Performance Optimization: Rook on Kubernetes

Ryan Tidwell

Senior Software Engineer

rtidwell@suse.com

Mark Darnell

Senior Product Manager

mdarnell@suse.com

Agenda

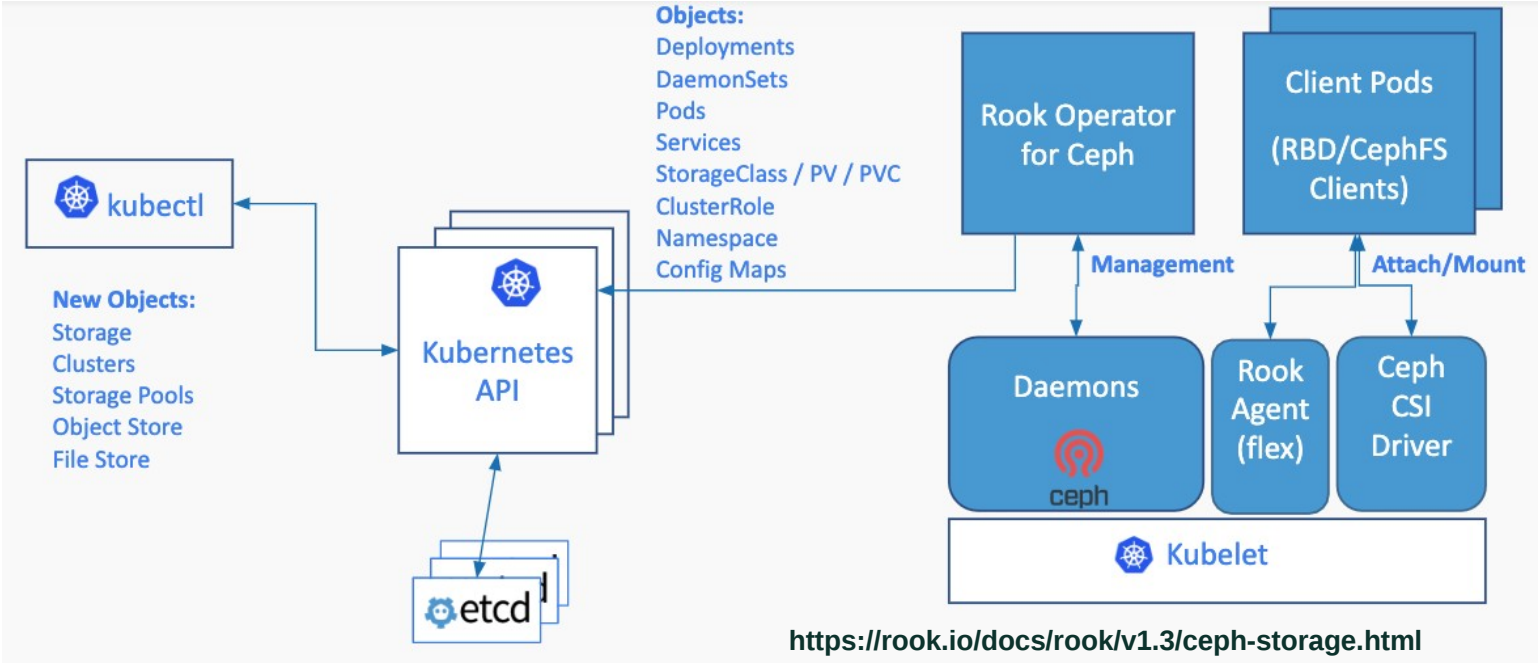
- Introduction
- Benchmark Environment
- Benchmark Methodology and Results
- Insights
- Future Work
- Q&A

Introduction

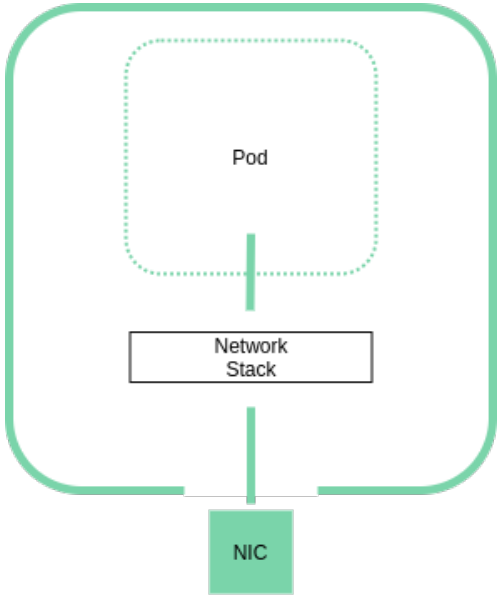
Motivation

- The choice of networking technology will affect the performance of modern storage systems like Ceph
- Develop an understanding of how the choice of CNI plugin affects a Rook+Ceph cluster
- What can we learn and apply to projects like Calico, Cilium, Multus, NSM, etc. ?

Rook+Ceph Basics



Kubernetes Networking: The Basics



- Use of standard linux interfaces such as veth, macvlan/ipvlan, physical interface, SR-IOV VF, etc.
- Host networking allows direct, native access to the node's network devices
- Let's explore how different technologies stack up

