Making Ceph Fast in the Face of Failure

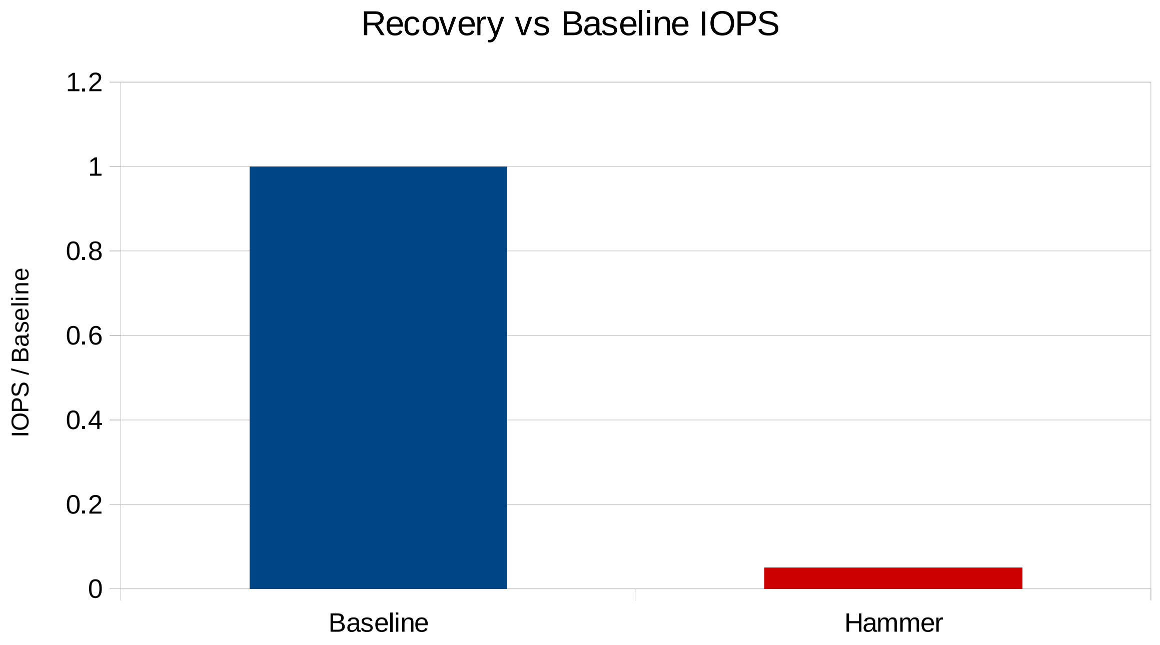
Josh Durgin and Neha Ojha

2018.08.28

The Dark Ages

The Triumph of Death - Pieter Bruegel the Elder

Hammer



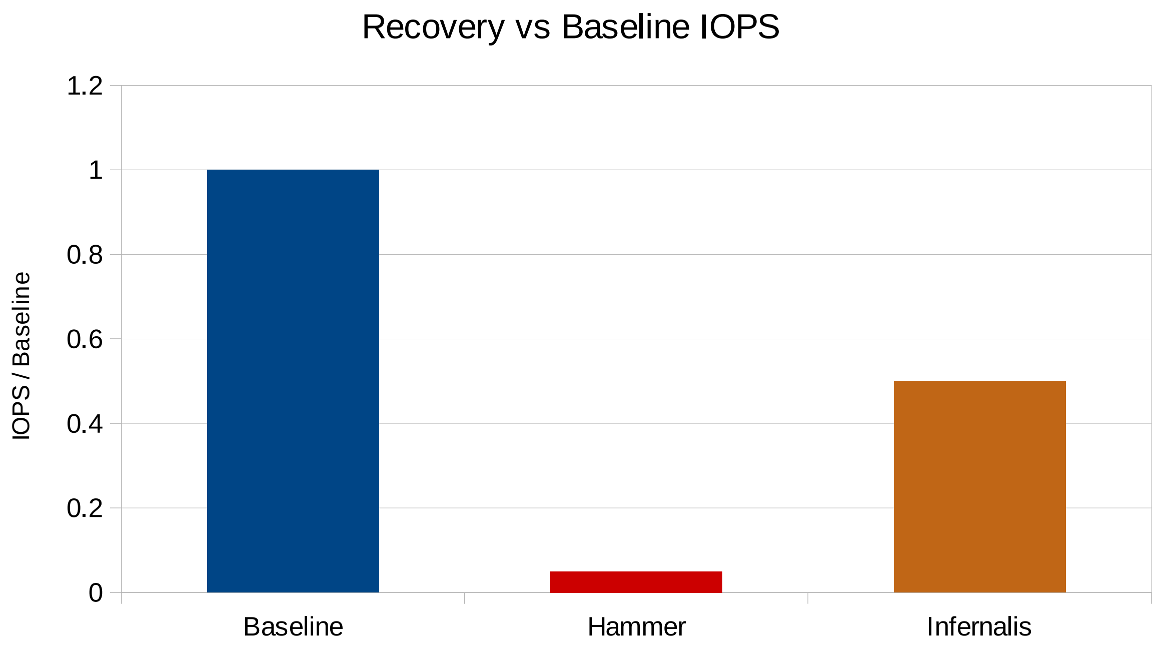
Recovery vs Baseline IOPS
IOPS / Baseline
1.2
1
0.8
0.6
0.4
0.2
0
Baseline
Hammer

