lightbits

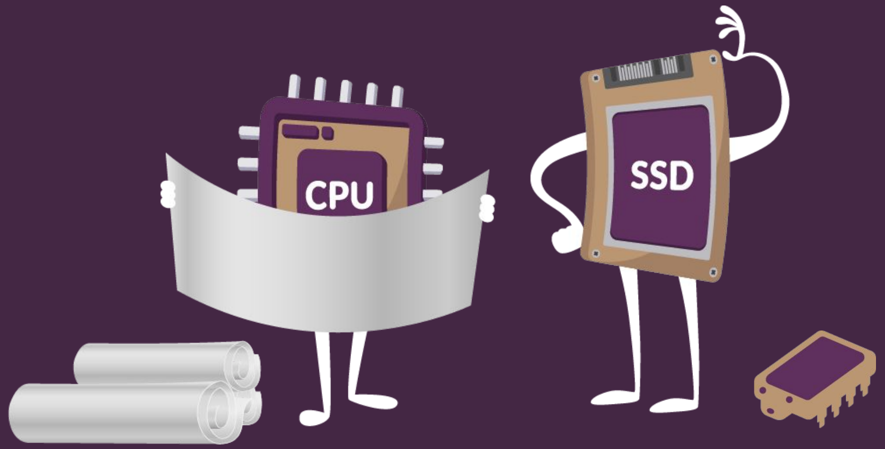
Achieving High Performance and Low Latency with NVMe/TCP

Sagi Grimberg
Co-Founder & Chief SW Architect

Agenda

- Motivation for NVMe/TCP
- Short architectural overview
- NVMe/TCP in Linux
- Some performance measurements
- Talk about common storage services with NVMe-oF