Benchmark Environment

Hardware Specs

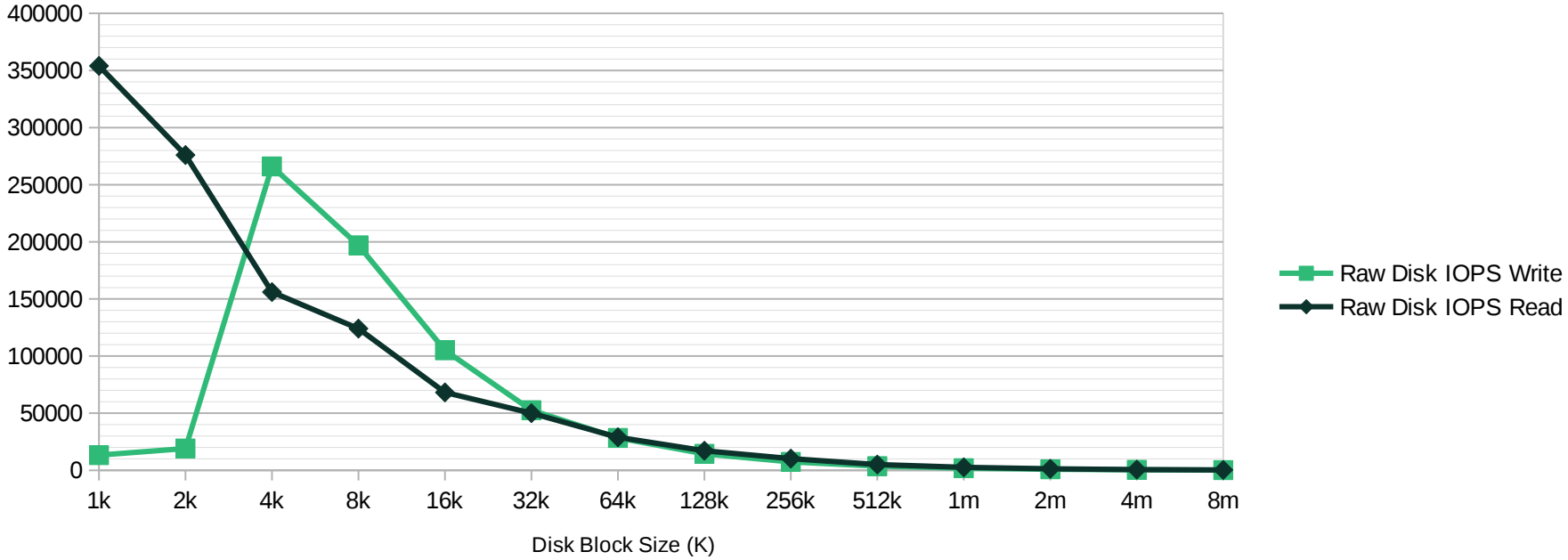
Ceph Nodes

- 2x 8-Core Intel Xeon E5-2620
- 64GB Memory
- Intel DC P3700 NVMe 800GB SSD
- Mellanox MT27800 100Gb NIC

Client Node

- 2x 8-Core Intel Xeon E5-2620
- 64GB Memory
- QLogic QL4500 25GbE NIC (Bonded Pair)