Hammer

- Favored maximum recovery speed
- Default client impact was huge
- Tuning could help

```
osd max backfills = 10 → 1
osd recovery max active = 15 → 3
osd recovery op priority = 10 → 3
osd recovery max single start = 5 → 1
```

Infernalis


Recovery vs Baseline IOPS
IOPS / Baseline
1.2
1
0.8
0.6
0.4
0.2
0
Baseline
Hammer
Infernalis

Luminous – High-level Priority

Manual way to recover higher-level constructs, e.g. rbd images

pg force-recovery command

```
ceph pg force-recovery {pg-id} [{pg-id #2}] [{pg-id #3} ...]
ceph pg force-backfill {pg-id} [{pg-id #2}] [{pg-id #3} ...]
```

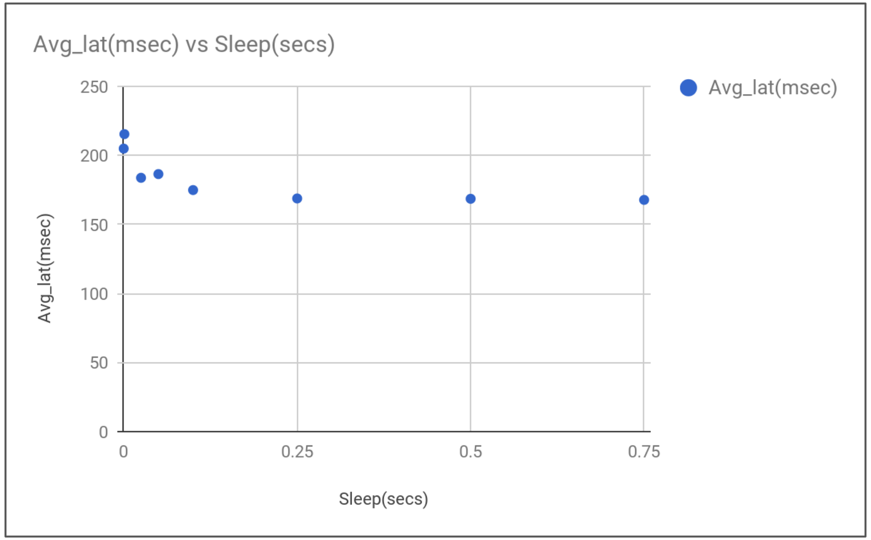
If you change your mind or prioritize wrong groups, use:

