DigitalOcean

How we Operate Ceph at Scale

Ceph Day NYC 2024

Matt Vandermeulen, Storage Systems

Contents

- Who are we?
- Ceph use at DO
- Automation
- Cluster Operations
- Observability
- Reflection
- Q&A

A brief history

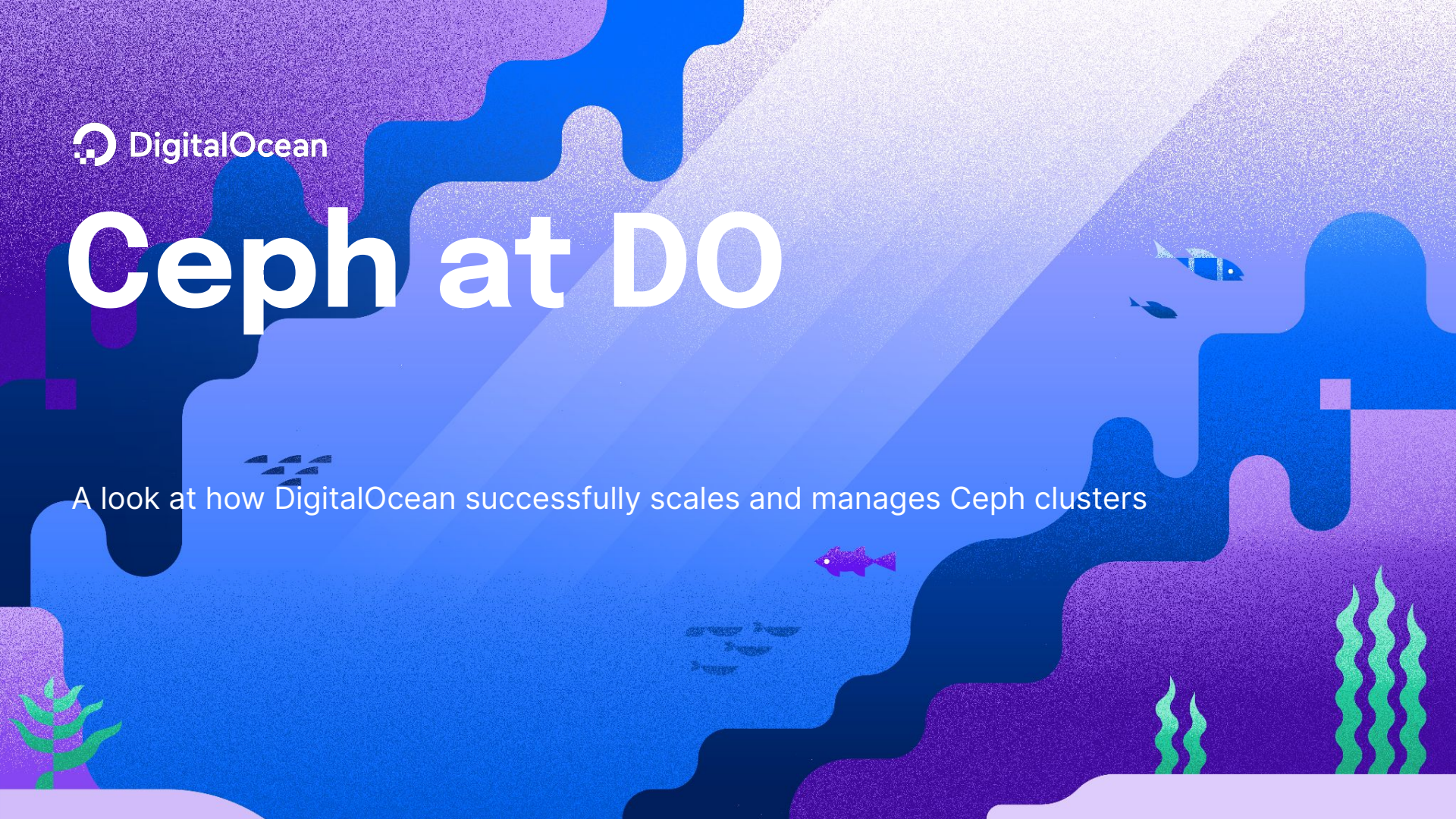
What is DigitalOcean?