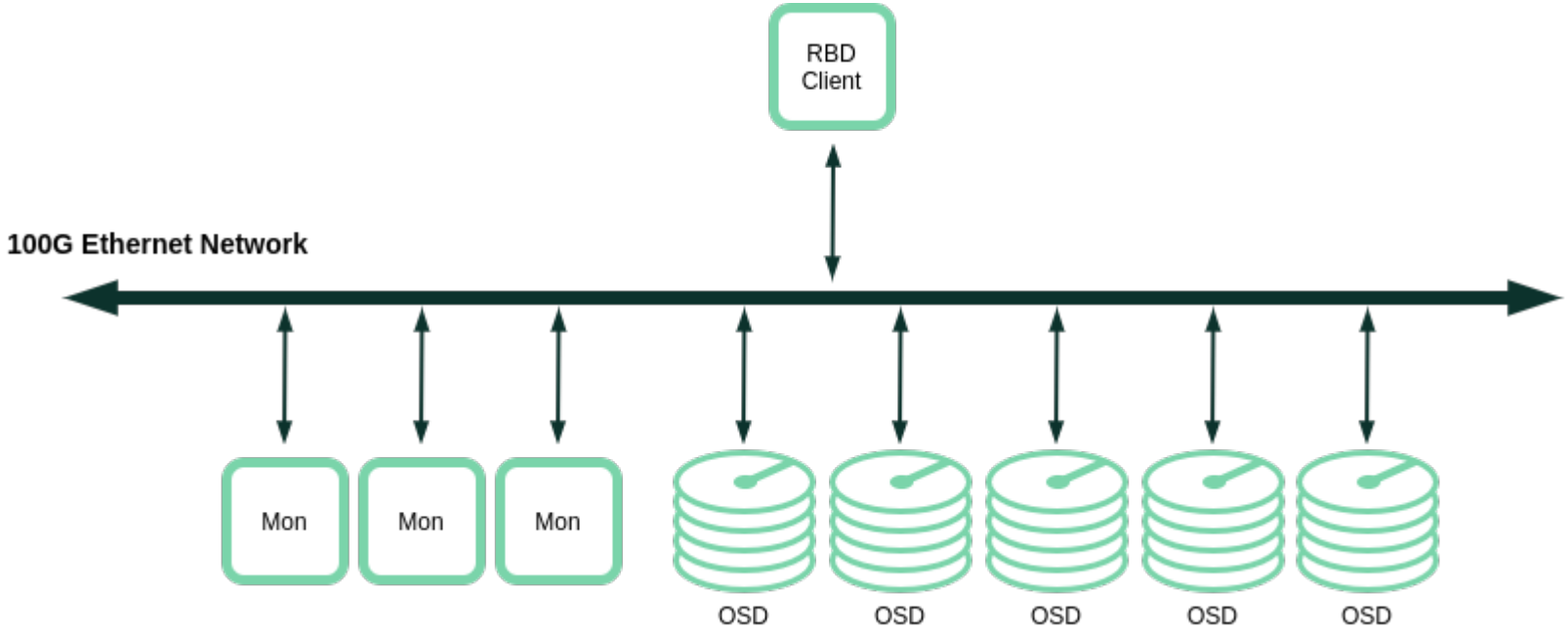
Hardware Specs: SSD Baseline



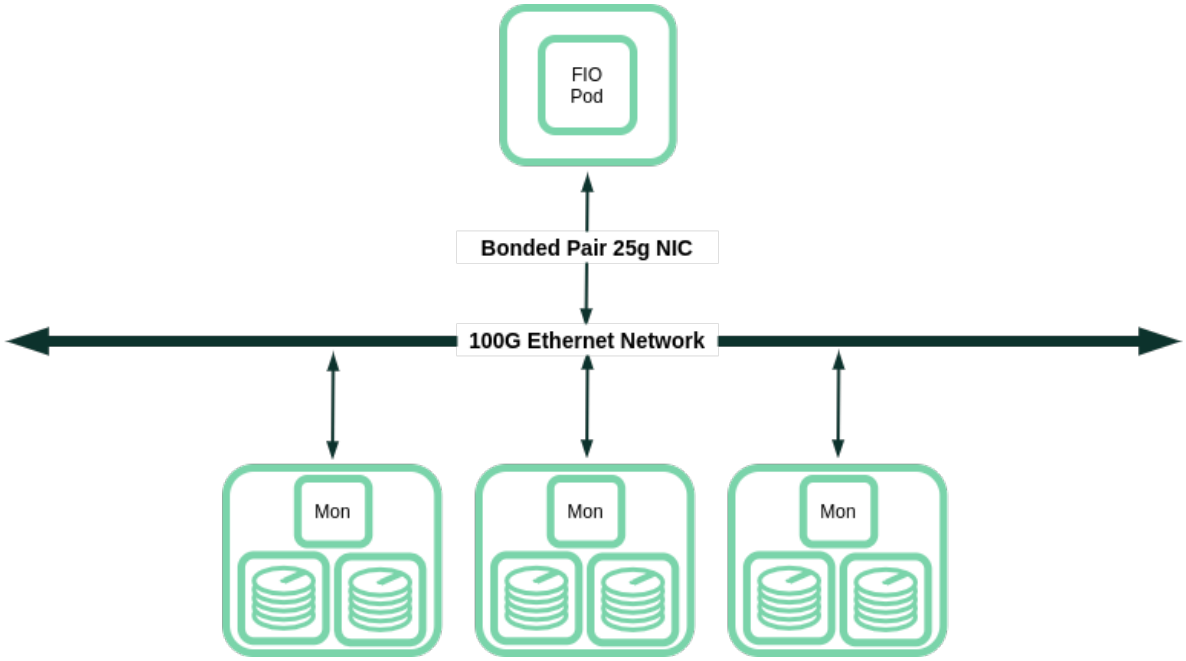
Software

- SLE 15 SP2 (Kernel 5.3.18)
- Rook 1.3
- Ceph 14.2.6
- Cilium 1.7
- Calico 3.14

Logical Cluster Design

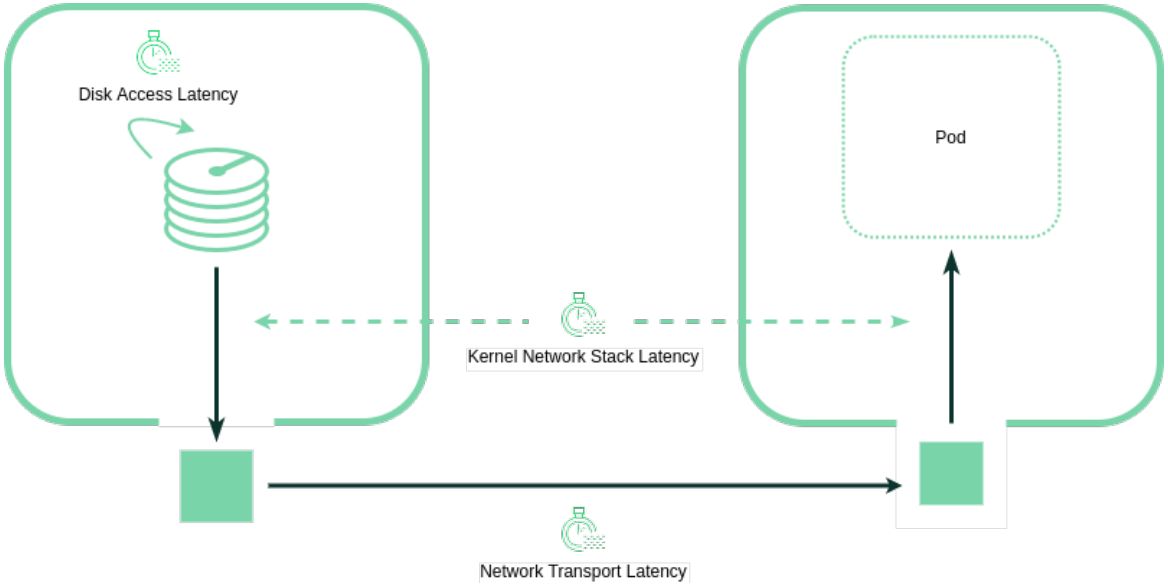


Physical Cluster Layout



Benchmark Methodology and Results

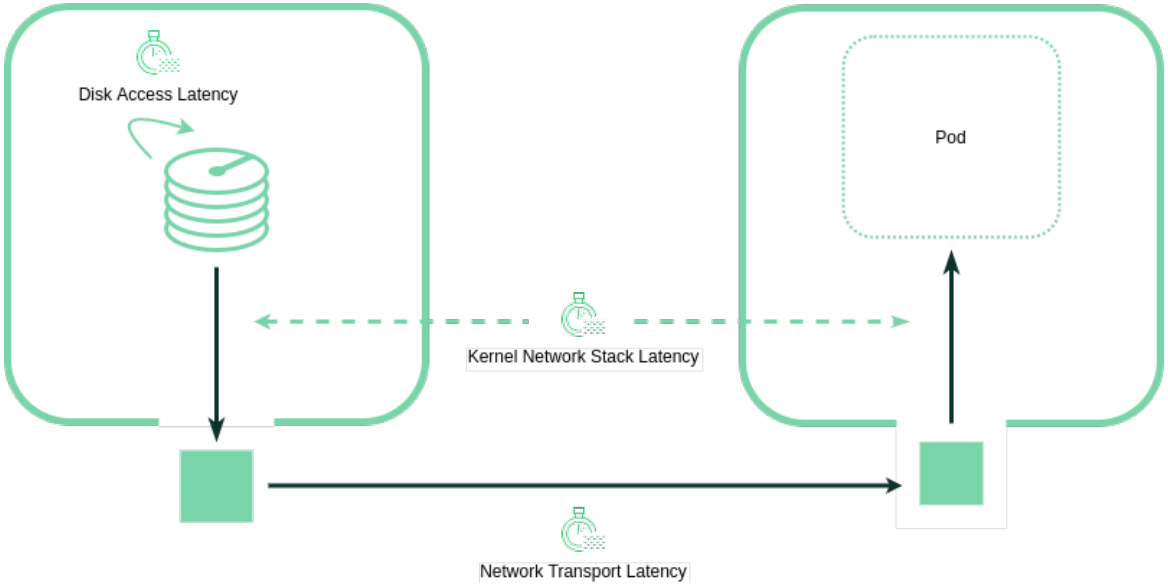
Latency is the Enemy (says the network engineer)