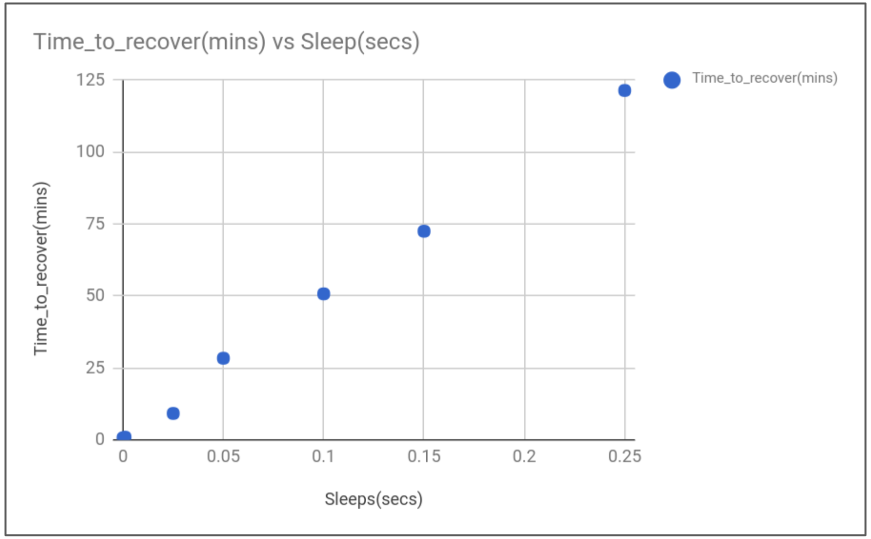
```
ceph pg cancel-force-recovery {pg-id} [{pg-id #2}] [{pg-id #3} ...]
ceph pg cancel-force-backfill {pg-id} [{pg-id #2}] [{pg-id #3} ...]
```

Luminous – Throttling Recovery

- osd_recovery_sleep
- changing this will shift the balance between recovery and client I/O
- Different configurations based on underlying hardware

