Erasure Coding Enhancements for Tentacle

Bill Scales
Alex Ainscow

Vision

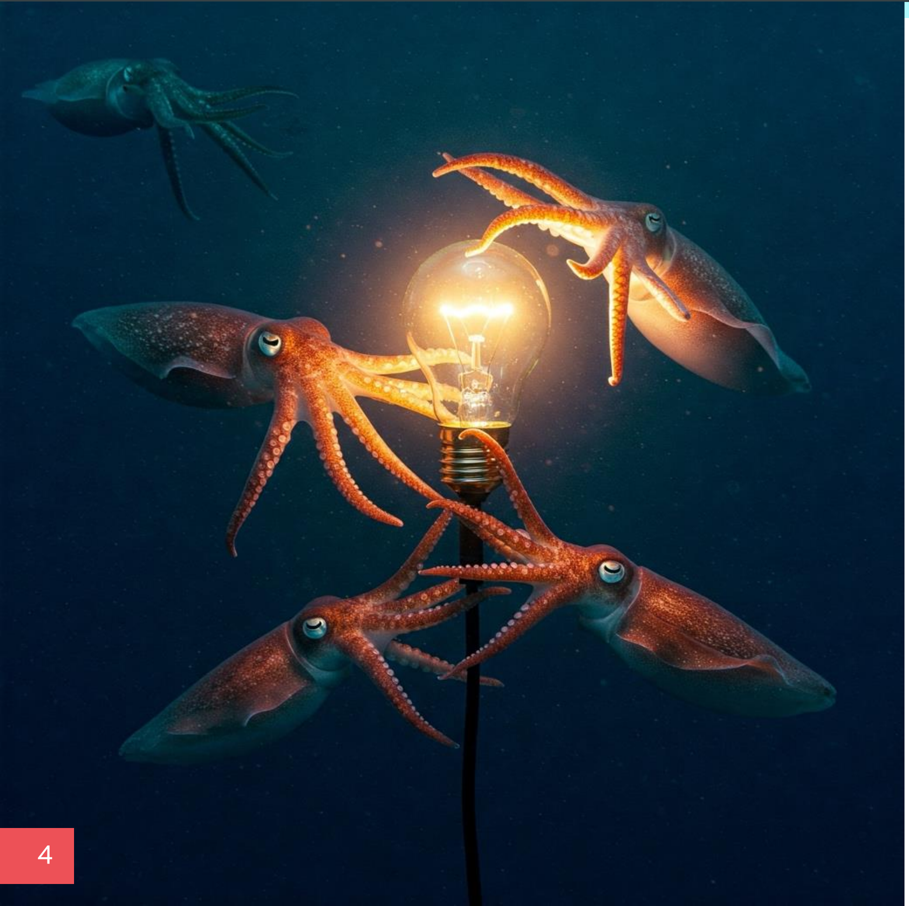
Optimize I/O Performance for Erasure Coded Pools

- Enhance performance to be similar to Replicated Pools...
  - But with lower Total Cost of Ownership (TCO)
- Make Erasure Coded pools viable for use with block and file

Contents

- How to turn on new EC
- Recommended profiles and configurations
- Implemented features from proposal
- Extra features
- Performance review

Enabling “Optimised” EC



- By default, all optimisations will be turned off
- Optimisations can be turned on for each pool
- A configuration flag can be changed to create new pools with optimisations on
- Profiles cannot be changed on enabling
- OPTIMISATIONS CANNOT BE SWITCHED OFF!
- All OSDs, MONs and MGR must be upgraded to Tentacle or later
- Backward compatible with old clients

Configuration options...

- Enable optimizations for a pool:

```
ceph osd pool set <pool_name> allow_ec_optimizations true
```

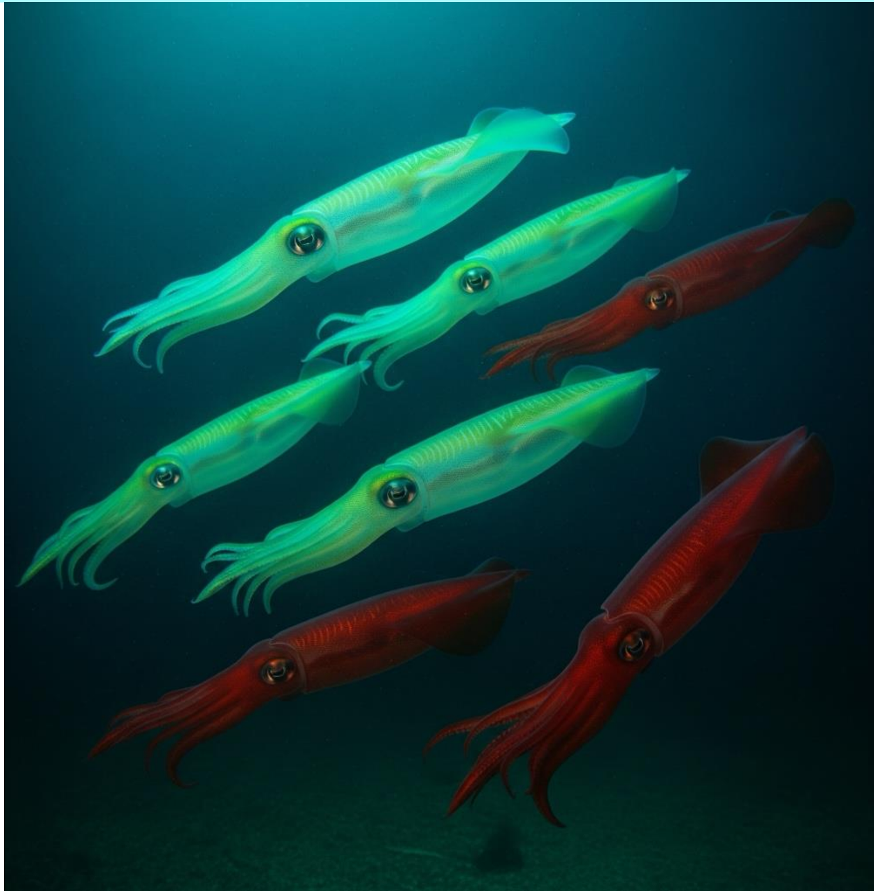
- Enable optimizations by default for new pools in [mon] (or global) in ceph.conf:

```
[mon]
      osd_pool_default_flag_ec_optimizations = true
```