- Latency is variable delay inserted by components in a pipeline
- Look to the left to see where latency is added
- Disk access latency is not influenced by CNI and network configuration
- Network transport latency is influenced by bandwidth, congestion, bonding & switch configuration, etc.
- **Kernel latency is highly dependent on CNI configuration**

Total Latency = Disk Access + Network Transport Latency + Kernel Network Stack Latency

Storage folks talk in terms of IOPS



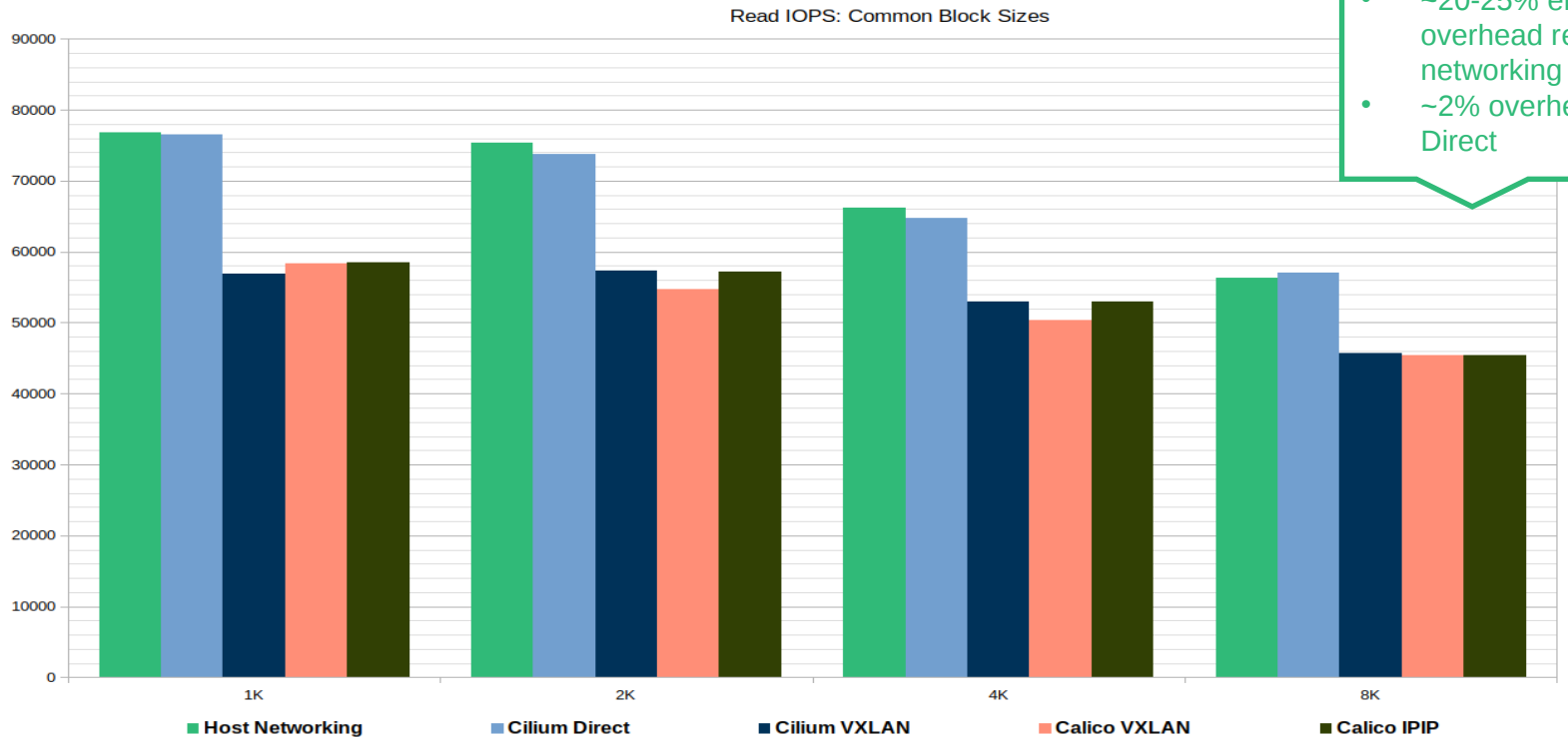
- IOPS = I/O per Second
- A single I/O operation incurs all network-related overhead
- Less network latency = more IOPS

Total Latency = Disk Access + Network Transport Latency + Kernel Network Stack Latency