Recovery Sleep - HDDs

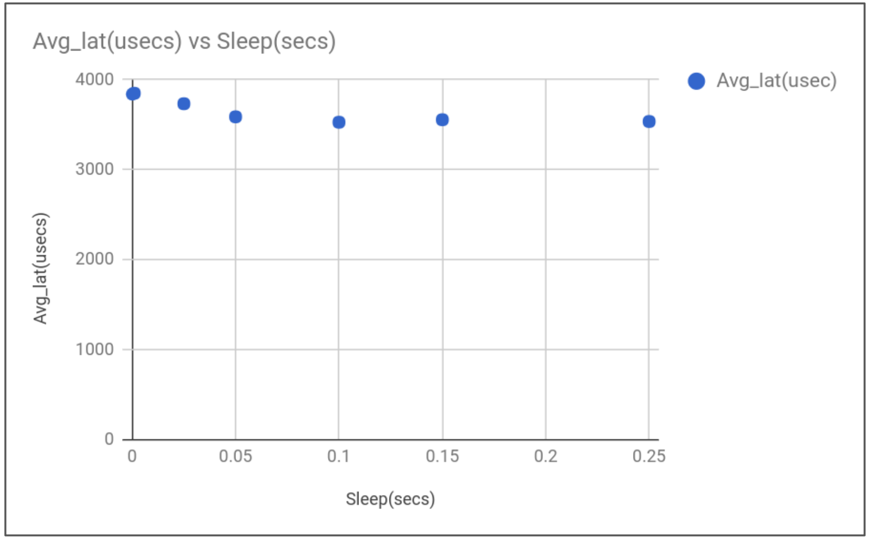


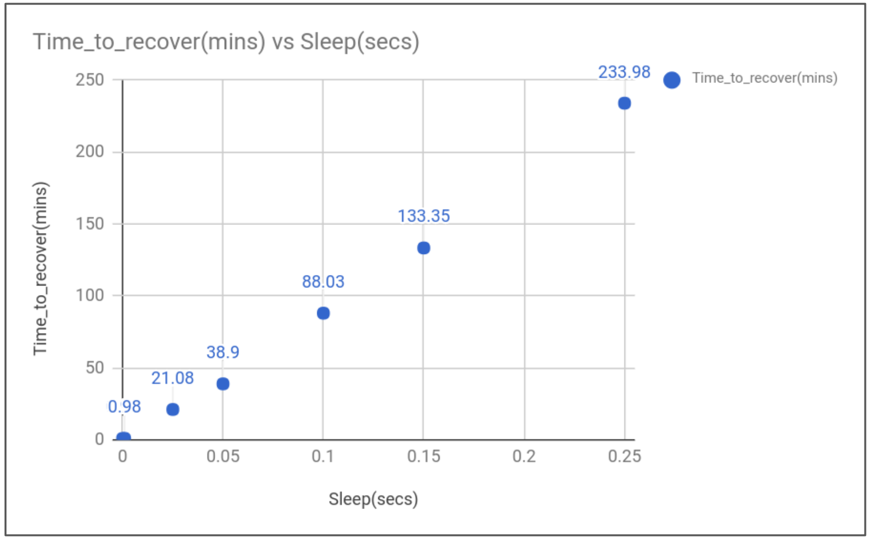


BlueStore on HDDs with Fio 4k random writes

- ➢ osd_recovery_sleep_hdd chosen as 0.1

Recovery Sleep - SSDs

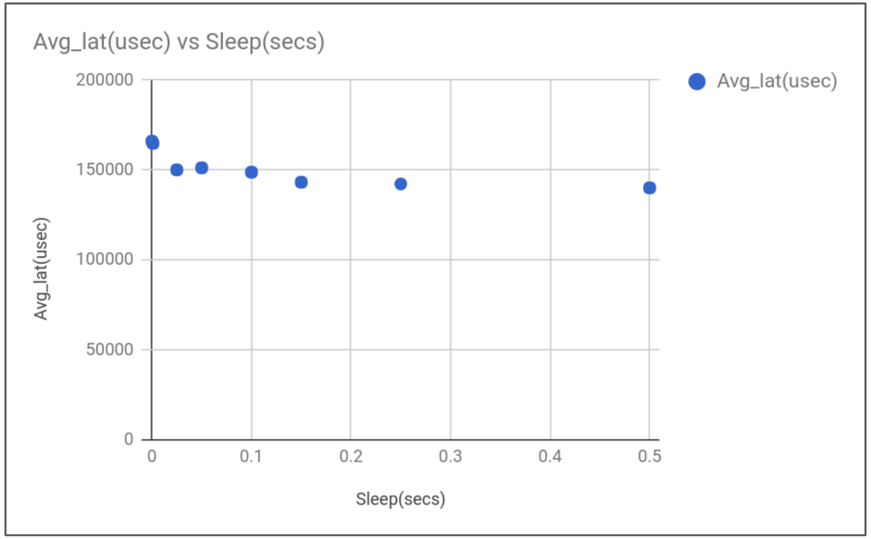


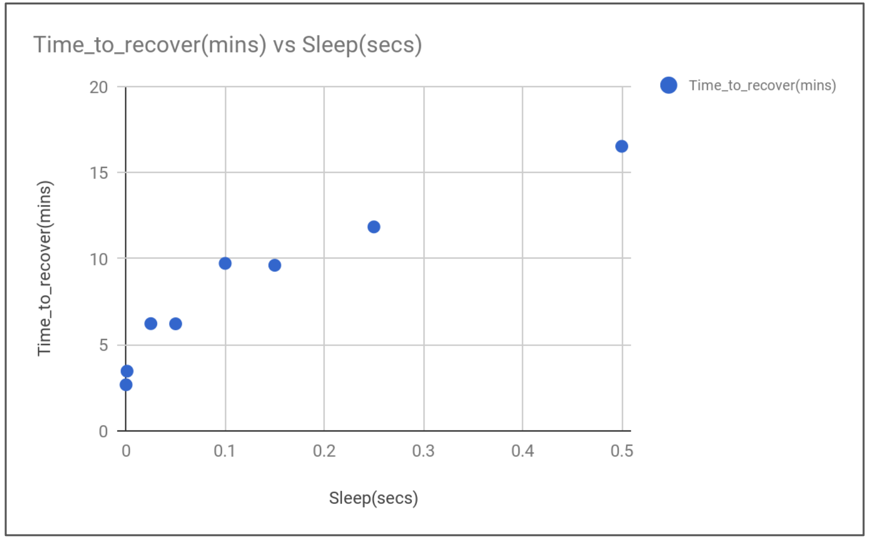


BlueStore on SSDs with Fio 4k random writes

- osd_recovery_sleep_ssd chosen as 0

Recovery Sleep - Hybrid

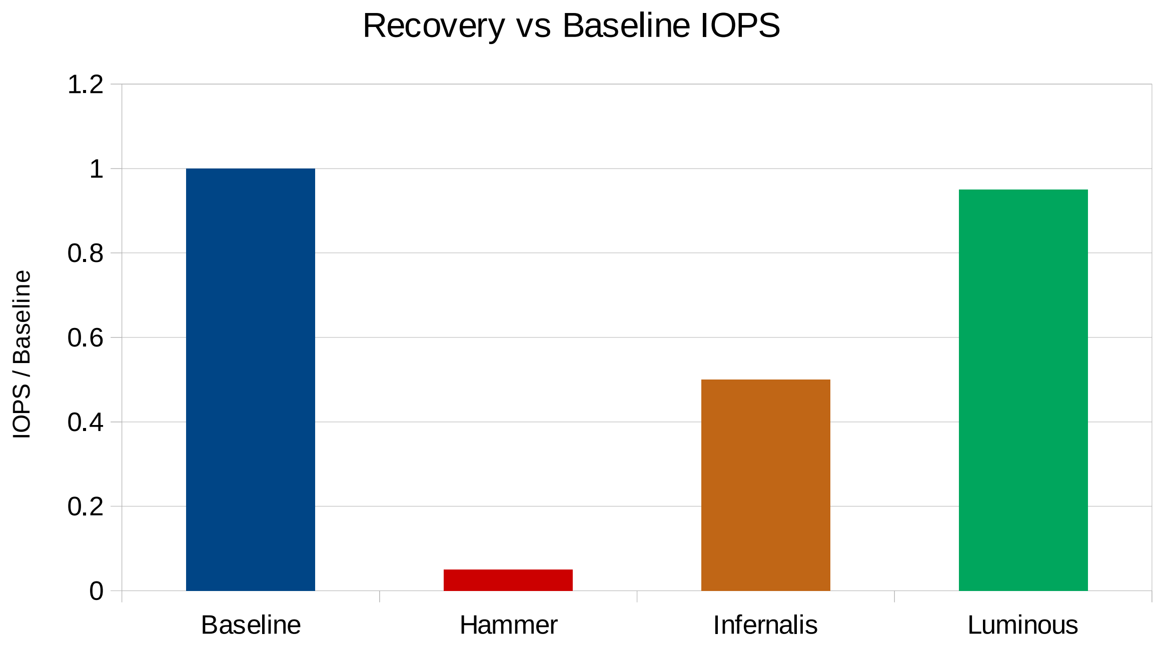




BlueStore on HDDs+SSDs with Fio 4k random writes

- ➢ osd_recovery_sleep_hybrid chosen as 0.025

Luminous Defaults are much better



Improving Latency during Recovery

Recovery in Ceph has been a synchronous process - it blocked writes to an object until it was recovered.

Problem: This increases write latencies and affects availability.

Solution in Mimic: Asynchronous Recovery

Do not block writes on objects, which are only missing on non-actingset OSDs.

Perform recovery in the background on an OSD, out of the acting set, similar to backfill, and use the PG log to determine what needs recovery.

When do we perform asynchronous recovery?

Async recovery targets - OSDs that are not part of the acting set and are chosen based on the following:

- approximate magnitude of the difference in length of logs is used as the cost of recovery, async recovery targets have higher cost to recover
- threshold ***osd_async_recovery_min_pg_log_entries***(default value=100) is used to determine when asynchronous recovery is appropriate
- ***min_size*** replicas should be available