- Founded in 2012
- Core concept: Simplicity
- SSD backed VM was very attractive in 2012
- Four years later we introduce Volumes (block)
- Many more products since, including Spaces (object) and several SaaS offerings such as DBaaS, DOKS (k8s), Serverless Functions, and managed hosting with Cloudways
- More than a dozen datacenters in 9 regions
- IPO in March 2021

2012 Droplet: Introduction of the SSD-backed VM

2016 Volumes: Ceph backed detachable droplet storage

2017 Spaces: Ceph backed object storage


DigitalOcean
Ceph at DO
A look at how DigitalOcean successfully scales and manages Ceph clusters

Ceph Usage

- Block storage (Volumes)
- Object storage (Spaces)
- Other teams consume Volumes and Spaces for many products

Quick Stats

50 Cephs

47 Production Pacific clusters

8 Staging clusters (some Reef!)

200+ PB

Total raw Ceph capacity

12+ PB in our biggest cluster

28,500+

OSDs in the fleet across **1,600+ nodes**

Automation: Tooling

Chef runs on a staggered cron throughout the fleet ensuring packages are installed as expected, and other things we don't manage, like the kernel, are up to date.
We heavily rely on Ansible/AWX to do many maintenance operations that are long running, cluster deployments, augments, etc.
There are many bash scripts still written and used as necessary. Don't let perfect be the enemy of progress.

01 Chef for general configuration management

02 Ansible via AWX for just about everything Ceph

03 Hacky bash scripts or other one-off use case specific tools

Automation: Ansible Use Cases

- Net-new Ceph cluster deployment
- Existing Ceph cluster augment: Nodes and disks
- Node preflight checks
- Node reboots: On demand, and when required (i.e. kernel updates)
- Maintaining node-local and centralized Ceph configuration
- Ceph upgrades, and other long-running operations
- OSD restarts
- Node/Cluster decommission
- Lots of utilities/goodies

Automation: Snowflakes

- **Automation thrives on consistency**
- **Snowflakes, being unique, are not consistent**
- **Snowflakes include**
  - CPU, RAM, Disk models
  - Network configs
  - Centralized Ceph configuration
  - That one script you forgot was running…
- **Melt your snowflakes!**

Operations: Deployment

- **Create the initial `monmap`, deploy monitors, establish quorum**
  - This requires some integration with our secrets solution
- **OSDs are created ahead of time, then deployed in parallel**
  - CRUSH is populated first, then all OSDs deployed in parallel
- **Deploy and start OSD containers**
- **Verify cluster is healthy and bored**
- **Future: Zero-intervention deployment**