Methodology

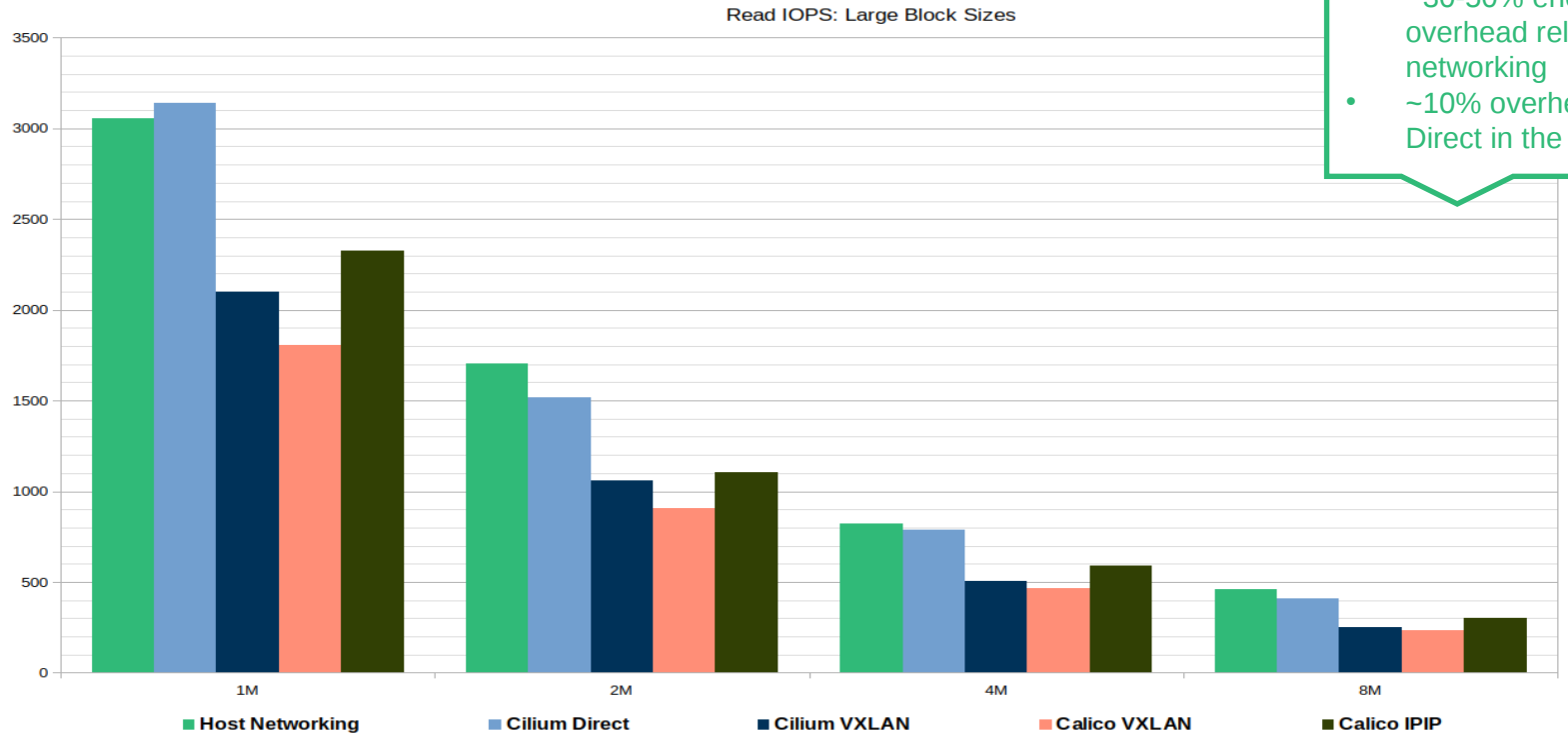
- Scientific method – change one variable; hold all others constant
- Optimize base system – jumbo frames and make disk faster than network
- Note - disk access time is constant regardless of CNI plugin used
- Run a single RBD client on dedicated node measuring IOPS, latency, peak bandwidth demands
- **Execute the prior step for each CNI plugin under evaluation**

Read Benchmarks



- ~20-25% encapsulation overhead relative to host networking
- ~2% overhead for Cilium Direct

