



22-23 March 2018 | BEIJING

# **RADOS:**Improvements and Roadmap









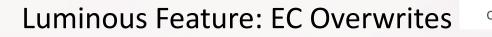
No more full-data journaling! (Twice as fast on streaming writes!)
Checksum all data

–Detect bit flips and media errors on every read, not just on scrub

In-line compression

"filestore splitting": totally gone!







•Erasure coded pools in Ceph can now overwrite data, not just append

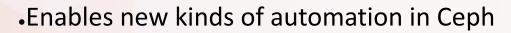
- –(Still not supported: omap, object classes)
- •RBD, CephFS enabled

 Performance: mostly higher latency, but also higher large-block streaming bandwidth (less data to write to disk!)

•Relies on BlueStore features to function







•Offloads some work from monitors (PGStats!), improving cluster scalability

Web dashboard

-See Lenz Grimmer tomorrow at 2:30 for the latest news







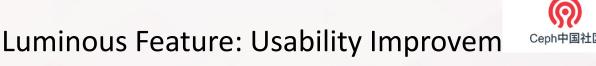
Luminous Feature: Performance/Predict

•"Client Backoff": if the client sends too much I/O for a stuck object/PG, the OSD sends a "backoff" message and throws it out

–Means we focus I/O on live objects instead of blocking the whole OSD/cluster

Async Recovery Deletes: objects are deleted during recovery (while the PG is live and serving I/O) instead of blocking peering







•Separate configuration defaults for SSDs and HDDs

"ceph osd [new|destroy]" commands simplify adding and replacing OSDs

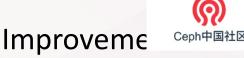
•Centralized version reporting for cluster and clients

-"ceph features" makes it clear if you are running different versions

-"ceph osd set-require-min-compat-client" disables features that break your clients

 Jumbo pings: OSD heartbeating will break if jumbo frames break your network







Luminous Feature: Usability Improveme

•PG overdose protection: limits the number of PGs per OSD, to prevent outof-memory conditions

 Device classes: CRUSH automatically generates different rules for SSDs and HDDs

-Good for easily putting index pools on SSDs

But does not work if you mix SSDs and HDDs in one pool





 CRUSH placement is pseudo-random, so the data is not evenly distributed across OSD nodes

•ceph-mgr Balancer module improves that data distribution

-"crush weight sets" for better balance that is compatible with old clients

-"upmap" for perfect balance, but requires Luminous clients



# Mimic: Centralized Config





Stored on monitors

One place to update, validate

•See history, diff running vs configured

•Manageable by dashboard in the future





Recovery blocks I/O

Backfill avoids this by putting OSDs outside the write path

Async recovery takes the same approach

Particularly helpful for highly contended objects, like RGW bucket indexes



#### Mimic: PG Merging





•pg\_num only able to increase in the past

•No longer worry about creating too many PGs

•Enables automatic tuning of PG count

 Groundwork for the future, so users setting up a pool do not need to know about PGs



#### Mimic: ceph-volume





Replacement for ceph-disk

•No udev – predictable, no longer race condition prone

Works with existing OSDs

•Uses lvm to store metadata for new OSD disks



## Future Work



 msgr2 protocol – on-the-wire encryption, groundwork for zero-copy, extensible for future changes

 Partial recovery – optimizations to recover changes to objects, rather than the entire object

•QoS – ongoing integration of dmclock

•Usability improvements – see dashboard and mgr talks



## OSD Future: Background



Storage devices becoming faster

•OSD uses too much CPU per op to take advantage of fastest NVMe devices

Callback-based style is difficult to maintain, allocation and copy heavy

Async reads needed for fully-featured EC pools



#### OSD Future: SeaStar





Designed for high performance

shared-nothing architecture among cores

•Core-local memory allocator

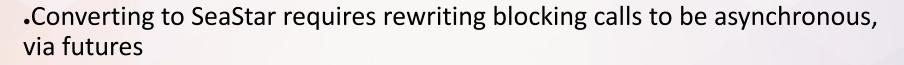
•No atomic operations or locks required unless crossing cores

Asynchrony based on polling for network and storage devices

•Future-promise programming model







•Start at the network layer, and go down

•To enable posix threads to talk to SeaStar, add 'alien' thread concept

 Eventually use DPDK for networking, SPDK for storage, and SeaStar for scheduling – kernel bypass, zero-copy I/O