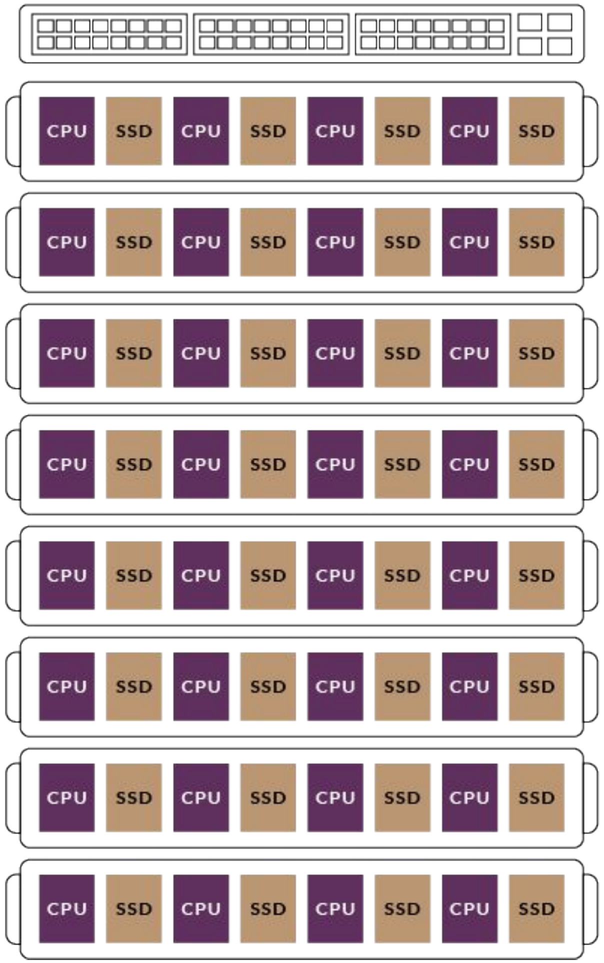
Motivation

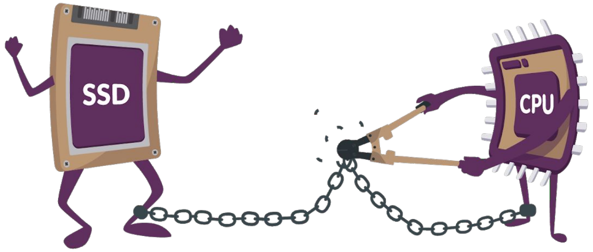


From direct-attached to a disaggregated cloud

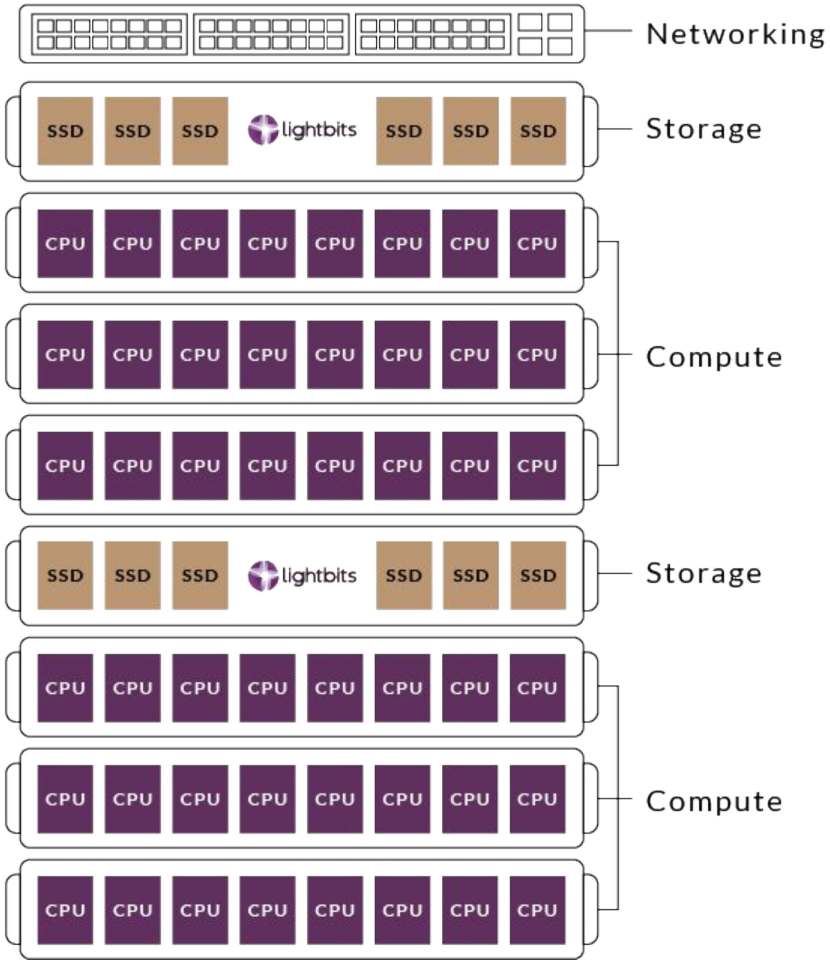
Flash Memory Summit

Direct-Attached Architecture





Lightbits Cloud Architecture

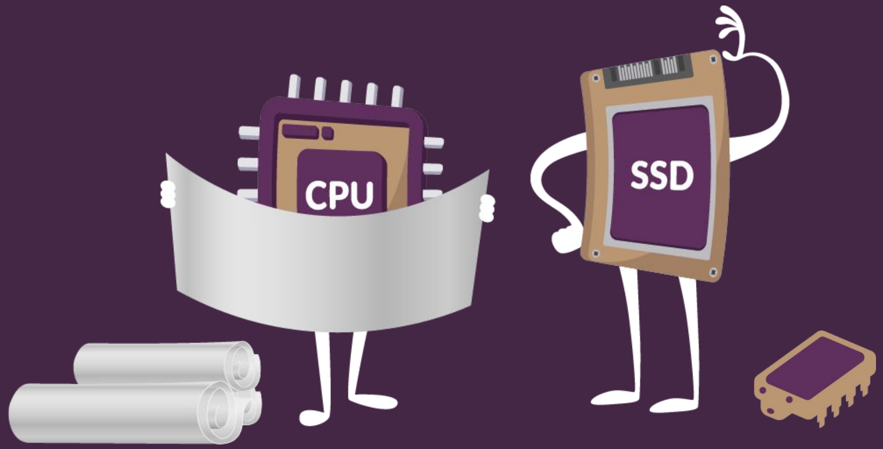


- Maximize utilization
- Reduce TCO
- Easy to maintain, operate and scale
- Better user experience
