ceph days
LONDON 2026

RGW Scalable Full Object Deduplication

Gabriel BenHanokh | Ceph Days 2026
gbenhano@ibm.com

ceph days LONDON 2026

Agenda

- Motivation - Why deduplication matters for object storage
- Features - What the system provides today
- Server-Side Copy Foundation - How RGW already shares data
- Architecture - How dedup works under the hood
- Full-Object vs Chunk-Level - Design rationale
- Roadmap - What's coming next and call for community

ceph days LONDON 2026

The Redundancy Problem

- Object storage clusters inevitably accumulate redundant data:
  - Backup/DR pipelines writing same objects across multiple buckets
  - CI/CD artifacts duplicated across namespaces
  - Data lake ingestion producing identical copies under different keys
  - Versioned buckets retaining identical versions
  - Cross-tenant duplication in multi-tenant clusters
- This redundancy translates directly to:
  - Unnecessary RADOS pool capacity consumption
  - Increased recovery and rebalance overhead
  - Higher operational cost per useful byte stored

Feature Overview

- Full support for:
  - Object versioning
  - Multi-tenants
  - Compressed objects (coming soon)
- Not Supported:
  - Dedup across zones/pools/storage classes
  - Encrypted objects (TBD)
- Two operating modes:
  - Estimate: safe read-only scan, reports dedup ratio
  - Exec: performs actual deduplication

ceph days LONDON 2026

Operations and Control

- CLI: radosgw-admin dedup <command>
  - estimate | exec | pause | resume | abort | stats | throttle
  - Full lifecycle: start -> pause -> resume -> abort
  - Stats tracks progress with a per-shard heartbeat
- REST API: /{admin}/dedup?op=<command>
  - Same commands available over HTTP (requires 'dedup' capability)
- Runtime throttling (adjustable while running):
  - Bucket-index reads per second (each read = 1000 entries)
  - Metadata access operations per second
  - Object Read operations per-second (TBD)

ceph days LONDON 2026

Filtering

- Bucket filtering (allow/deny lists from file):
  - Process only specific buckets, or exclude specific buckets
  - E.g. skip very active buckets, skip corrupted buckets …
- Storage class filtering (allow/deny lists):
  - Process only specific storage classes, or exclude specific storage classes
  - E.g. skip Glacier storage class

ceph days
LONDON 2026

Server-Side Copy Foundation

ceph days LONDON 2026

Sharing Tail Object

- RGW object layout:
  - Head object: Holds the S3 identity (name, attributes..) and can't be shared
  - Tail objects: Plain data – can be shared (ref_count)
- Server-Side Copy (existing mechanism):
  - Increments ref-count on source tail objects (no data copy!)
  - Copies the head object (attributes + manifest from source)
  - Target manifest points to the SAME tail objects as source
- Data inlining in Head-Object
  - RGW optimization stores the first 4MB of the obj data in the head object
  - It can't be shared because the Head-Objects holds the S3 identity

ceph days LONDON 2026

From Server-Side Copy to Dedup

- Dedup builds directly on server-side copy infrastructure:
  - Same ref-counting code for tail objects
  - Same atomic manifest-rewrite operation
  - Same rollback semantics (dec ref-count on failure)
- But dedup adds a critical extension - Split-Head:
  - Server-side copy always copies the head data - can't share it
  - Dedup moves head data into a NEW tail object
  - Head becomes empty (attributes only)
  - New tail can now be shared via ref-counting

ceph days LONDON 2026

From Server-Side Copy to Dedup

- Dedup builds directly on server-side copy infrastructure:
  - Same ref-counting code for tail objects
  - Same atomic manifest-rewrite operation
  - Same rollback semantics (dec ref-count on failure)
- But dedup adds a critical extension - Split-Head:
  - Server-side copy always copies the head data - can't share it
  - Dedup moves head data into a NEW tail object
  - Head becomes empty (attributes only)
  - New tail can now be shared via ref-counting

ceph days
LONDON 2026

Architecture

ceph days LONDON 2026

Double Sharding – BI sharding

- Ingress sharding mapped to native BI sharding
- Create 255 Worker-Shards Tokens
  - Scale-up to 255 RGW worker servers
- RGW Servers compete over tokens using compare-and-swap
- Process all BI shards modulo worker-id
  - BI can have up-to 1999 shards and worker-id 255
  - Worker-id 17 will process BI shard [17, 17+255, 17 +255*2 …]
- When finished, grabs the next token and so on …