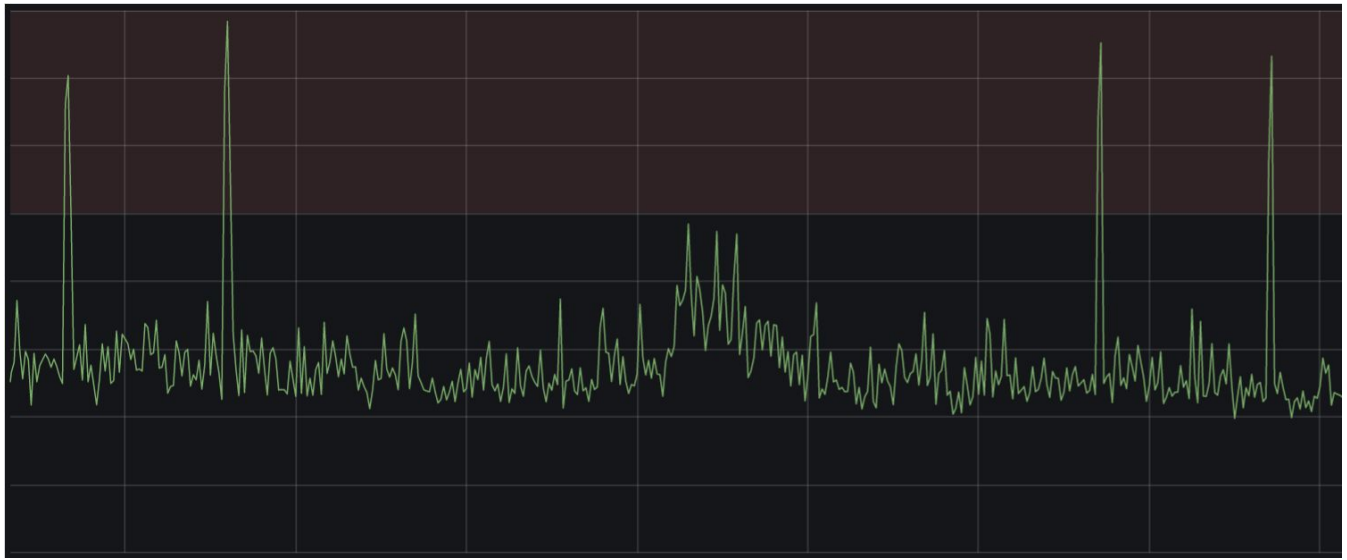
Operations: Augment

- **Block and Object used to be slightly different**
  - Block used to slowly upweight OSDs over time to mitigate peering latency
- **We now cancel all remapped backfill and slowly undo them**
  - First set `nobackfill` and `norebalance` OSD flags
  - Makes use of `pgremapper`
  - Mostly to maximize concurrency and minimize degradation
- **It's possible we could make use of the normal upmap balancer**
  - Less control of concurrency here
- **The way we approach growing our capacity has been changing**
  - We are moving towards deploying new clusters, and away from adding capacity to existing clusters

Operations: Maintenance

- Cluster augments
- OSD restarts, failures, flaps
- Node reboots, failures
- Peering storms

