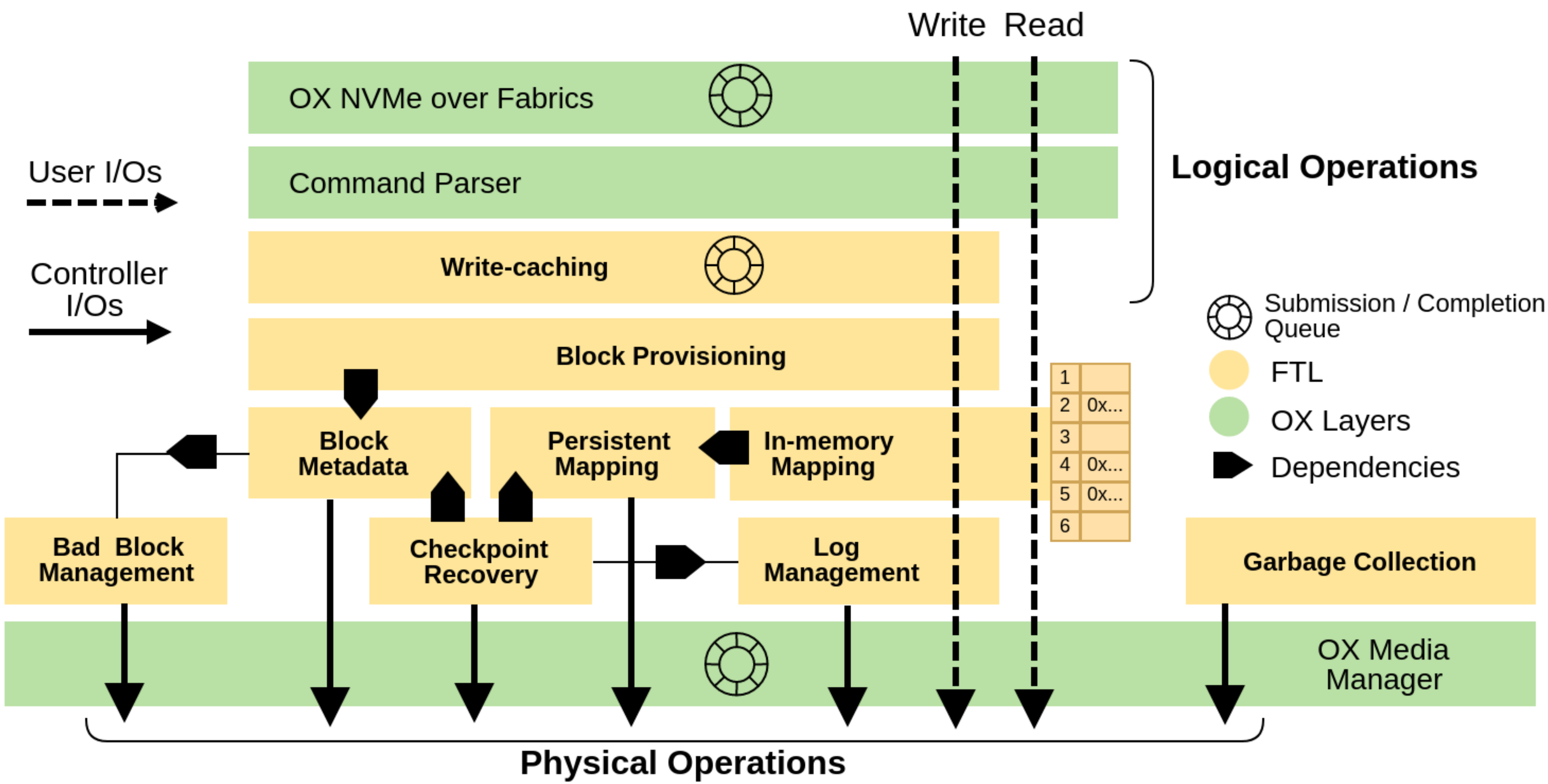


OX to program storage controllers

https://github.com/DFC-OpenSource/ox-ctrl



deconstructing FTL with OX



conclusion

- need to be careful about data movement
  - computational storage would help
- application-specific FTLs would naturally allow computational storage on SSDs
  - use cases:
    - LSM (ongoing work in our lab using RocksDB)
    - BwTree (what Dave Lomet talked about during gongshow)
- open issues
  - ZNS (zoned namespaces)
  - what to push down?

thank you!