



WELCOME

Sayafidine Said

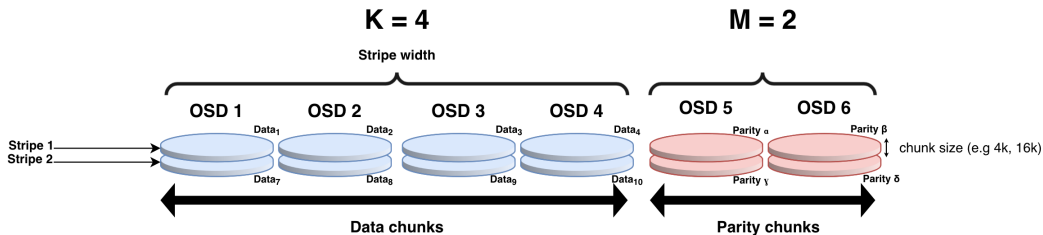
14th October 2026, Zurich

Performance of FastEC Optimizations for RGW

Evaluating the performance of Fast Erasure Coding optimizations for Ceph in RADOS
Gateway workloads

Sayafine Said (CERN, sayafine.said@alumni.cern)

Erasure Coding in short



Historical EC Optimizations

- Since Luminous (v12), only `fast_read`
 - Read requests sent to all chunks (data + parity)
 - Once K chunks are retrieved from OSDs, the data is reconstructed immediately and returned to the client
 - Helps with tail latency at extra IO cost

Recent EC Optimizations (FastEC)

- Developers talk at *Cephalocon25*
- FastEC introduced in Tentacle (v20)
 - **Partial Reads:** Read minimal data to serve client request instead of reading the whole stripe (save read IO and bandwidth)
 - **Padding removed for small objects:** Previously needed when the object was smaller than the stripe (save IO and space)
 - **Partial Writes:** On object updates, only read data chunks not being written, encode new parities, and write only modified chunks (save read+write IO and bandwidth)
 - **Parity Delta Writes:** On object updates, read data chunk and XOR new data, then apply delta to parity chunk (save read+write IO)

Performance Evaluation

Testbed Setup

- Hardware:
 - 30 nodes with 4x 3.84TB NVMe (block.db + WAL) + 30x 22TB HDDs (data), 32 physical cores, 384GB, 100Gbps Ethernet
 - Total: 900HDDs, 19.8 PB raw capacity
- Software:
 - Ceph Tentacle (20.2.1-575-g67b62888) with cephadm
 - 3 MONs, 3 MGRs, 900 OSDs, 60 RGW instances (2 per node)
- Benchmarking Tools:
 - *elbencho* v3.0.1
 - *warp* v1.5.0
 - 30 multithreaded clients on OSD nodes (no standalone client hardware)

Ceph Tuning

- mclock scheduler with balanced profile
- Several features disabled:
 - `osd_memory_target_autotune`, `bluestore_cache_autotune`
 - PG autoscaler, `rocksdb_cf_compact_on_deletion`
 - During tests: Scrubbing (shallow + deep) and Balancer disabled

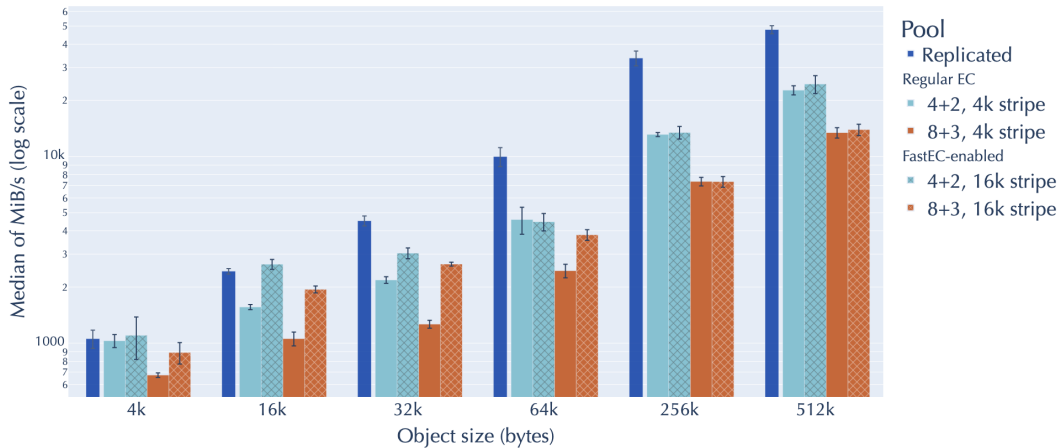
Pools Configuration

- ISA-L EC plugin
- Code rates: 4+2, 8+3
- Stripe units: 4k, 16k
- With and without FastEC optimizations
- One pool at a time, with optimal PGs/OSD ratio

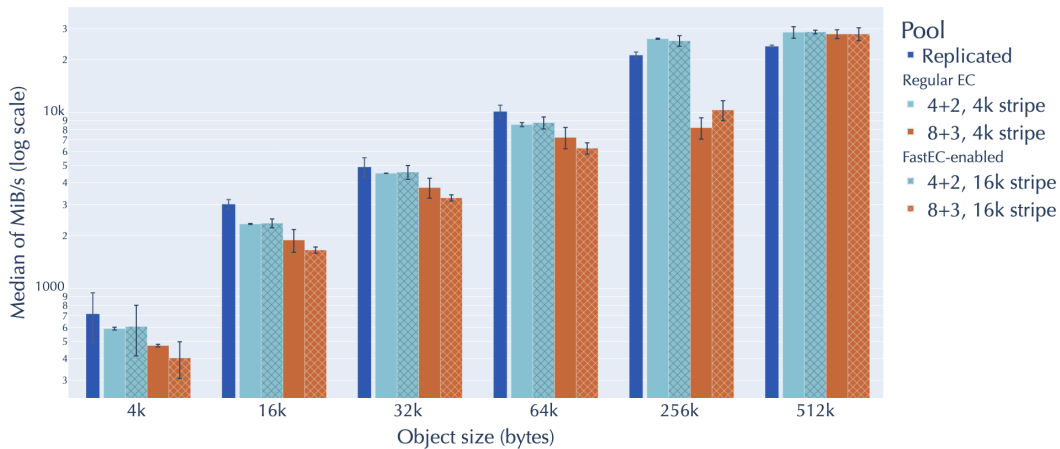
Workloads

- Evaluated several workloads, including:
 - Read
 - Write
 - Mixed read/write/delete
- Object sizes ranging from 1 kB to 16 MB
- 5 runs for each combination of code rate, stripe unit, and object size
- Presenting most relevant results for read and write workloads

Read, Median MiB/s per Object Size



Write, Median MiB/s per Object Size



Conclusion

- FastEC optimizations are beneficial for read-heavy workloads, especially for small objects and higher code rates
 - Noticeable improvements (~10%-50%) for smaller object sizes (<64kB)
 - Bigger chunk size (16k) and higher code rates (8+3) showed more pronounced benefits
 - Mostly driven by small objects benefiting from optimizations (no padding, partial reads)
- Surprisingly worse performance with some write workloads
 - RGW workloads are always entire object writes
 - FastEC optimizations would be primarily beneficial for object updates (allow_ec_overwrites)

Thank You for Your Attention!

- Questions/comments?
- Explore the *dataset*, a full report and a blog post will be available soon!

We are hiring!

<https://careers.cern/jobs/it-sd-gss-2026-96-ld/>