Feature Status

Feature status – Previously planned

- For details see Ceph London 2024: https://youtu.be/bwcntmYP7ic
- Partial Reads
- Partial Writes
  - NOTE: Partial metadata – unwritten shards have no processing
- Parity Delta Writes
  - Per-IO auto-switch between write methods
- Larger default chunk size
- Direct Read
- Direct Write
- Parity Delta Writes

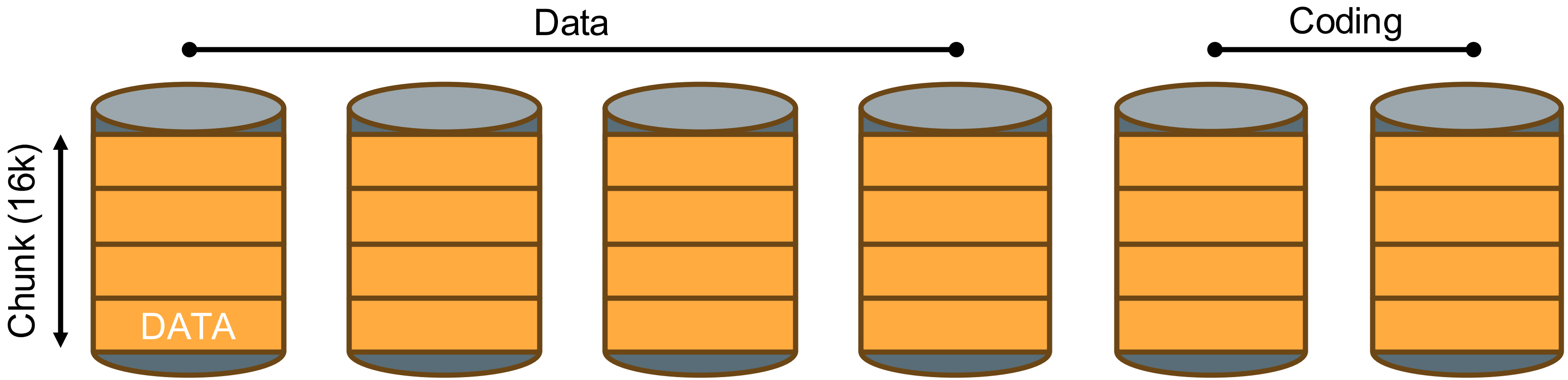


NEW: Space efficient small objects: Legacy

- Case study. 16k Chunk Size, 4+2 EC
- LEGACY: 4k Object will consume 96k:

🟧 Written

□ Unwritten

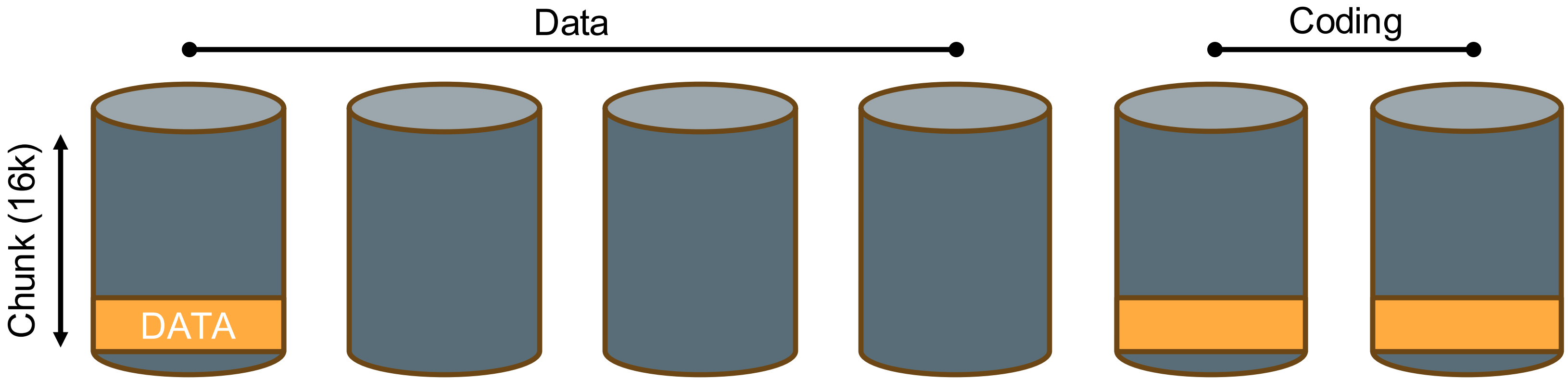


Space efficient small objects: Optimised

- Case study. 16k Chunk Size, 4+2 EC
- Optimised: 4k Object will consume 12k:

🟧 Written

□ Unwritten

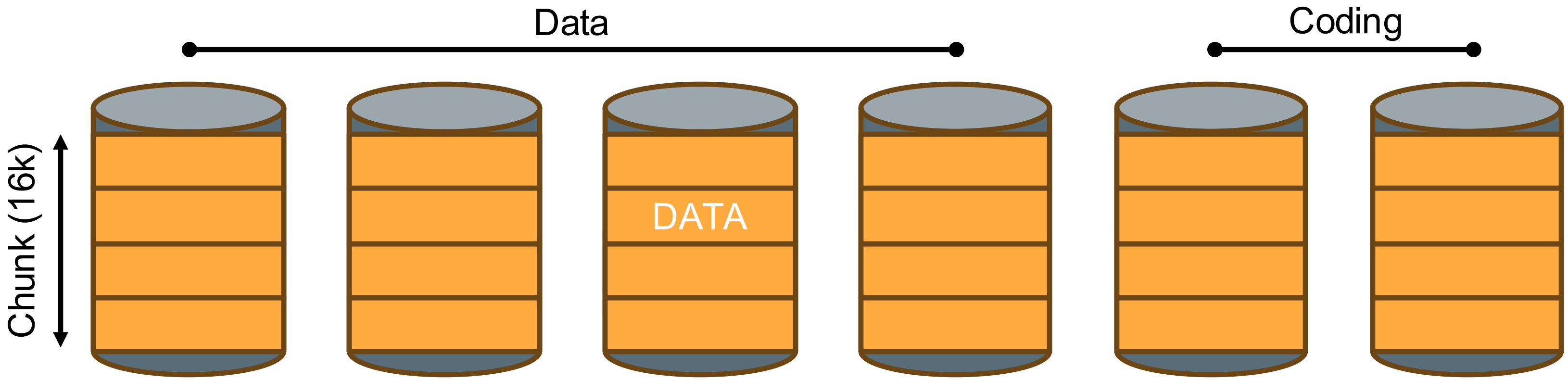


Space efficient sparse writes: Legacy

- Case study. 16k Chunk Size, 4+2 EC
- Object is >64k, write is to offset 40k
- 4k will consume 96k of space

