A Tour of Garbage Collection

Tobias Brunnwieser

Senior Infrastructure Engineer @ Hetzner Cloud GmbH

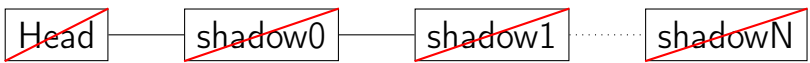
October 14th, 2026

Disclaimer

- I don't know about the history of garbage collection in RGW
- This talk is therefore only based on:
  - Reading the code
  - Own experiments
  - Operating and maintaining large Ceph clusters

Deleting objects may be costly (1)

Imagine deleting a *large* S3 object:



We'd need to:

- Update the bucket index
- Delete the head object
- Delete the tail (shadow) objects
  - up to 1.28k for objects uploaded via single PUT
  - up to 12.8M for objects uploaded via multipart

Deleting objects may be costly (2)

Imagine deleting many S3 objects *at once*:

```
POST /?delete HTTP/1.1
Host: Bucket.s3.amazonaws.com

<?xml version="1.0" encoding="UTF-8"?>
<Delete xmlns="http://s3.amazonaws.com/doc/2006-03-01/">
   <Object>...</Object>
</Delete>
```

- Up to 1000 at once (`rgw_delete_multi_obj_max_num`)
- *Single* request, possibly *billions* of RADOS objects!

Deleting objects may be costly (3)

Imagine cleaning up mulipart uploads:

We'd need to:

- Delete up to 1000 parts at once (`rgw_multipart_part_upload_limit`)
- Where each part can be made up of 1.28k RADOS objects

Therefore: Garbage Collection!

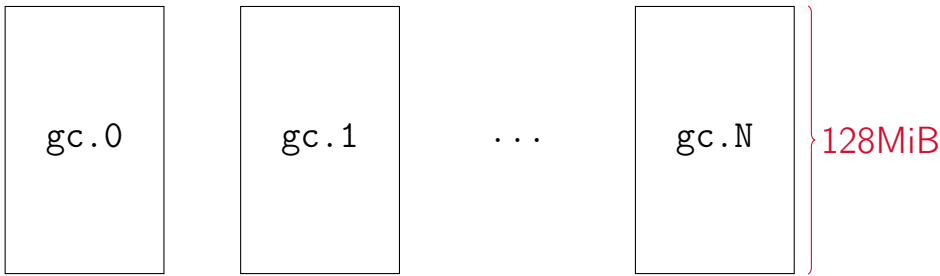
Idea:

- Place RADOS objects in queue(s) instead of deleting
- Let a background thread cleanup *later*
- With expiration delay (so we don't interrupt still running read operations)

Further benefits:

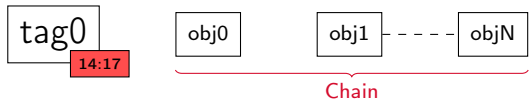
- May use low I/O priority
- Defer deletion to off-peak hours

GC Queue



- There are `rgw_gc_max_objs`[1] shards (default: 32)
- In log pool, namespace "gc" (e.g. `foo-cluster.rgw.log/gc`)
- Fixed maximum size `rgw_max_queue_size`

Queue Contents: Tags and Chains



- Each deletion is associated with a *tag*
  - tail / shadow: tail tag (alphanumeric)
  - multipart: original upload ID
- With the tag we store an expiration time (delete not before)
- And a list of RADOS objects that need deletion
- RADOS objects/keys associated with tag are called a *chain*

List Objects in GC Queue

We can list all entries in all shards using:

```
radosgw-admin gc list --include-all
```


```
[
    {
        "tag": "zg.463772309.12748744736315718323\u0000",
        "time": "2026-10-05T17:15:11.092258+0000",
        "objs": [
            {
                "pool": "foo-cluster.rgw.buckets.standard.data",
                "oid": "zg.463772721.91009__multipart_01M46F8XWV8A63A74DZHGN74XE.uploadid.1",
                "key": "",
                "instance": ""
            },
            {}
    }
]
```


Note: only if you pass `--include-all` you get non-expired tags!

GC Processing

1. Select GC queue and lock it
2. Obtain 100 entries (tags)
3. For each entry:
   - Skip tag if not yet expired
   - Otherwise: purge all RADOS objects
   - Throttled by by `rgw_gc_max_concurrent_io`[2] (default: 10)
4. Remove all completed tags from queue
5. Repeat until queue empty or `rgw_gc_processor_max_time`[3]

Deletion's That Aren't (Necessarily)

How does GC delete objects?

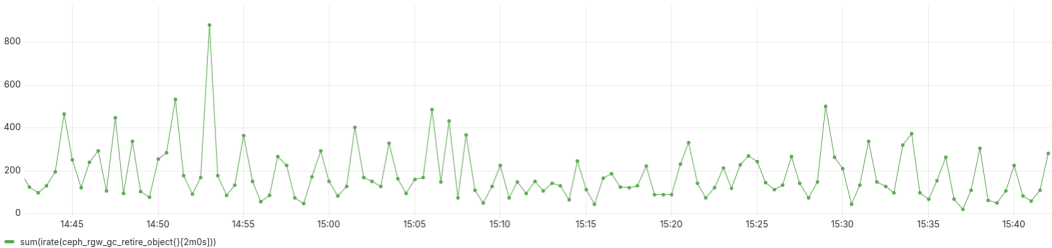
- GC never *directly* deletes a RADOS object
- Instead: reduce reference count (stored in xattr, fast)
- Only if count reaches zero, RADOS object is actually deleted

Why Reference Counting?

`CopyObject` allows fast cloning of S3 objects:

- Only store new head object (unique manifest, xattr)
- And increase reference count for each RADOS object on the tail

Monitoring: Prometheus Metrics



- Single metric only: counter `ceph_rgw_gc_retire_object`
- Represents: number of tags (complete chains) removed

Running GC Manually

Simply: `radosgw-admin gc process --include-all`

- Same processing as single GC thread, but in *foreground*
- Useful to purge objects *immediately* if combined with `--include-all`

Increasing Processing Throughput

Increase Maximum Concurrent OPs

- Increasing `rgw_gc_max_concurrent_io`[2] (default: 10)
- Does *not* increase IO priority!
- Beware of OP latency

Add more GC threads

- Every RGW with `rgw_enable_gc_threads` runs *one* thread
- Each RGW GC thread operates on a single shard at a time
- More GC RGWs process more shards concurrently

Add GC Queue Space

What if you want more "buffer"?

- Increasing GC shard size without interruptions seems impossible
- But you can add more shards (we did successfully):
  - Stop all RGWs running GC thread (or simply disable thread)
  - Increase `rgw_gc_max_objs`[1] (default: 32)
  - Re-enable threads / start RGWs
- Missing shards will automatically be created!

Questions

Questions?

What happens if GC queue runs full?

- No object is lost: immediate deletion
- But: possibly *massive* performance penalty!

References I

- Config option rgw_gc_max_objs
  https://docs.ceph.com/en/latest/radosgw/config-ref/#confval-rgw_gc_max_objs
- Config option rgw_gc_max_concurrent_io
  https://docs.ceph.com/en/latest/radosgw/config-ref/#confval-rgw_gc_max_concurrent_io
- Config option rgw_gc_processor_max_time
  https://docs.ceph.com/en/latest/radosgw/config-ref/#confval-rgw_gc_processor_max_time