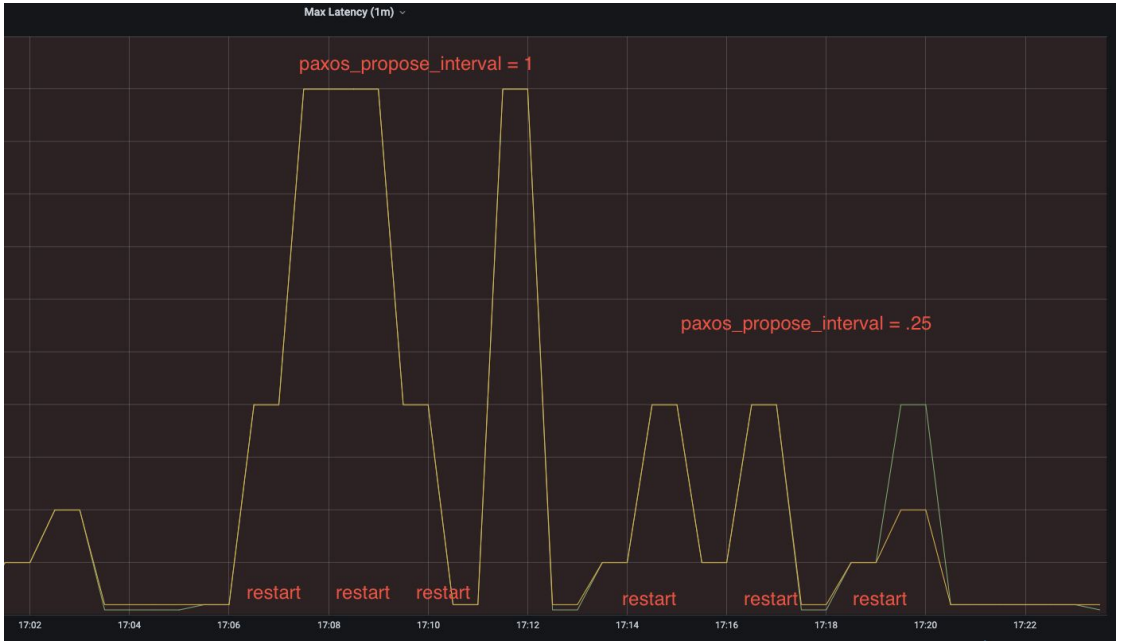
Operations: Peering Latency



Operations: Peering Latency

- In a mailing list message, Sage suggested reducing `paxos_propose_interval` from its 2s default
- We observed that this made peering latency (slightly) worse
- What if we don't let peering begin as the OSD process starts?
- Set `noup`, bring OSDs up, wait for `status = preboot`, unset `noup`
- Reduces CPU contention when starting a host full of OSDs at once
- Reduces `osdmap` updates, too!
- Now reducing `paxos_propose_interval` seems to help!

Operations: Peering Latency



Operations: Index Pain

- **Our Spaces clusters have billions (and billions) of objects**
- **RGW index layer does not like buckets with millions of objects**
  - Rule of thumb: 100k objects per shard
  - Too many shards make for poor listing performance
- **A lot of cpu time was spent in `rocksdb`**
- **RGW defaults are not tailored to handle this scale**

Operations: Index Squish!

- **Setting `osd_async_recovery_min_cost=0` has helped a lot**
- **With Nautilus, the `ttl` option in `rocksdb` was available**
- **This option forces compaction on data that reaches a given age**
- **We now have consistently higher load on our index nodes**
  - Absolutely worth the trade off
- **We were able to discard *tons* of junk data**
  - We have an upper bound on tombstone lifetime!
- **We disabled periodic compactions in favour of `ttl` compactions**
- **This has been a silver bullet!**

Operations: Index Squishier!

- **`rocksdb_cf_compact_on_deletion` options have been introduced**
  - Introduced by Mark in #47221
  - Available in 16.2.15 (disabled by default)
  - Available in 18.2.2 (enabled by default)
  - Didn't help upgraded clusters without resharded rocksdb
  - #55676 from Josh Baergen fixes this for the default column family
- **Compactions triggered by iteration instead of after a period of time**
- **Three options related to compact on delete**
  - A flag to turn it on and off
  - Sliding window value, a number of iterated entries
  - A trigger value, observed tombstones triggers compaction
- **We've backported this to our Pacific branch**
  - Used in conjunction with TTL on index
  - Compact on deletion used for Pacific fleet

Operations: Observability

- ceph_exporter vs prometheus module
- storexporter
- marigraph
- Alert on actionable things
- Make use of inhibition as appropriate