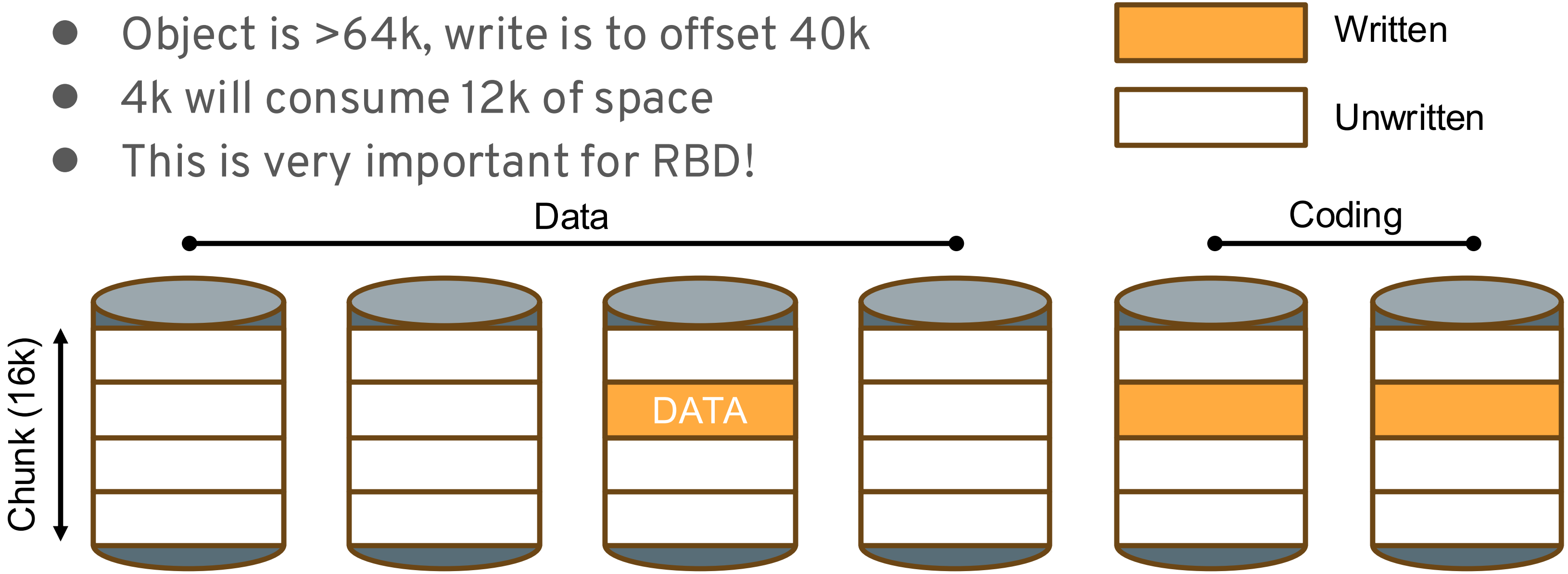
🟧 Written

□ Unwritten

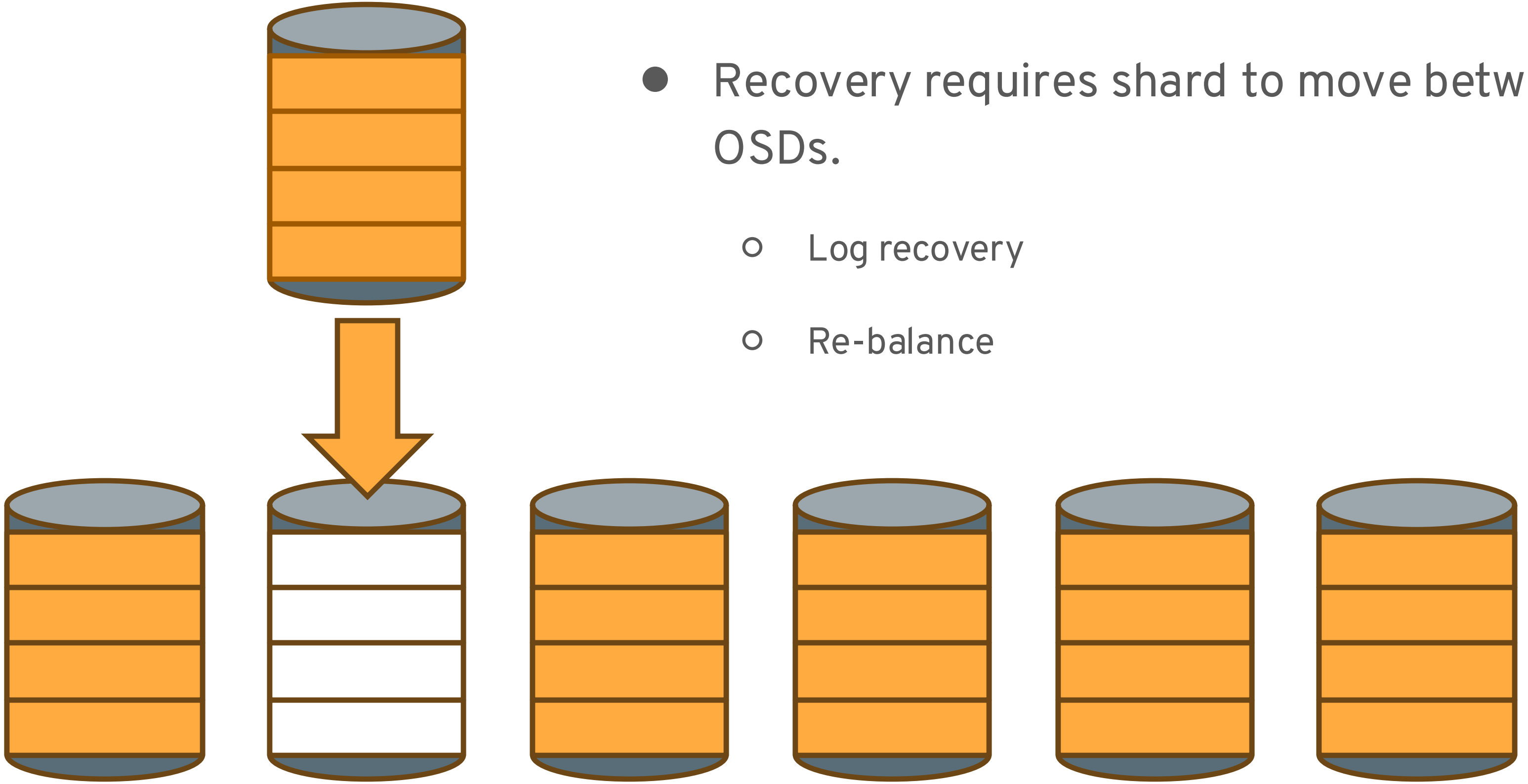


Space efficient sparse writes: Optimised

- Case study. 16k Chunk Size, 4+2 EC
- Object is >64k, write is to offset 40k
- 4k will consume 12k of space
- This is very important for RBD!

