packet processed storage in a software defined world

Ash Young

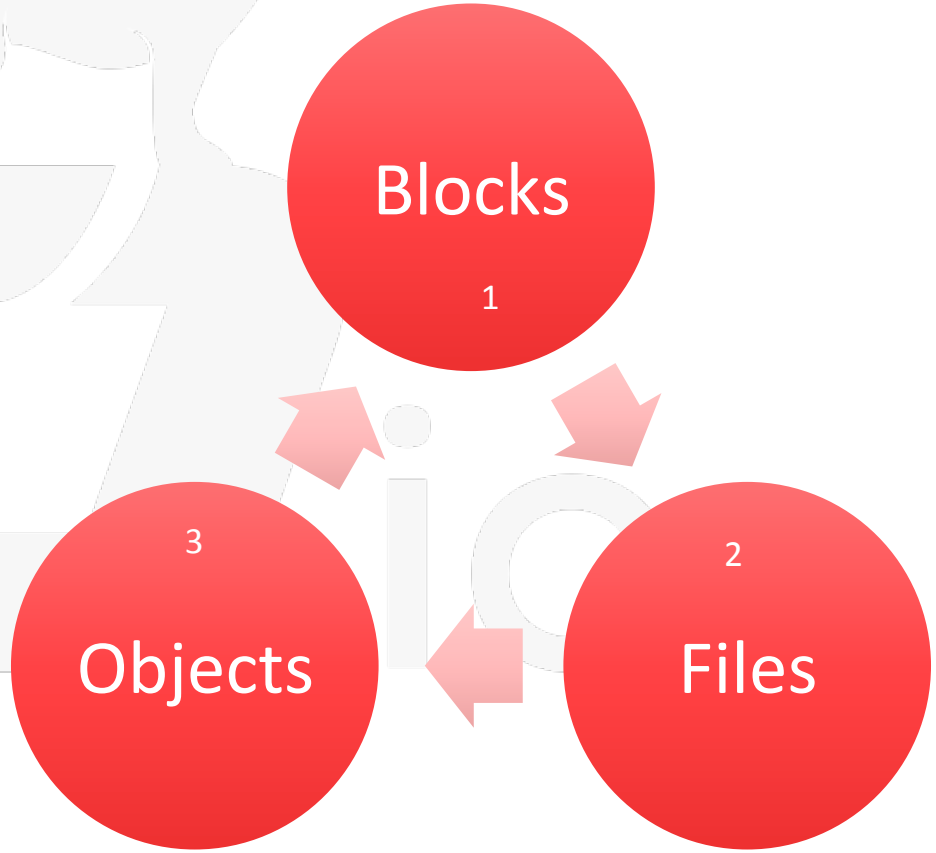
Intro

Abstract

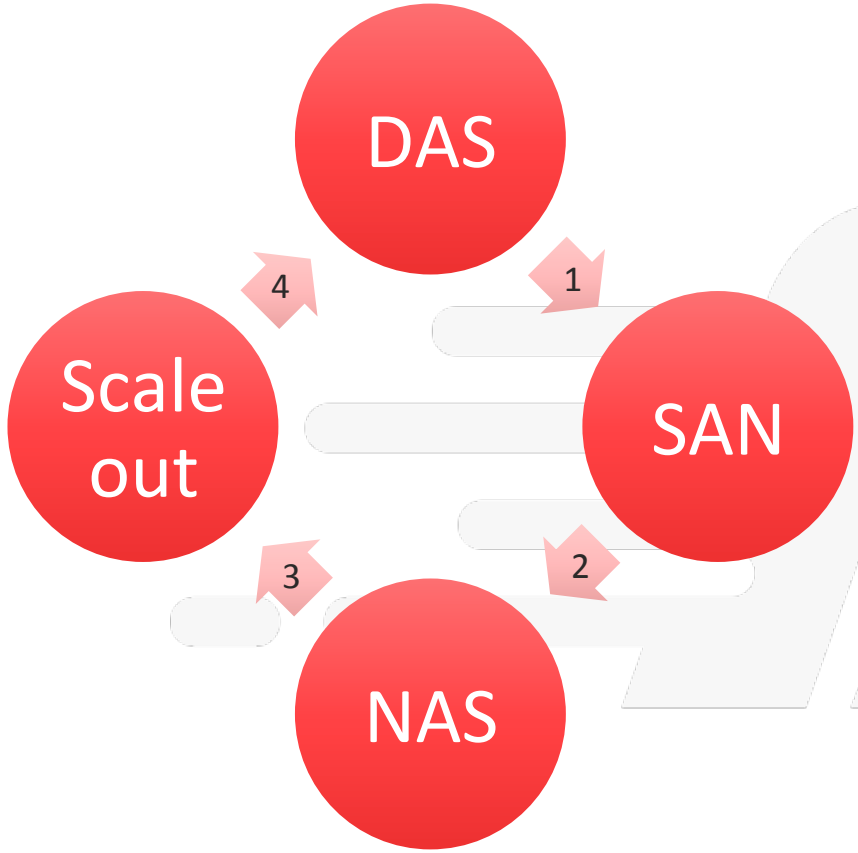
The storage arena has strived to significantly improve its storage capabilities over the years via technologies such as FCoE and NVMe. Simply reducing where the protocol sits in terms of the OSI model has proven to not be the solution. Similarly, eliminating the CPU from being the bottleneck for local storage I/O is also not the answer. Instead, we must assume that storage blocks or objects must ultimately be exchanged over a network; and if storage ultimately goes over the network, then we must re-think storage as packet processing.