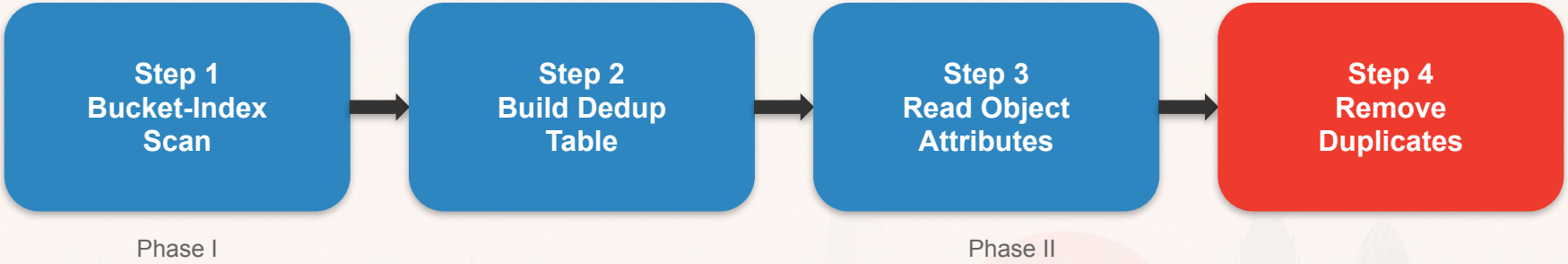
ceph days LONDON 2026

Double Sharding – MD5 space sharding

- Egress sharding by MD5 – shard_count maxxed out at 512 (2048)
- Group objects with the same (MD5 % Shard_Count)
- Ensures dedup candidates land in same hash table
- Each MD5 shard processed independently
- Benefits:
  - Robust: no need for external coordination
  - Load-balanced and Scale-out: small shards distribute work evenly
  - Memory-efficient: only one shard in memory at a time

ceph days LONDON 2026

The 4-Step Pipeline



ceph days LONDON 2026

Step-1: Bucket-Index Scan

- Unordered scan reading BI shards in native mode
- RGW members split BI bucket-shards between all members
  - **Linear scaling**
- More efficient and scalable than ordered bucket-listing
- Read entries in bulk (1000 per RADOS call)
- Extract: MD5, size, part-count, storage-class, instance
- Shard records by MD5 into on-disk SLABs
- ~50K objects/sec per RGW — a very fast pass

ceph days LONDON 2026

Step-2: Build Table

- Read SLABs sequentially into open-addressing hash table
- 32 bytes per entry (24B key + 8B value)
  - 6 Bytes On-disk-address (in the context of current MD5 shard)
  - 2 Bytes – state/count/flags…
- Count duplicates per unique [MD5, size, storage-class]
- ESTIMATE MODE STOPS HERE - reports dedup ratio

Step-3: Read Object Data and ATTRs

- Purge singleton entries from table
- For each candidate: Read RADOS object attributes
- Filter out: encrypted, compressed (if disabled), changed
- Compute Blake3 hash over object data
  - Reads the full Object Data
  - Decompress (if needed) and calculate Blake3
  - Only computed for candidate objects (not singletons)
- Output reduced SLABs with full metadata + hash

ceph days LONDON 2026

Why do need Blake3 and is it enough?