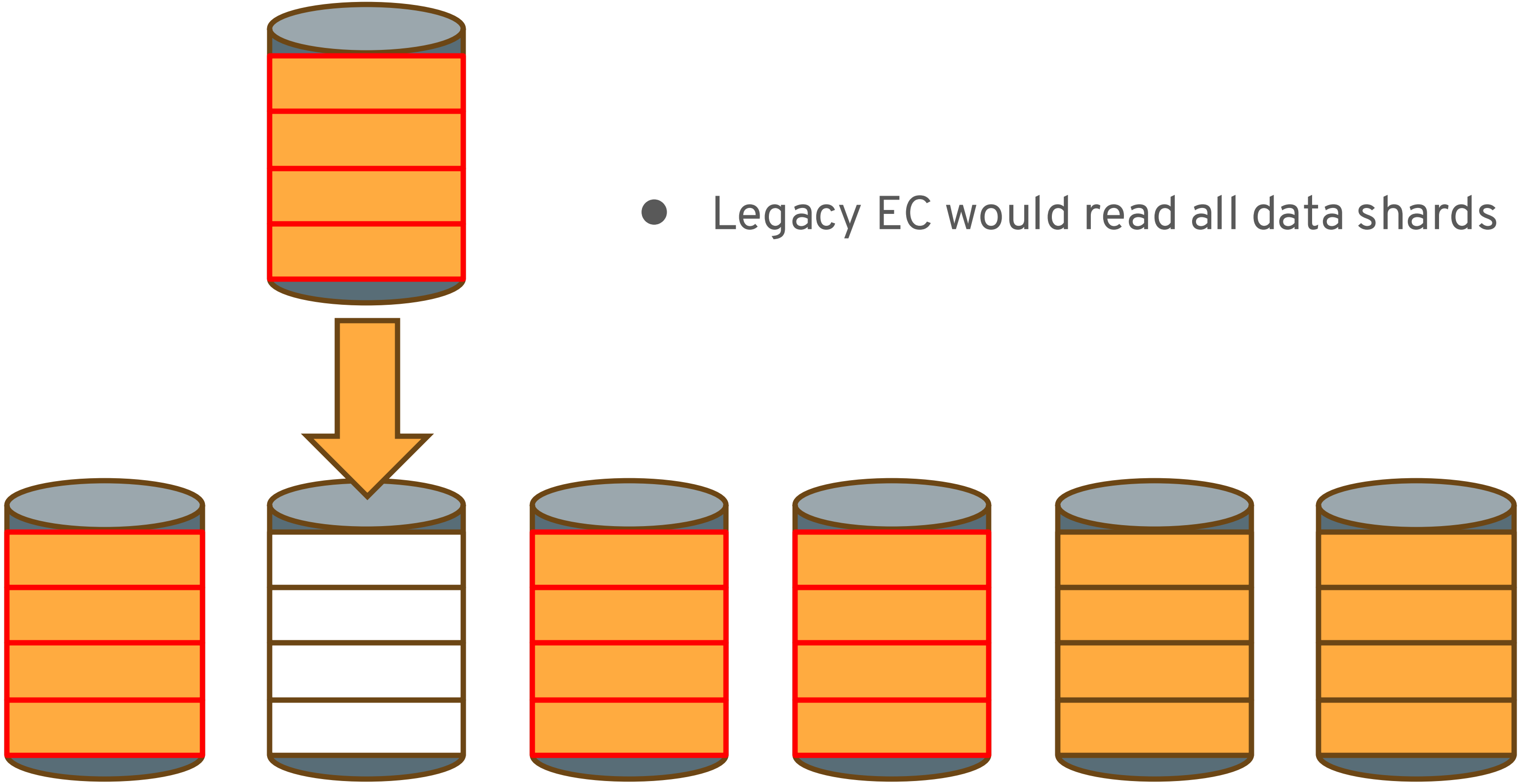


Re-balance improvements



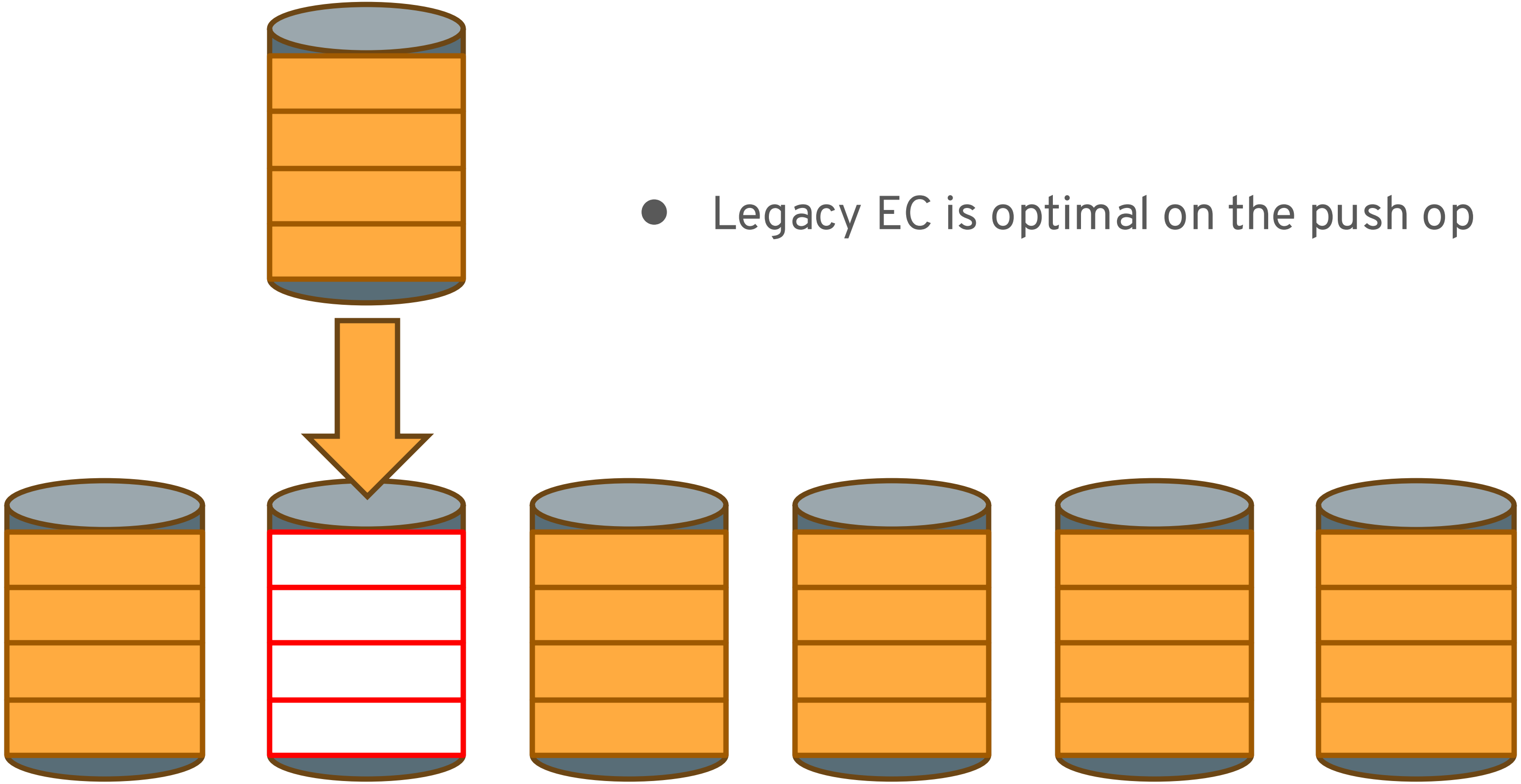
- Recovery requires shard to move between OSDs.
    - Log recovery
    - Re-balance