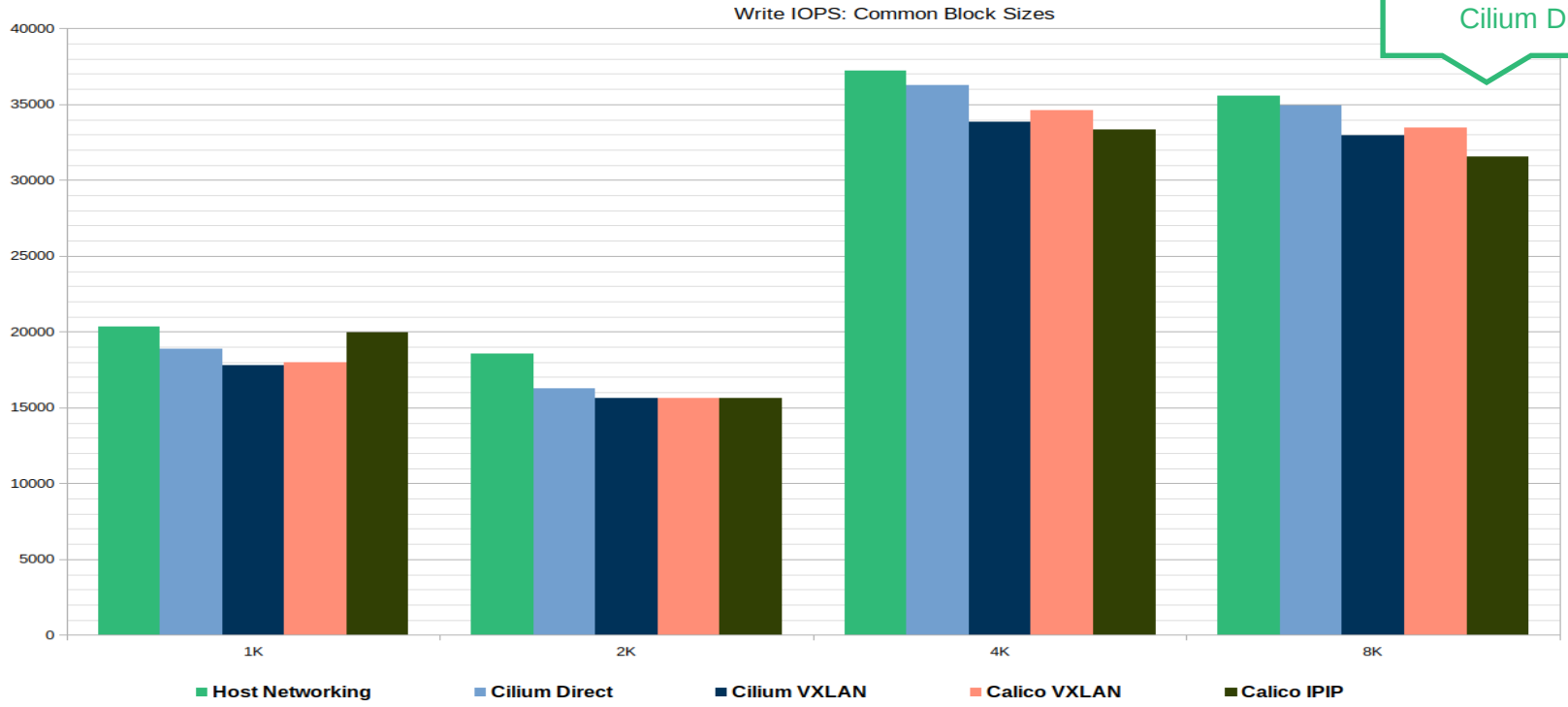
Read Benchmarks



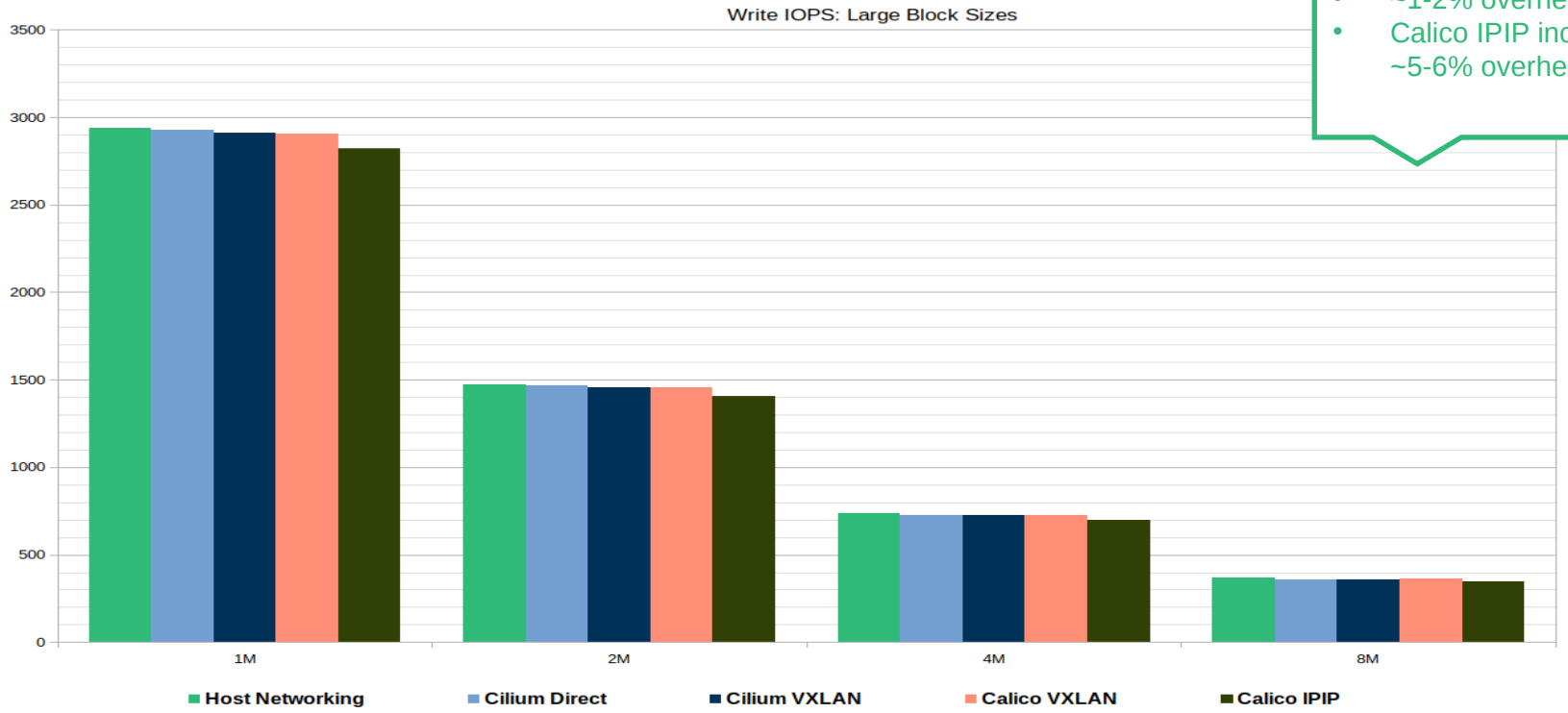
- ~30-50% encapsulation overhead relative to host networking
- ~10% overhead for Cilium Direct in the worst case