Recovery Experiments

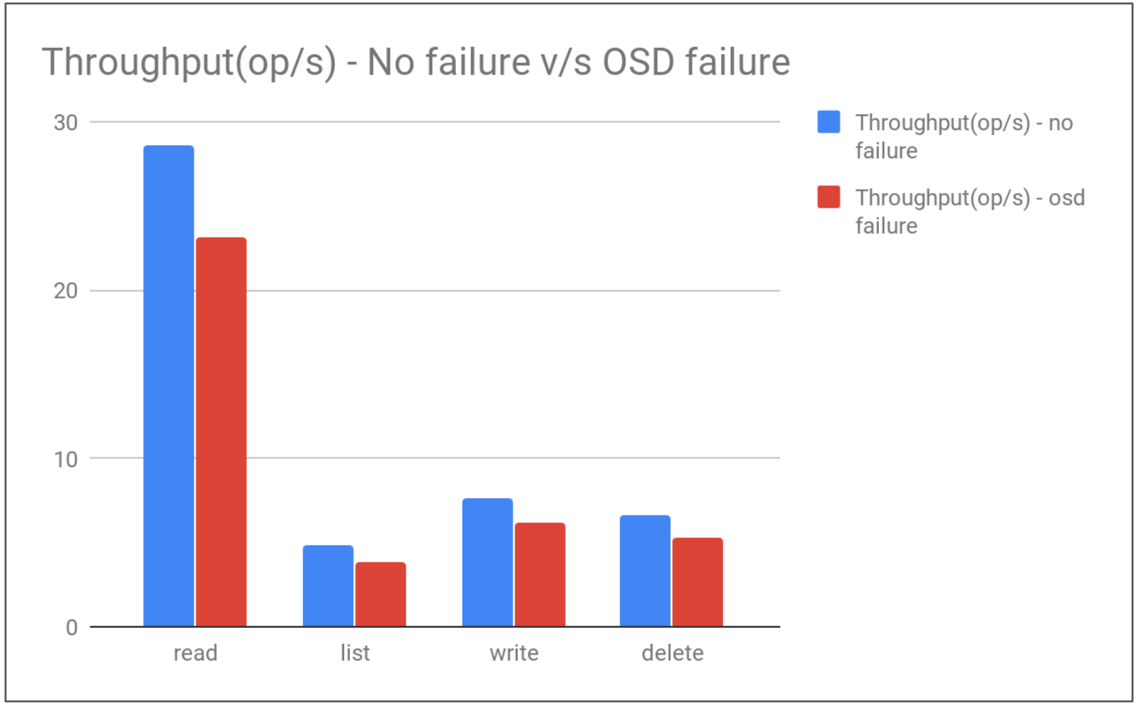
RGW Workload generated using Cosbench.