Re-balance improvements - Legacy



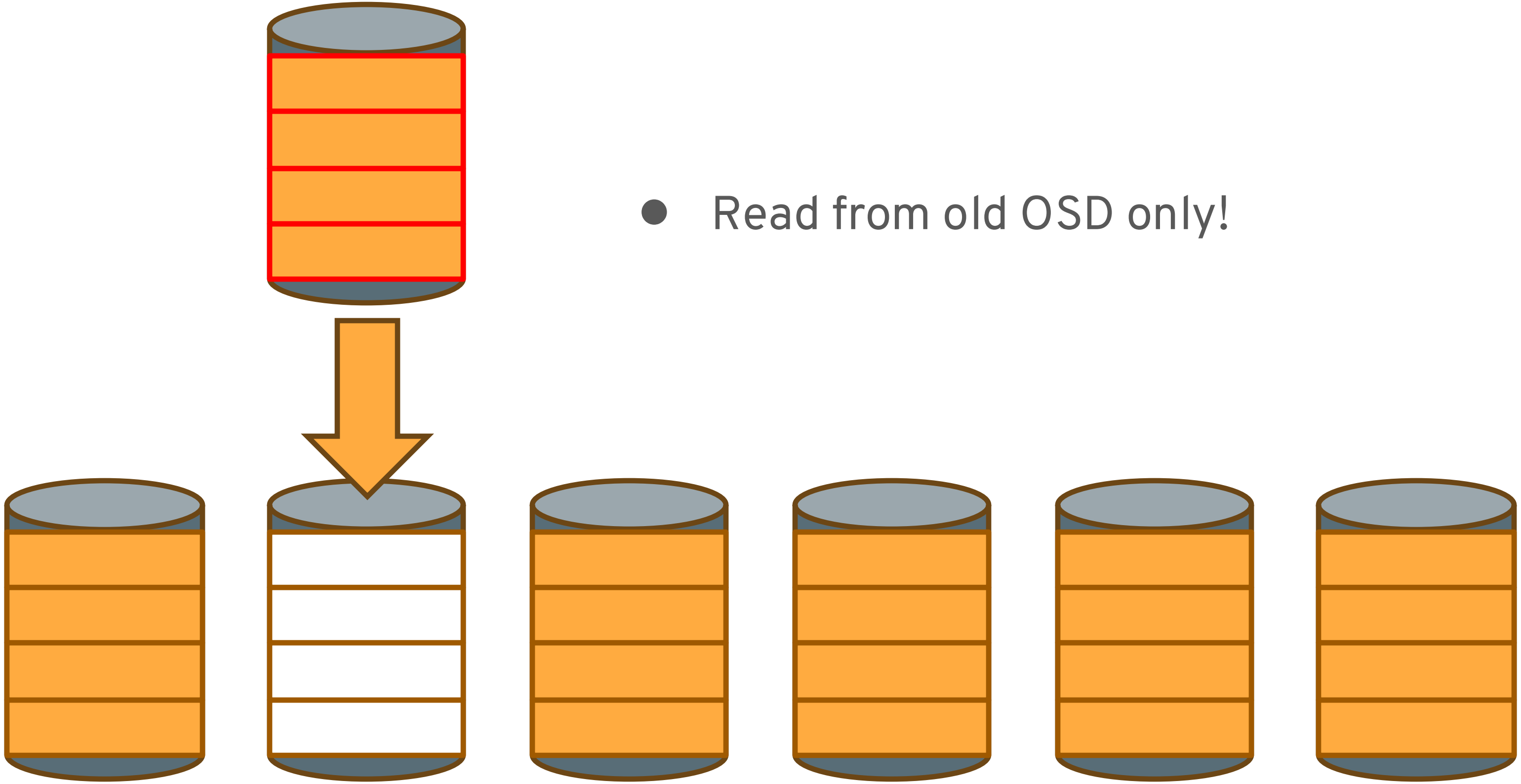
- Legacy EC would read all data shards

Re-balance improvements - Legacy



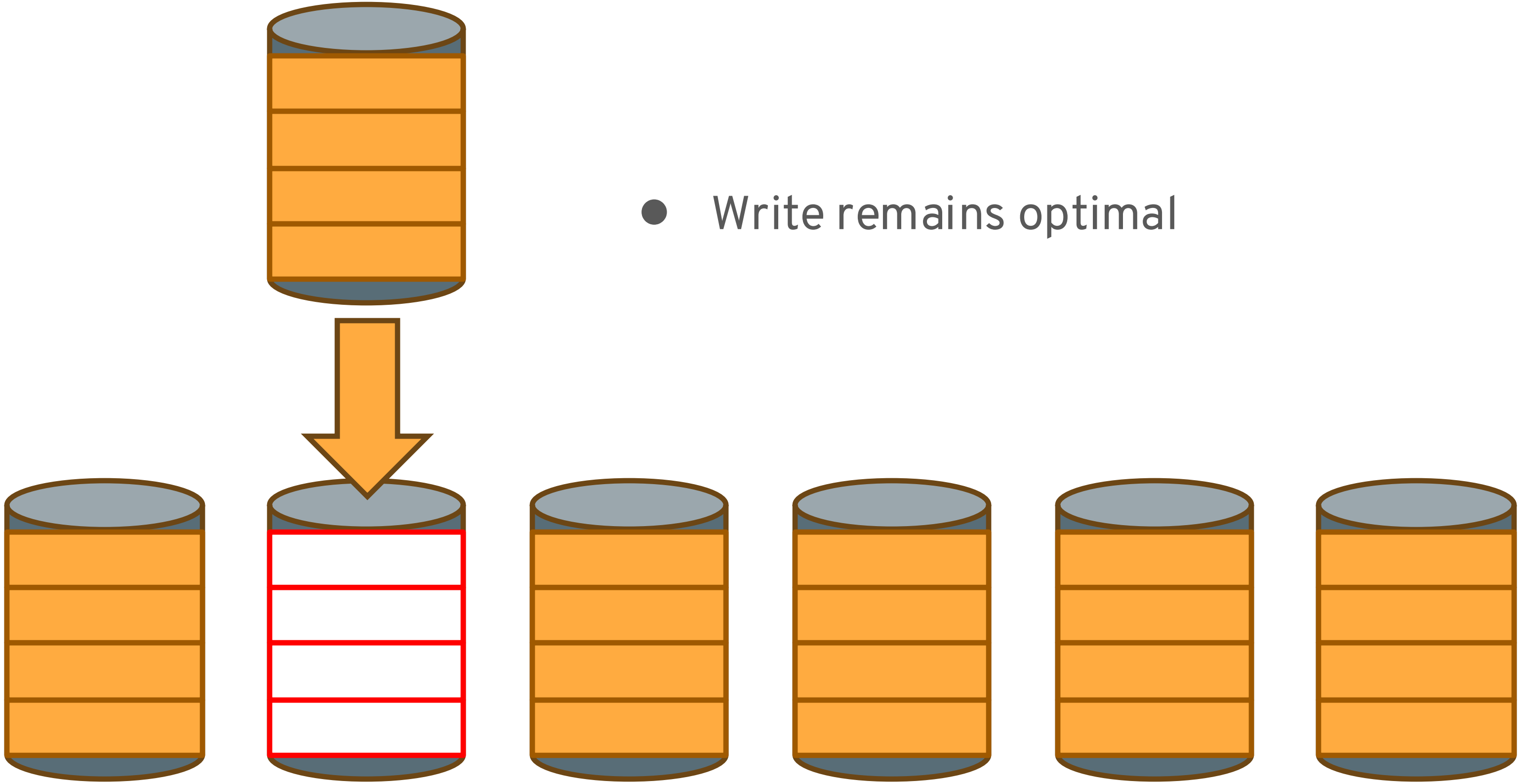
- Legacy EC is optimal on the push op

Re-balance improvements - Optimised



- Read from old OSD only!

Re-balance improvements - Optimised