Evolution of storage– data types

1. We started with simply accessing blocks because it was fast and cheap
2. We added added embedded file systems to make content sharing a lot easier
3. And now we're migrating to objects because it allows us to make the data more intelligent– especially in light of rapid growth



Evolution of storage– access modes



1. Changed data access paradigm to a shared model for amortization
2. Added multiple protocols and networking for easier access
3. Decided it was time to deal with concurrency and latency
4. And now we're back to DAS!!!

So, what's the issue?

- What goes around comes around…
- NVMe depends on PCIe
  - ✓ Not so shareable → amortization is a tradeoff
  - ✓ # of lanes forces additional tradeoffs in # of devices vs I/O
- Media is so fast that traditional async block I/O access is driving up CPU utilization

The bottom line…

- This isn't about access models
- Nor is about data or access types
- Faster media is highlighting our root issue in our endeavor for fast storage
- Our problem is unnecessary context switching between user space and kernel space!

Keeping Fast Data Fast…

At some point, we have to terminate

- There is a high degree of interest in optimizing the forwarding plane, for networking
- But unless our data never has to be persistent, we must also address how we terminate at the end points
  - ✓ Could be reading data from storage
  - ✓ Could be writing data to persistent storage
- Today's server packet processing techniques presume we're avoiding crossing the kernel/user space boundary

Keeping it moving…

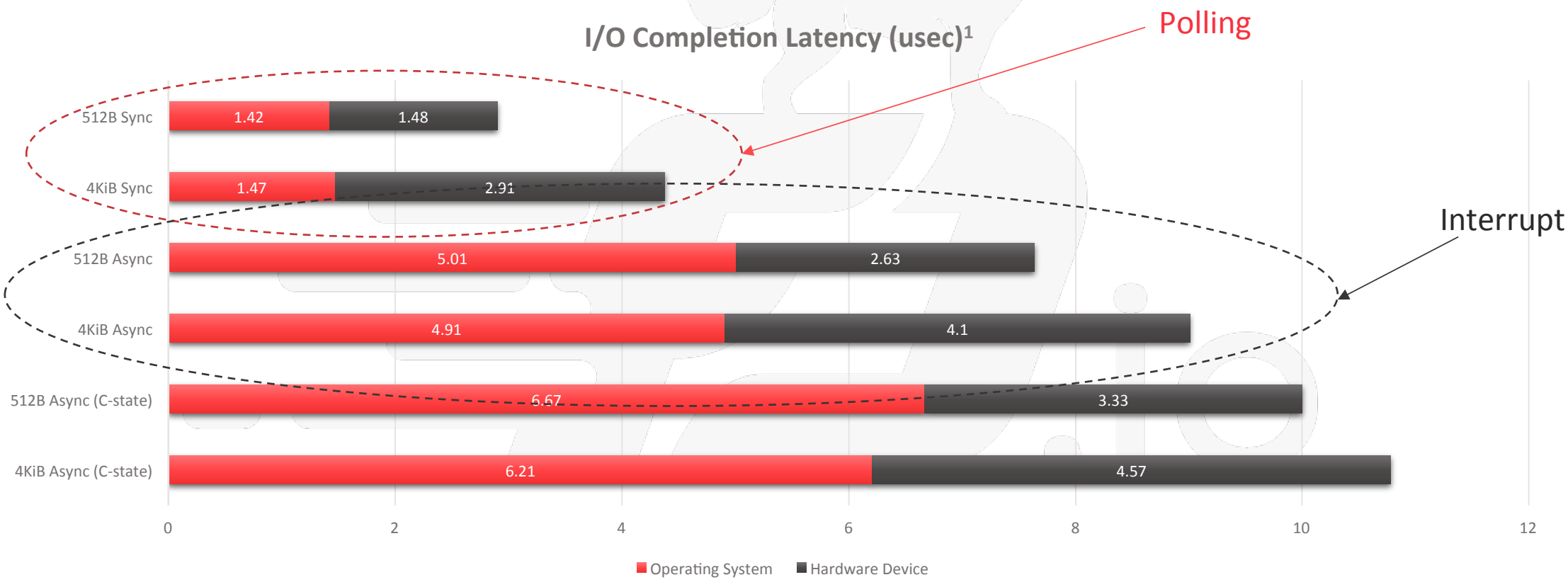
- We have fast NICs with drivers that can aggregate lots of packets from user space apps (e.g., Vector PMD for Intel's IXGBE)
- We have blazingly fast user space switching via FD.io's Vector Packet Processing (VPP) to get the packets from A to B
- And we have ways of getting these packets back off of the switch and into end node
- But we need to do this without creating a queueing issue

Ahh… but we have some building blocks

- VPP as the virtual switch for our VMs and containers
- Poll Mode Drivers (PMD) vs Async Block I/O kernel driver
  - ✓ IXGBE driver
  - ✓ Data Plane Developer Kit (DPDK)
  - ✓ Storage Performance Developer Kit (SPDK)
- We might have to do some additional file system tweaks (TBD)
- Also, need to look into the impact with object storage
- But block storage just got a whole lot sweeter!!!

Let’s whet the appetite…

Example of polling on random reads



[1] Yang , J., Minturn, D., & Hady, F. (n.d.). When Poll is Better than Interrupt

What else is there?

Once we start thinking of storage in terms of packets, it opens up new avenues for us

- ✓ We can start to think of how to dynamically route the traffic
- ✓ We can perhaps give content type different priority levels
- ✓ Perhaps we can re-model how we think of high-availability or even distributed storage
- ✓ Perhaps the switch can be a storage device or vice versa

Next steps

- Get involved at wiki.fd.io
- Currently scoping out a storage sub-project
- Looking for some collaboration from all communities
  - ✓ OPNFV
  - ✓ OpenStack
  - ✓ Minio
  - ✓ Others
- Reach out to me via email at ashlee@wildernessvoice.com