Operations - read, list, write, delete

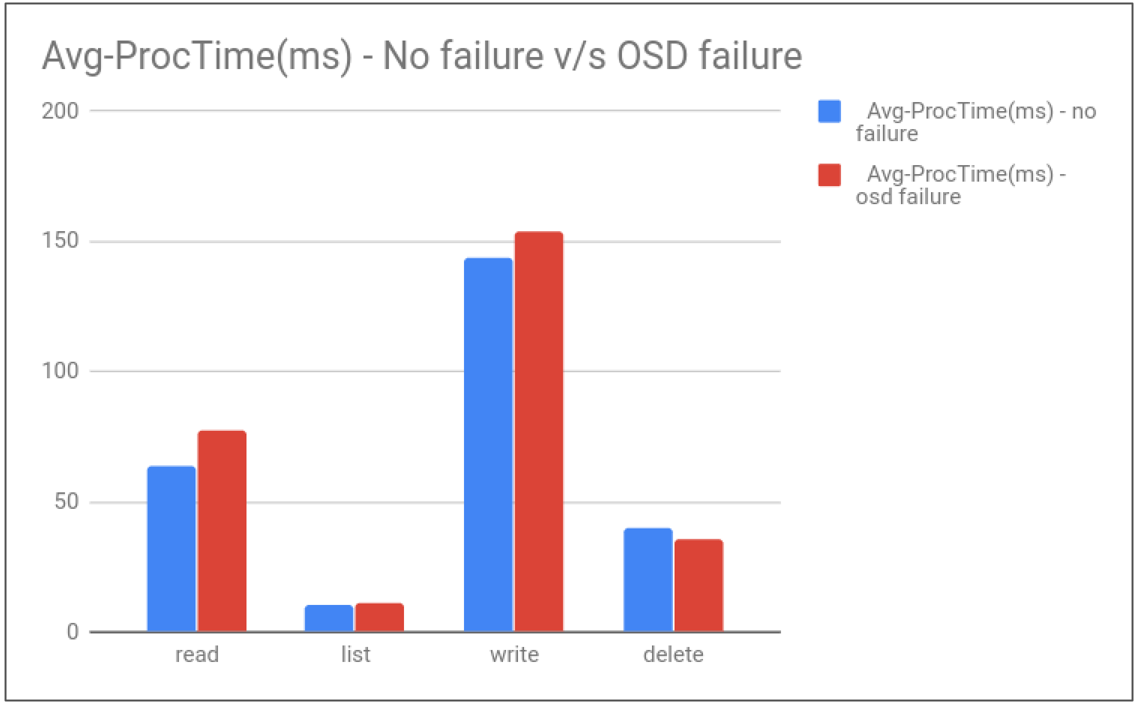
Cluster prefilled ~ 30%, 40000 objects

Kill one OSD to induce recovery

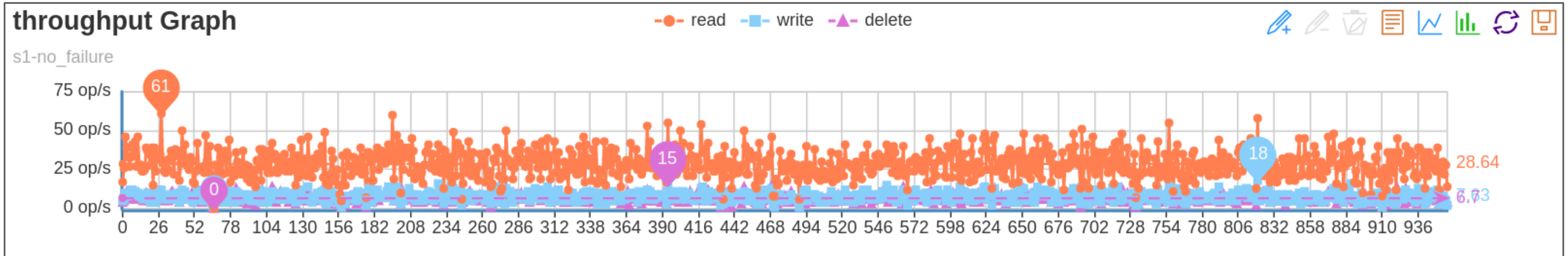
Impact on Throughput during Recovery



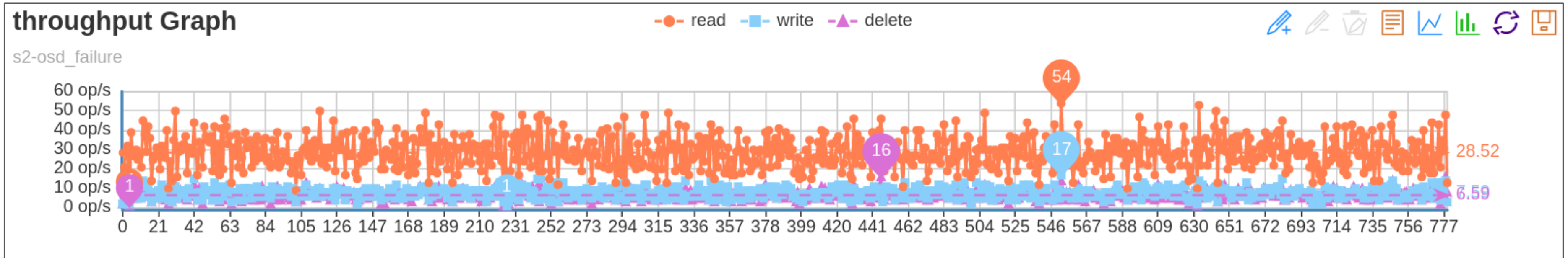
Impact on Latency during Recovery



Throughput Comparison - COSBench



No Failure



OSD Failure

Future Work

- Optimizations
  - Partial object recovery
  - Speed up backfill scanning and transfer
- Adaptive recovery throttling - set the value of osd_recovery_sleep based on client load.
- QoS
- Recovery Order

Summary

- Upgrade! Much better defaults and finer tuning in Luminous and Mimic
- Recovery sleep is all you need if you want to change client i/o vs recovery balance
- Tuning older options (e.g. max active, single start, max backfill) only needed if you want to increase recovery/backfill throughput

THANK YOU

Josh Durgin : jdurgin@redhat.com
Neha Ojha : nojha@redhat.com