Observability: `marigraph`

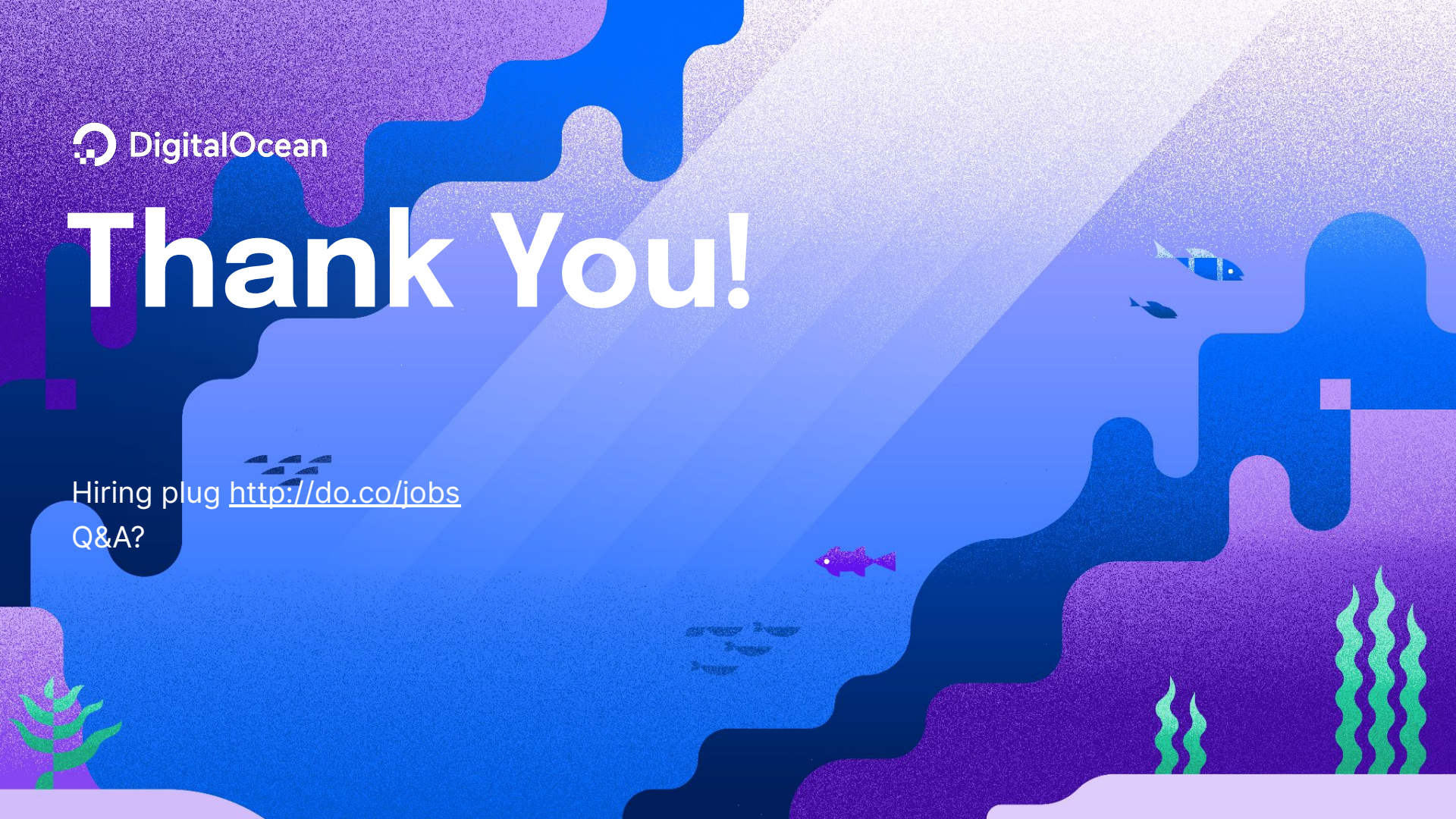
- **These dashboards are the number one stop for performance**
- **If we suspect any issue with a cluster, this is what we check first**
- **Dashboard graphs include:**
  - IO rate, timeouts
  - R/W latencies (avg/median, p90/p99/p100)
  - Histograms for those latencies
- **This has helped us find and track a few issues over the years**
  - Effects of flipping bluefs_buffered_io back and forth
  - Nautilus to Pacific post-upgrade write amplification
  - Higher latency ultimately caused by a backlog of discards

Observability: `marigraph`


DigitalOcean
Reflection
If we did it all again

Reflection

- Treat block and object clusters similarly
- Use a single source of truth… for any truth
- Finding that tradeoff between automation and services is tricky
- Melt your snowflakes as soon as possible!


DigitalOcean
Thank You!
Hiring plug http://do.co/jobs
Q&A?