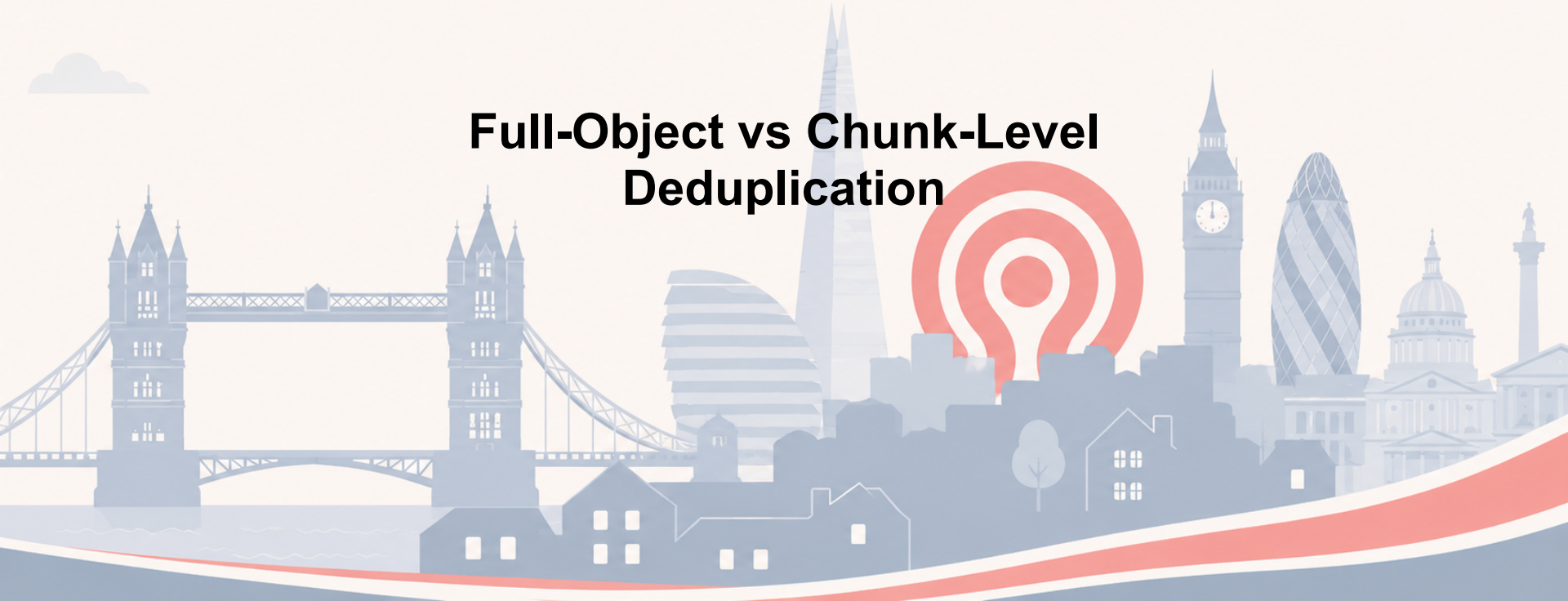
- Full Object dedup gets 128-bit MD5 (ETAG) for free from BI
- It is a very good predictor for objects-match, but not enough
- Not cryptographic safe:
  - Collisions can be produced by a malicious user
- Birthday Paradox MD5 with 2^50 objects:
  - Probability for a single collision is ~ 1/(2^29)
- Blake3 is a 256-bit strong cryptographic hash
- Birthday Paradox Blake3 with 2^50 objects:
  - Probability for a single collision is ~2^-157 — statistically impossible.

ceph days LONDON 2026

Step-4: Remove Duplicates

- Compare Blake3 hashes between SRC and TGT
- Increment ref-count on SRC tail objects
- Store Blake3 Hash on SRC (if not there yet) together with shared-manifest attribute
- Atomically: verify + overwrite TGT manifest with SRC manifest
- Delete old TGT tail objects
  - Inline delete to prevent GC pressure from building up
- Full rollback on any verification failure

ceph days
LONDON 2026



Full-Object vs Chunk-Level Deduplication

Why Not Chunk/Block-Level Dedup?

- Server-Side-Encryption is enabled by default on all on AWS
- Server-Side-Compression (SSC) is widely adopted
- This makes Chunk and Block Level Dedup ineffective
- Dedup must be done on clear uncompressed data which requires dedup to decrypt/decompress every object in the system for FP clac
- This is in stark contrast to Full-Object Dedup which first compares Etag to find dedup candidates and only decrypt/decompress objects which are very likely to be dedup

- Compression dedup granularity conflict
  - Either make compression blocks small (8KB-64KB) to match dedup granularity
    - This will degarde compression ration
  - Or make dedup blocks large (MB's) to match compression block granularity
    - This will degarde dedup ratio
- Keeping them both at native granularity
  - Will need to recompress object for every match without the remove dup block
  - The dedup block will need to be compressed alone as a small block (low ratio)
- Decryption/Decompress state must be kept per small dedup-blocks

The Inline Problem

- For chunk-level to work with encryption/compression:
  - Must dedup BEFORE encryption/compression (inline)
  - Requires fingerprint table of EVERY chunk in the system
- At scale this is impractical:
  - 8 PB user storage with ~8KB chunks = 1 trillion fingerprints
  - Memory: More than 32TB (256bit FP + management overhead)
  - Even with sharding the FP table adds enormous cost
- I/O Path Latency
  - Encryption/compression work slows down I/O path
  - In addition, it will need roundtrip to check FP Table

- Full Object dedup align natuarlly with SSE/SSC
- Etag is calcaulted before encryption / compression
- Only objects with matching ETAG will pay full decrypt/decompress
- No need to re-encrypt/re-compress
- Encryption/Compression attrs are shared once at object level
- PR 68965 (under review) adds compression dedup support
- Encryption support can be added using similar apprcoh
  - The tricky part is how to maintain keys of removed buckets

ceph days LONDON 2026

Release Status

- Ceph 9.0: Estimate mode only
- Ceph 9.2: Full dedup support from command line
- Main branch: Everything
  - Full dedup + compressed objects PR pending review
- Umbrella release: Lifecycle integration
  - Active service doing periodical dedup
  - Telemetry and dashboard, in addition to CLI/REST
- Seeking community involvement:
  - Testing at scale, edge cases, production policy feedback

ceph days
LONDON 2026

Thank You
Questions?