- Support more users

Why NVMe/TCP ?

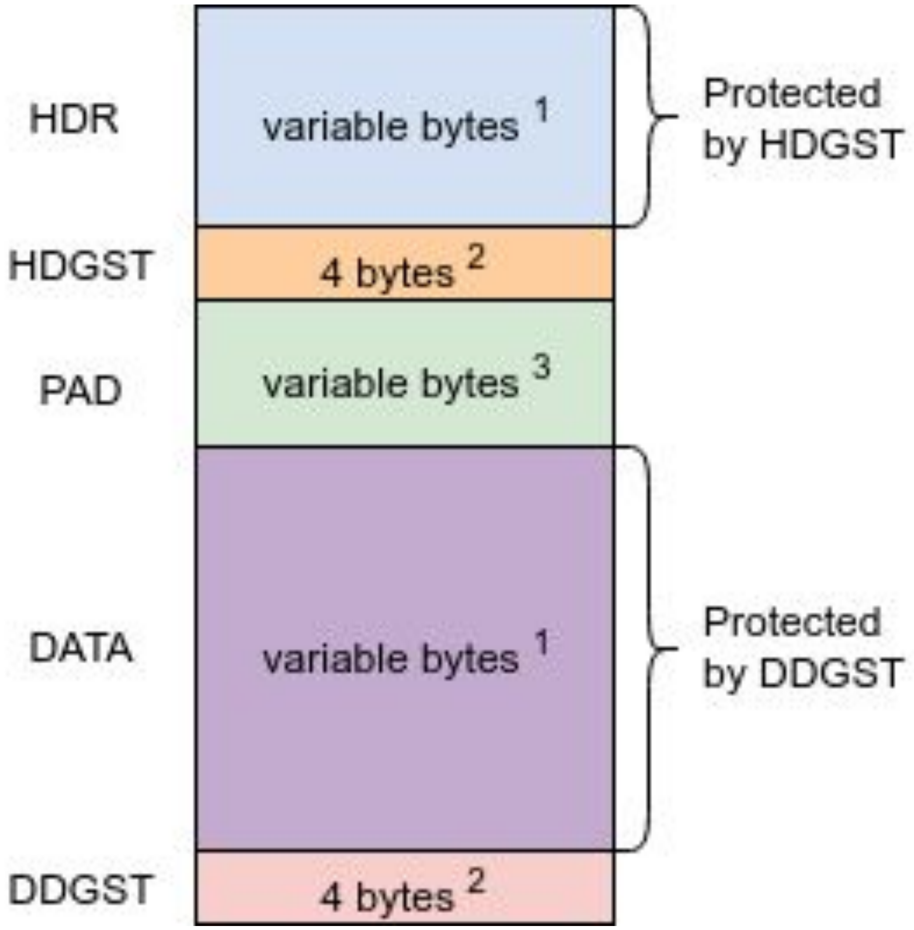
- Ubiquitous - No networking infrastructure requirements/constraints
- TCP is probably the most well-known and well-understood transport
- TCP is actively developed and maintained by the biggest players
- Delivers high performance and low latency
- Well suited for large scale deployments and longer distances