Write Benchmarks

- ~5-15% encapsulation overhead
- ~2% overhead for Cilium Direct

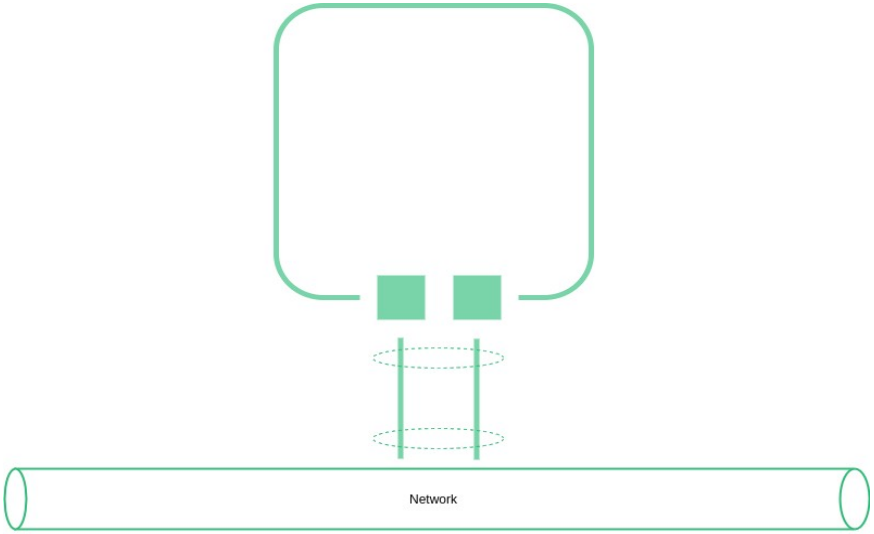


Write Benchmarks



- ~1-2% overhead
- Calico IPIP incurs ~5-6% overhead

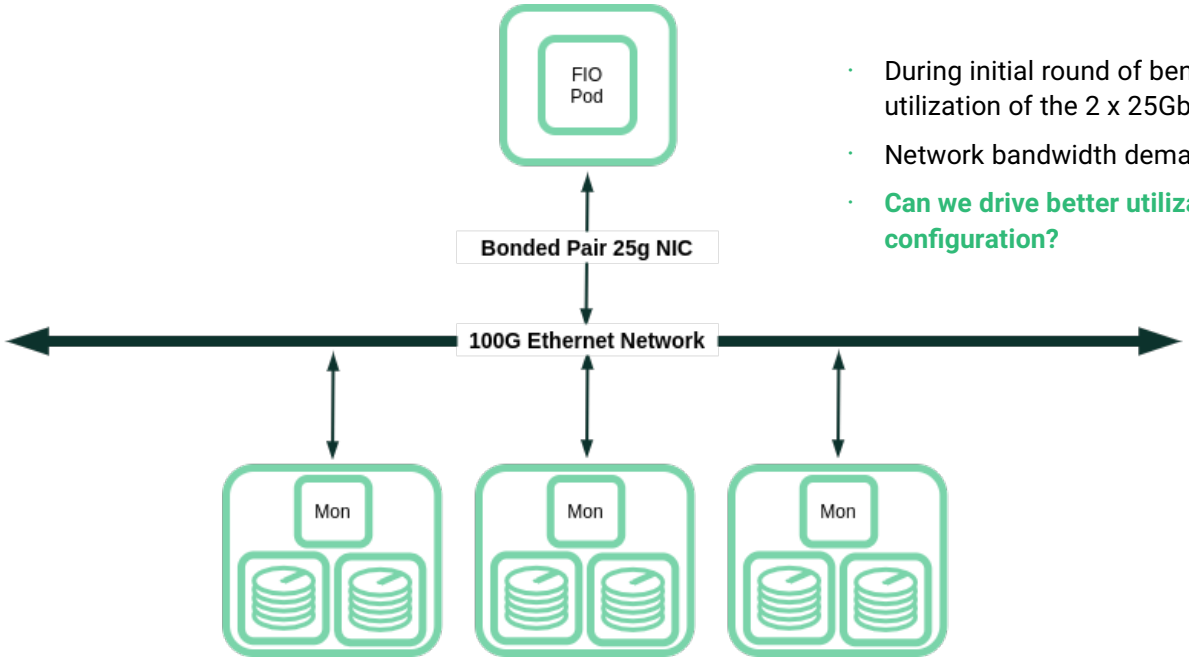
A Word About Bonding...



- Bonding makes multiple physical interfaces appear as a single interface with multiple "channels" working behind the scenes
- Linux supports a myriad of bonding modes, we used LACP (mode 4)
- Each bonding mode has its own tunable paramters
- **How traffic is balanced across channels of a bond will influence performance**

LACP hash policies determine which channel packets are sent and received on

Bonding And Why It Matters...



- During initial round of benchmarks we observed poor utilization of the 2 x 25Gb bond
- Network bandwidth demands maxed out at ~25Gbps
- **Can we drive better utilization by tuning the bonding configuration?**

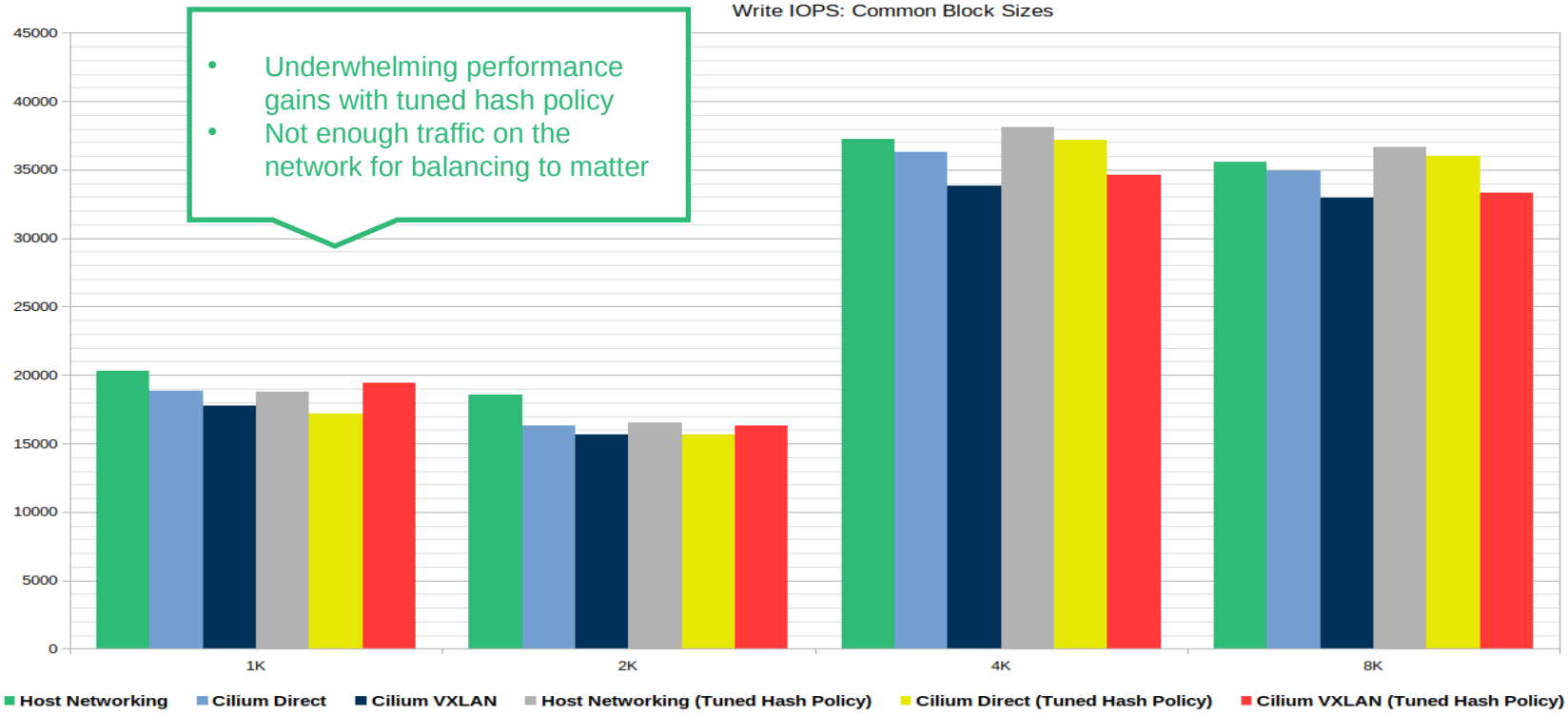
Tuned Bond Settings

	xmit_hash_policy
Host Networking	layer3+4
Cilium VXLAN	encap3+4
Cilium Direct	layer2+3