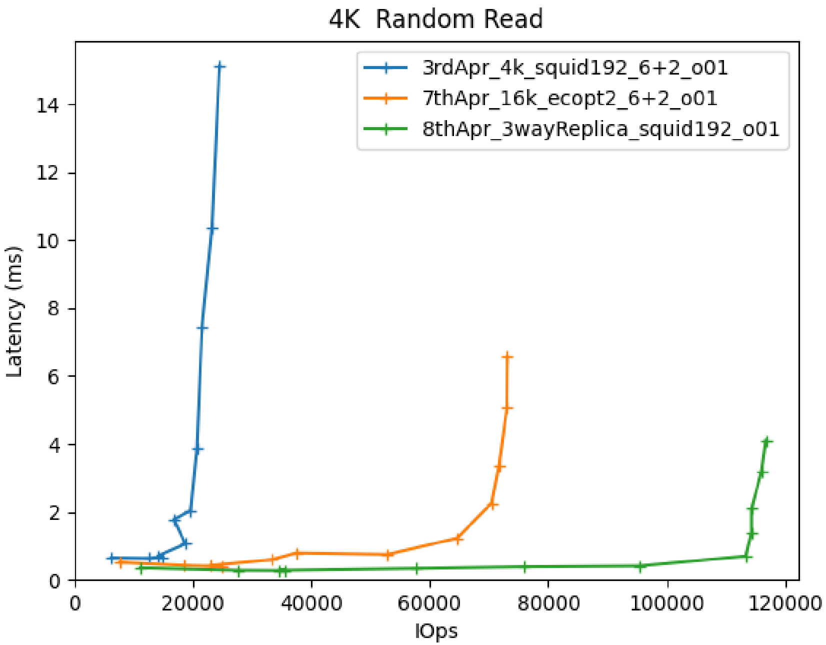
- Write remains optimal

Deep Scrub enhancements

- Using properties of CRC mathematics and common EC techniques, calculate coding CRC from data CRC.
- If a single corrupt shard, it will be identified (for m > 1)
- Shard CRCs do not need to be stored in metadata.
- New metadata checks for “partial” shards
- Thanks to Jon Bailey!