NVMe/TCP Overview



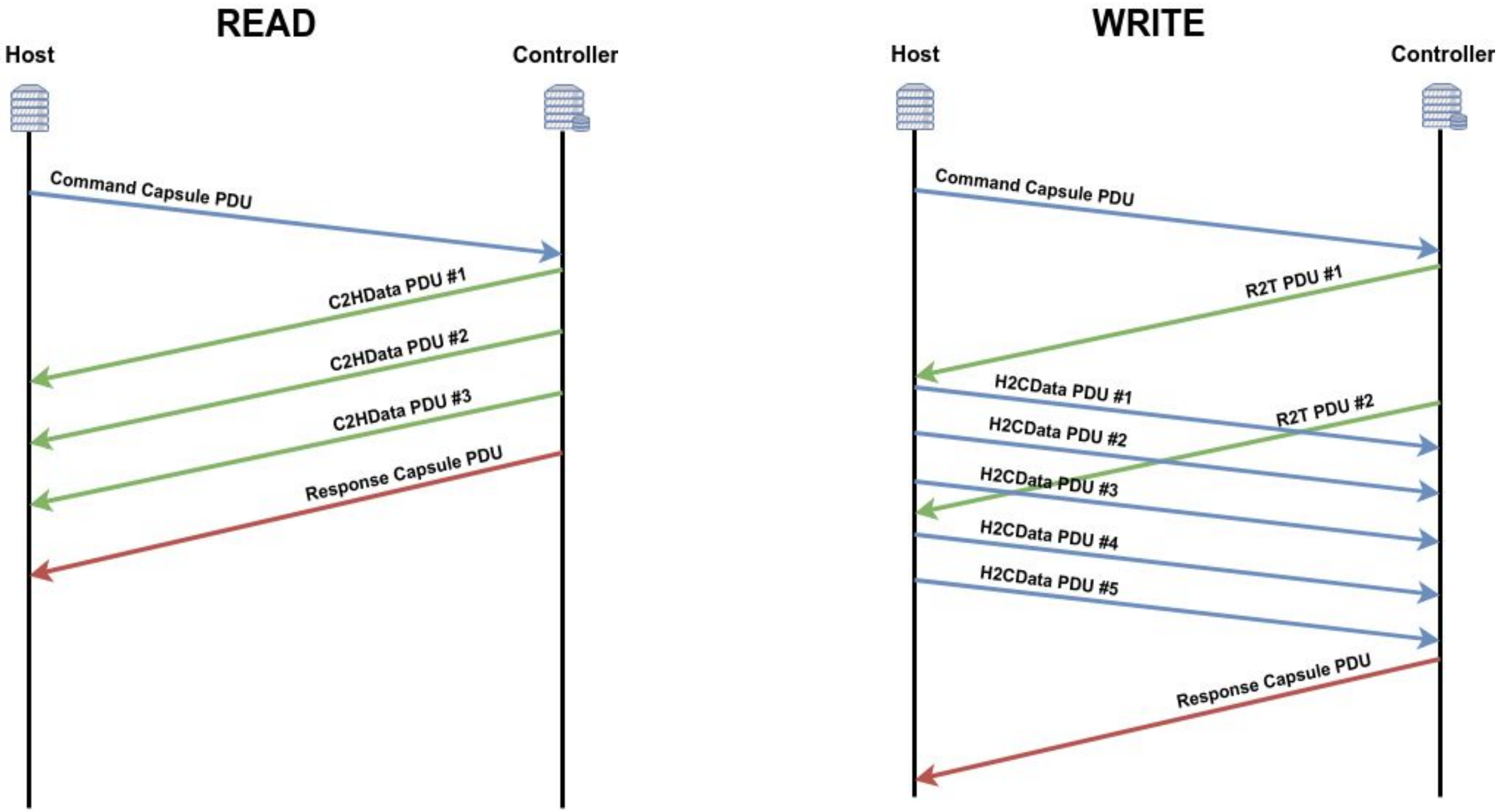
NVMe/TCP in a nutshell

- NVMe-oF Capsules and Data are encapsulated in NVMe/TCP PDUs
- PDUs have variable length
- PDUs contain optional Header and Data Digest protection
- PDUs contain optional PAD used for alignment enhancements