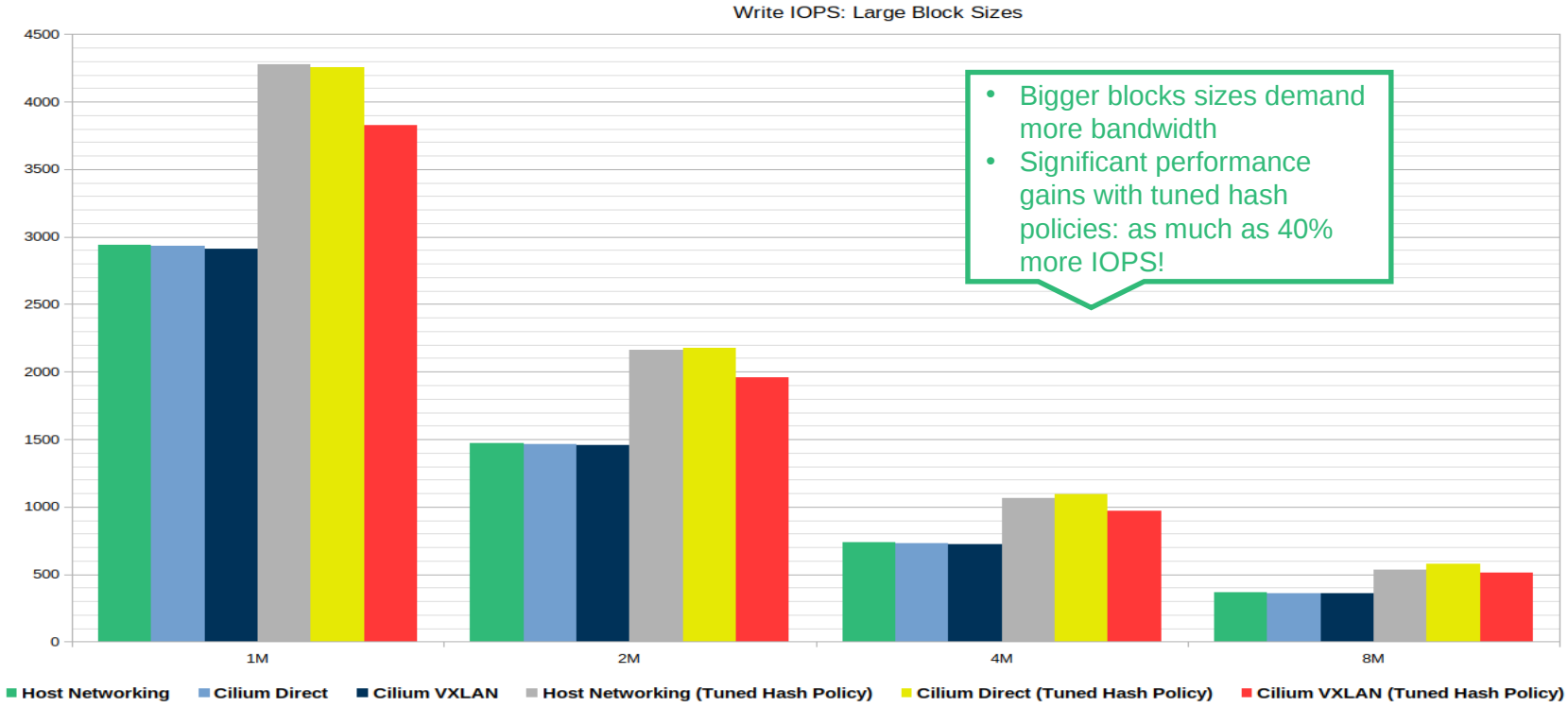
- **These settings yielded the most dramatic performance gains in our cluster**
- These settings were optimized for this specific cluster. Factors such as cluster size, bonding mode, ToR capabilities, etc. may call for different settings

Write Benchmarks: Tuned Hash Policies

- Underwhelming performance gains with tuned hash policy
- Not enough traffic on the network for balancing to matter



Write Benchmarks: Tuned Hash Policies



- Bigger blocks sizes demand more bandwidth
- Significant performance gains with tuned hash policies: as much as 40% more IOPS!

Insights

What To Know About Bonding

- Bonding modes and LACP transmit hash policies can make a significant difference
- Tuning *xmit_hash_policy* on the node and corresponding settings on the ToR switch enable better balance of traffic across channels in the bond
- As network bandwidth demands rise, so does the importance of bonding configuration
- **These configurations will be specific to your environment and depend on factors such as CNI configuration, scale, and hardware capabilities**

General Recommendations

- Overlays and encapsulation limit IOPS by introducing latency, avoid encapsulation where possible
- Bandwidth demands of a single client are highly correlated with block size – **Align with native block size!**
- When using bonds, pay attention to hash policies and load balancing settings on both the host and ToR switch – **Tuning these settings can yield significant performance gains!**

Selecting a CNI Plugin

- The best hardware without the best CNI will leave you wanting
- Host networking is easy and performs well but...a bit insecure
- IOPS matter, and un-necessary network latency hurts IOPS
- CNI policy enforcement may be better than Ceph policy enforcement

Future Work

Where To Go Next?

- Multus w / SR-IOV
- Calico BPF
- Explore improved inline instrumentation
- Scaling the workoad in the cluster – more clients & more storage nodes

Q&A

Thank You