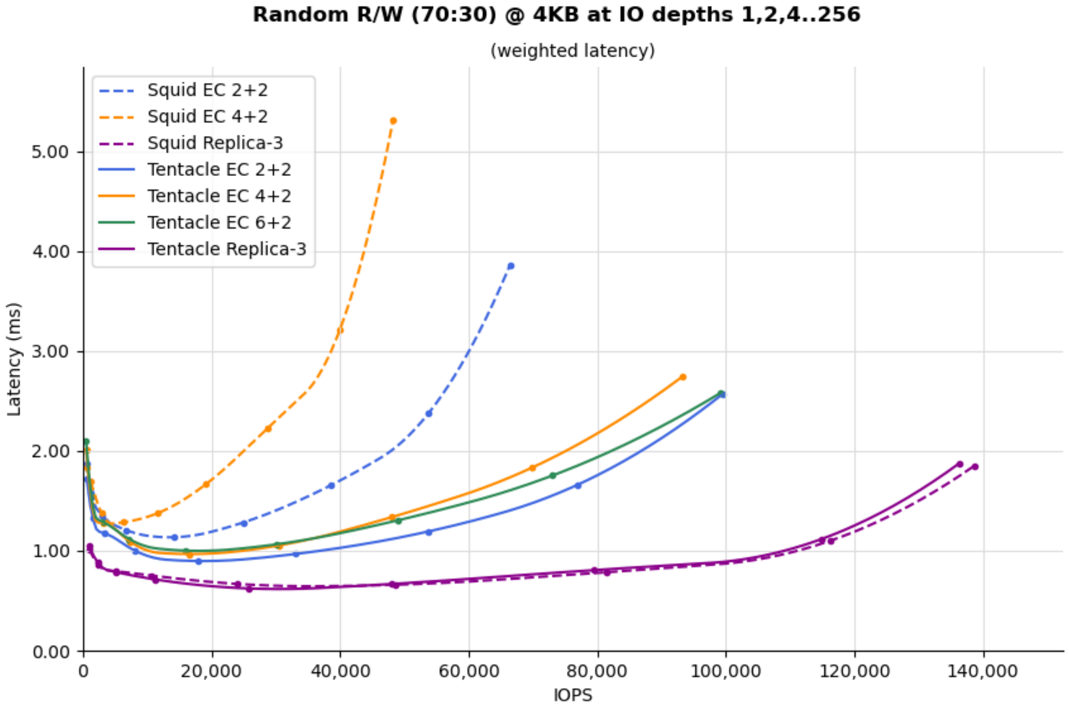
Performance

Read performance



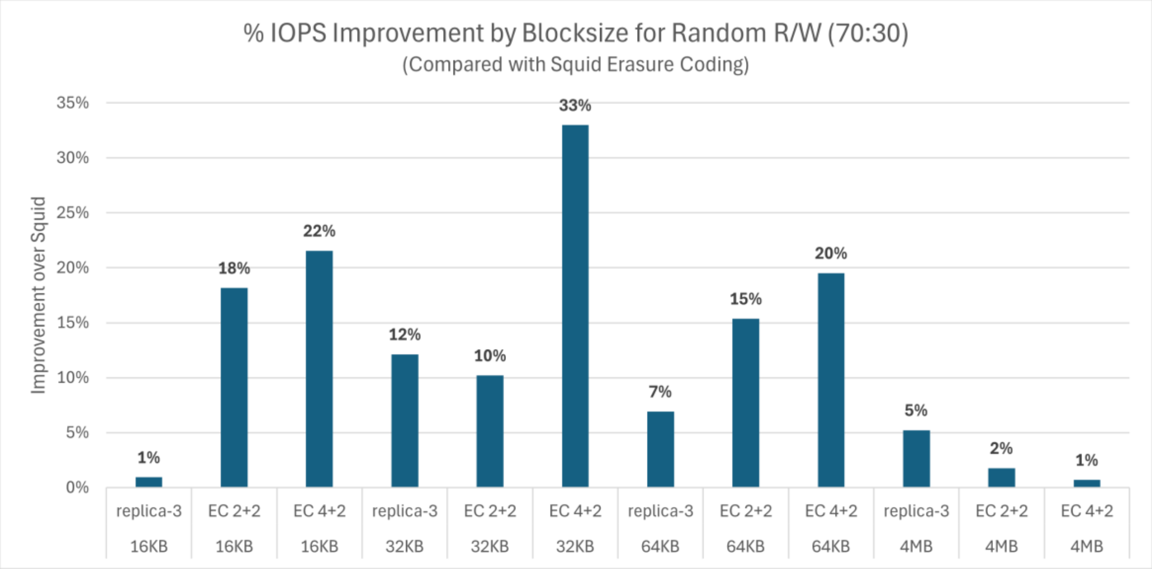
- 3-4x improvement in 4k random read performance
- Still not matching 3-way replica, but still compelling

Performance 1- 4k workload



- Chunk size is set to 16k
- Value of k has minimal impact on performance
- Latency is still higher than replica
- Lack of write cache means we cannot hide latency of Read-Modify-Write
- Direct-reads may improve latency and throughput

Performance summary Random R/W (70:30)



- All workloads show a performance improvement
- Very large IO (Object) workloads so a very small increase

Any questions?