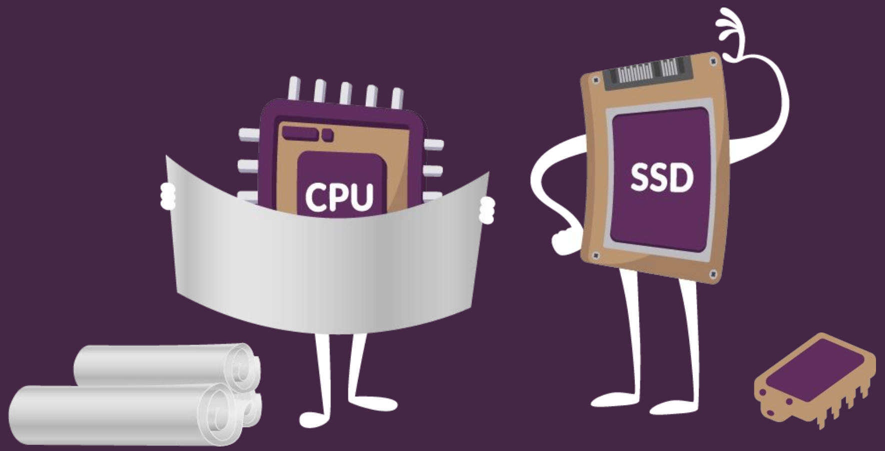


NVMe/TCP in a nutshell

- Host to Controller data direction can come either in-capsule or out of capsule

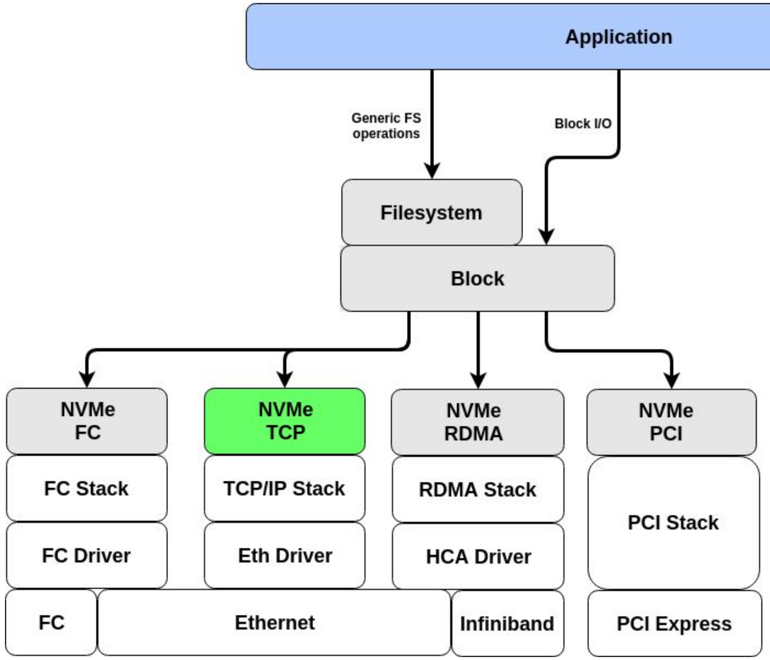


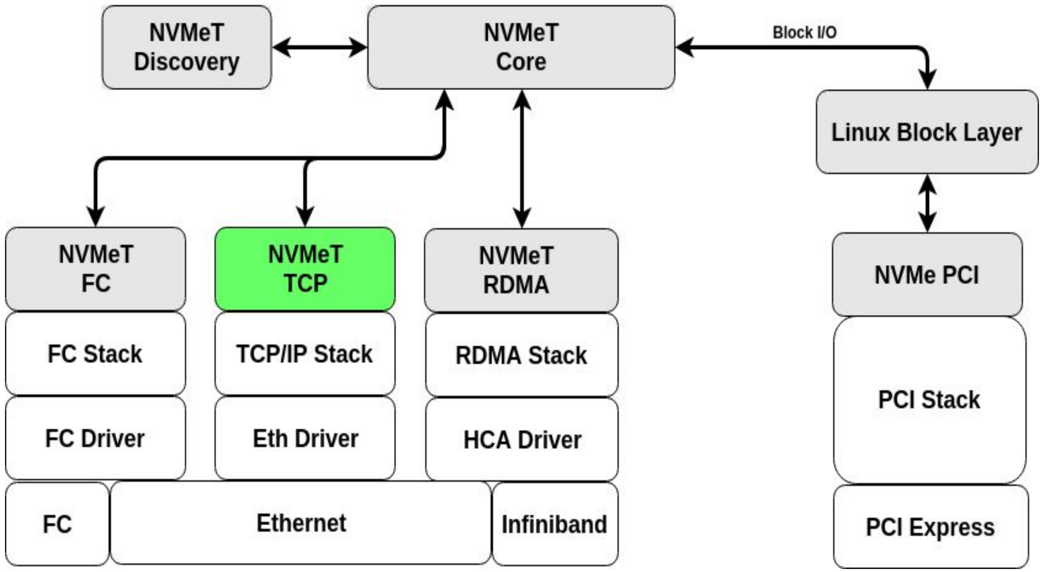
NVMe/TCP in Linux



NVMe/TCP drivers

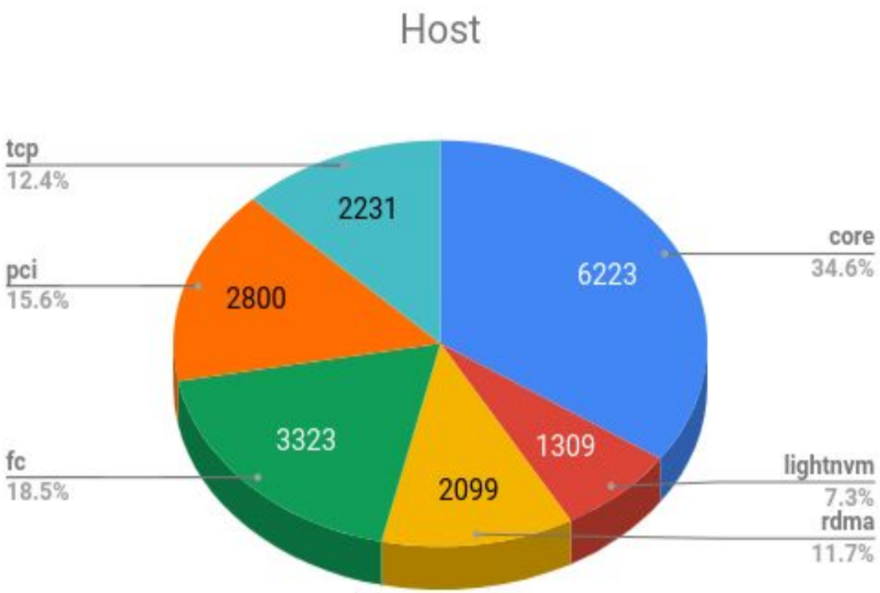
- Plugged into the stack as another fabrics transport in the NVMe subsystem
- Focused on simplicity and efficiency
- Aggressive code reuse and commonization (where makes sense)
- Not “reinventing the wheel” using common interfaces

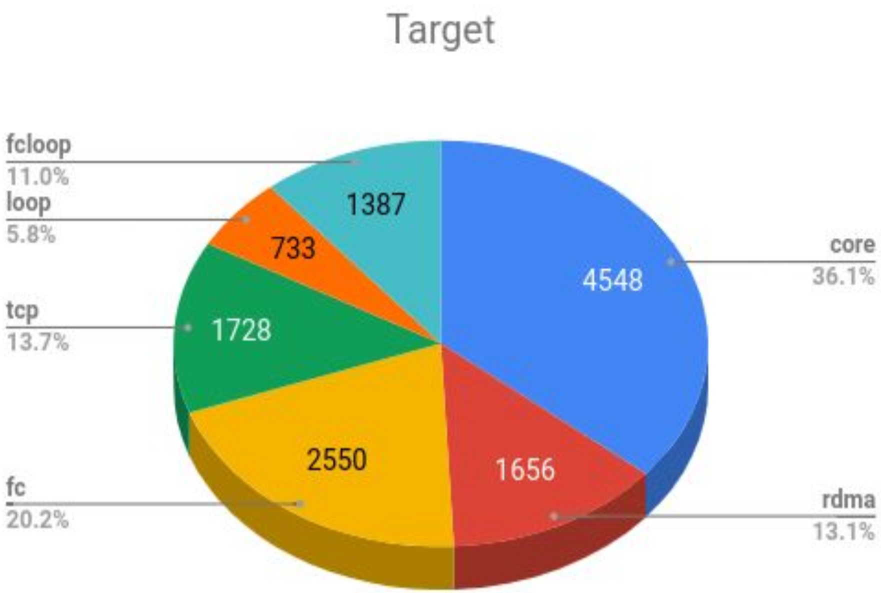




LOC count

- Linux NVMe subsystem is in pretty good shape where most of the code is common
  - We still have plenty of room for improvement...





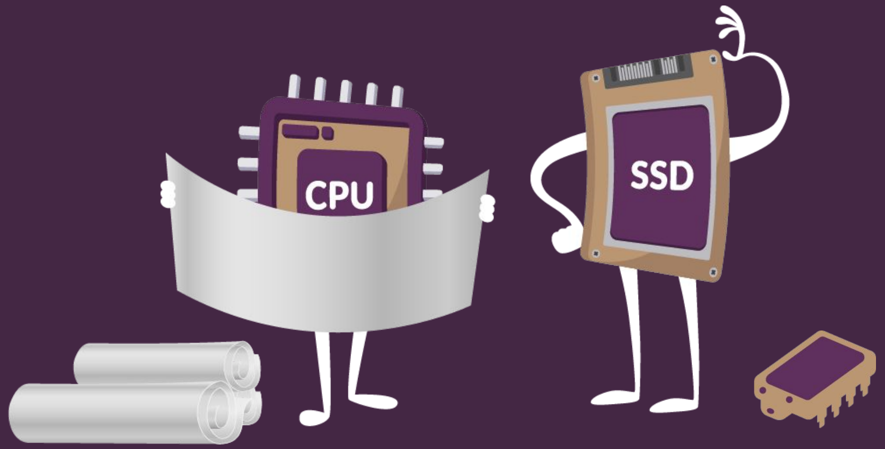
Drivers Design Guidelines

- Single reactor thread per-cpu
  - Each CPU core handles predefined number of NVMe queues
- ***NEVER*** block on I/O
- Aggressively avoid any data copy
- RX is handled in Soft-IRQ in order to complete as fast as possible
  - Called directly from NAPI
- Minimal set of atomic operations in the submission/completion paths
- Fairness and budgeting mechanisms multiplexing between NVMe queues

Features

- Zero-Copy Transmission
- Header and Data Digest
- CPU/NUMA affinity assignment for I/O threads (target side)
- TLS Support - Ongoing
- Polling mode I/O - Ongoing
- Automatic aRFS support - Future
- Out-Of-Order Data Transfers - Future

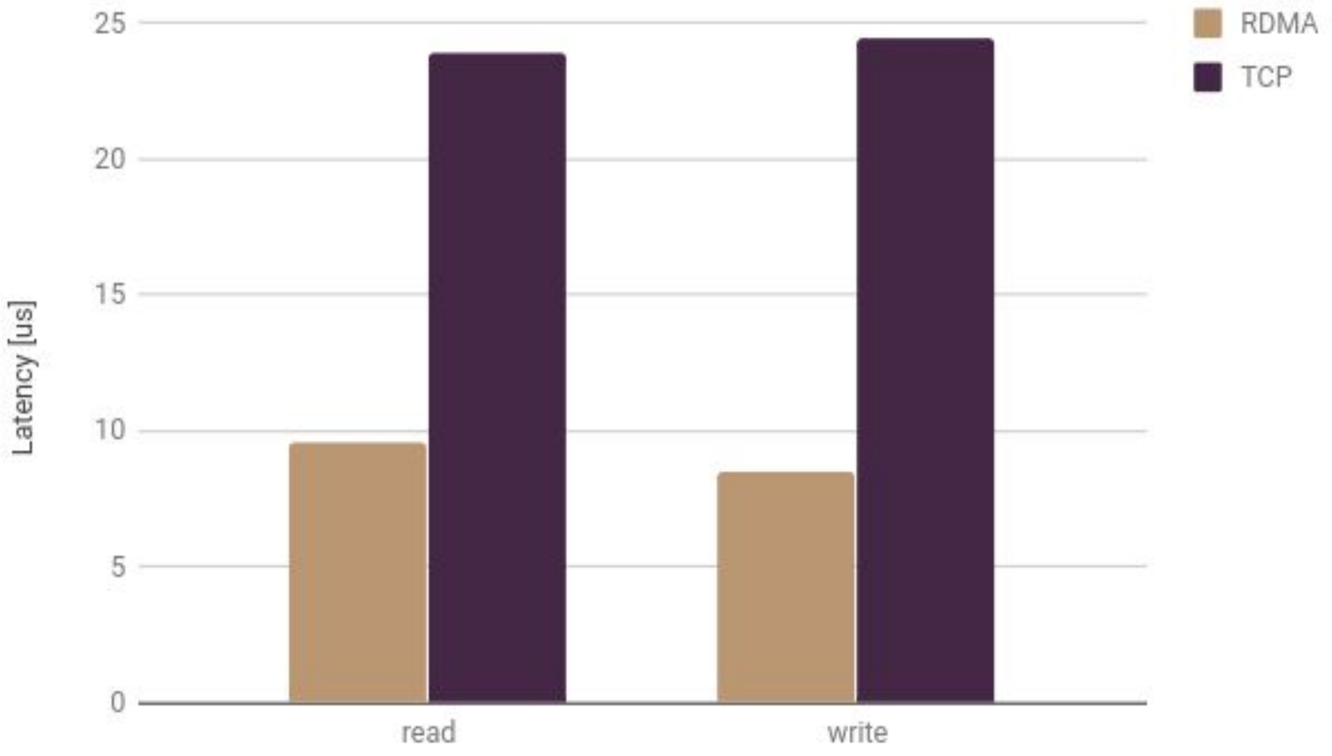
Some Performance Measurements



Canonical Latency Difference vs. DAS

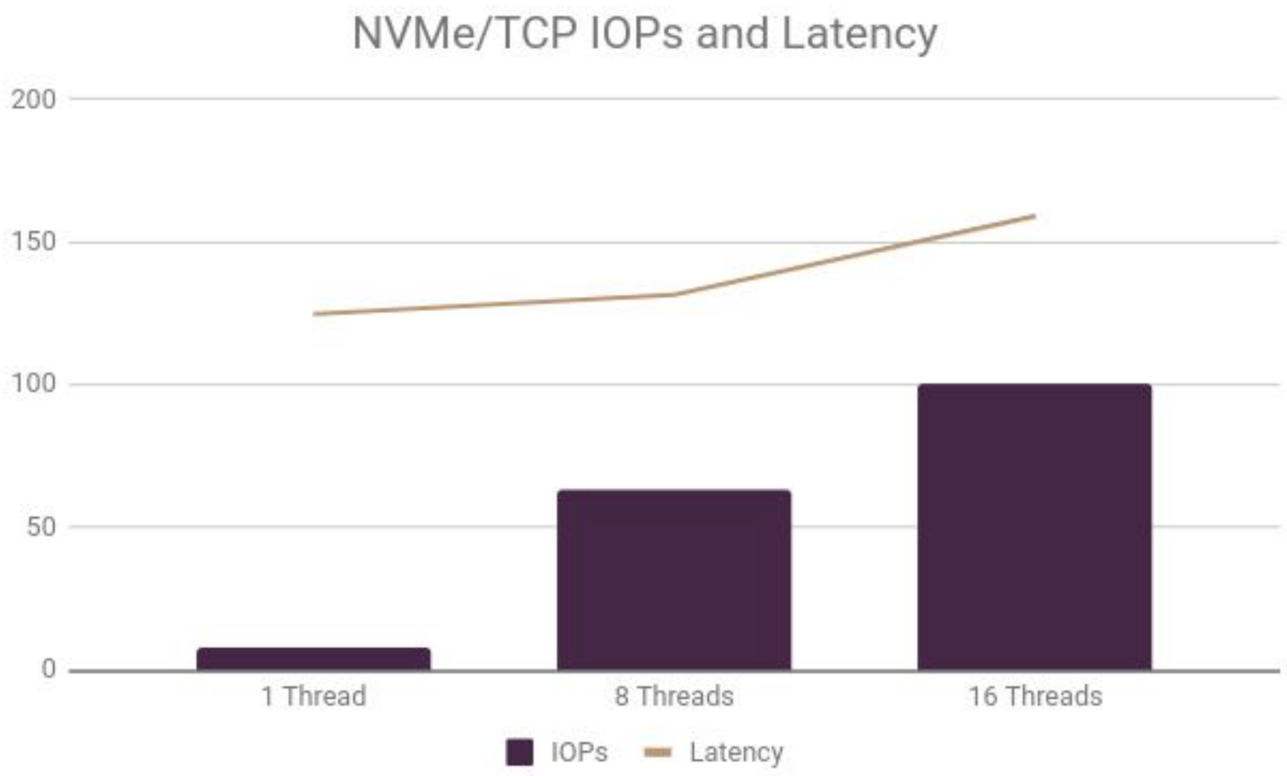
- Random Read
- 4KiB Block Size
- QD=1
- Null Backend device

While Latency is Slightly higher than RDMA, it is still very good

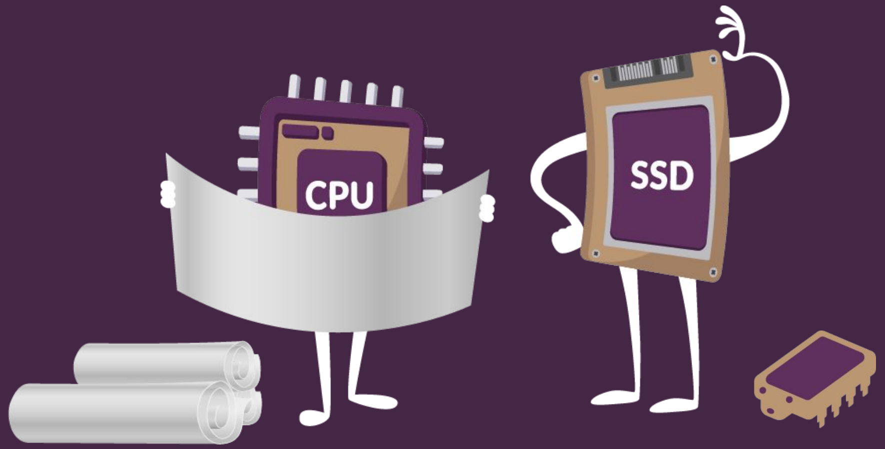


Thread Scaling

- Emulate multithreaded applications that issue blocking I/O (QD=1)
- NVMe/TCP performance scales with thread count and latency is not impacted

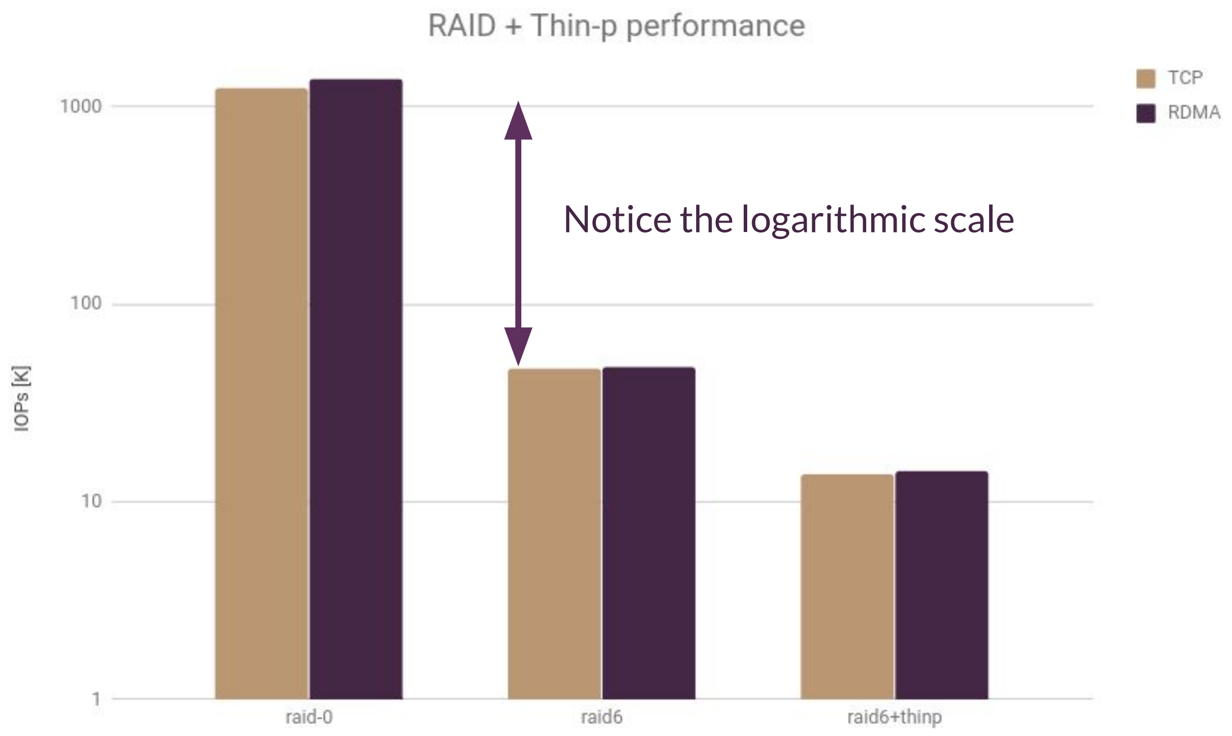


But what about common services?



Performance with RAID and Thin Provisioning

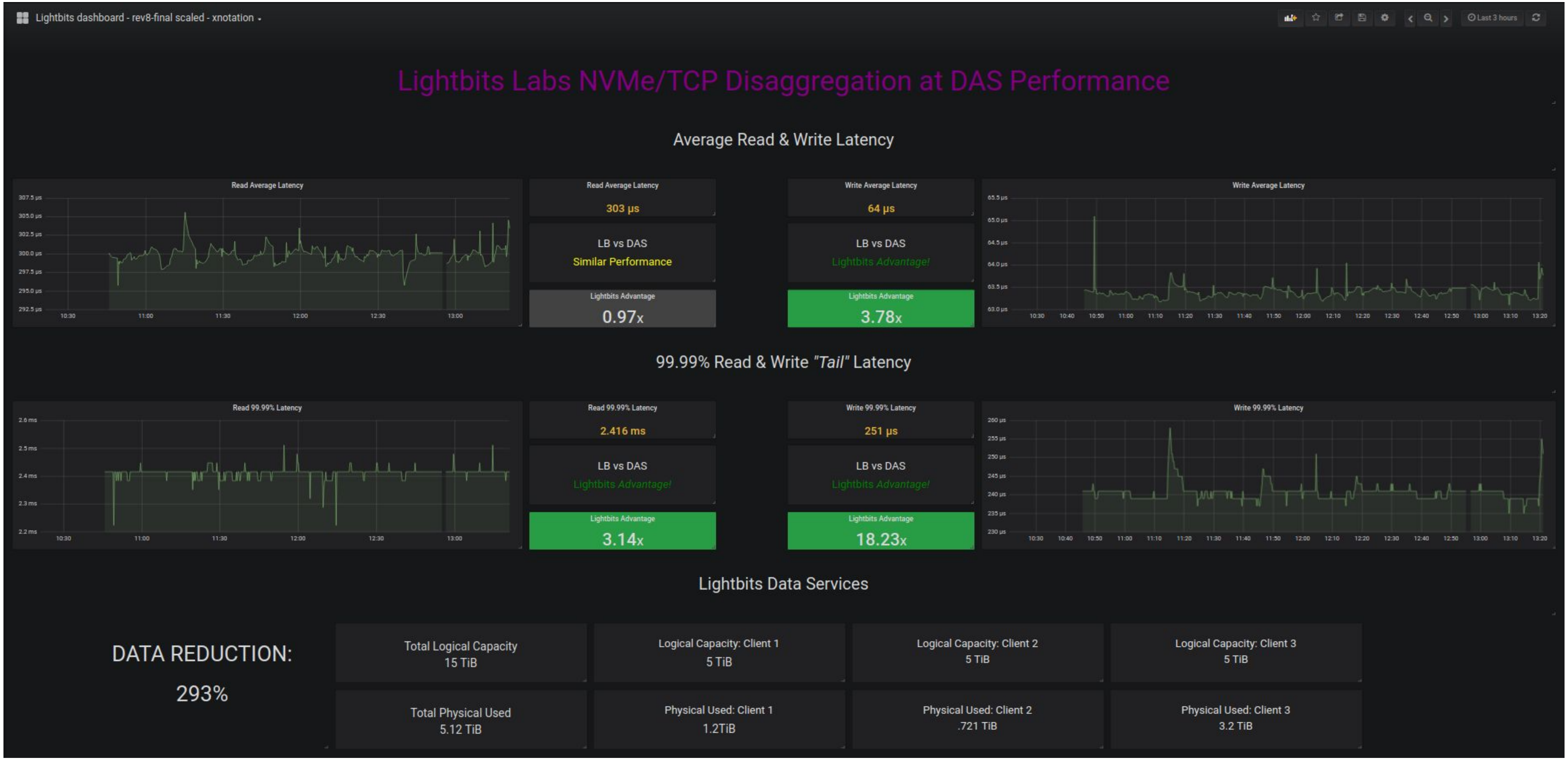
- Test is using 8 NVMe backend drives at 8k random 70/30 mixed workload



- Performance falls to the floor once features kink in..

Flash Memory Summit

Visit Lightbits Demo



Thank you!