RED HAT SUMMIT

# Optimizing Ceph Object Storage For Production in Multisite Clouds

Do’s and don’ts of multisite object storage

John Wilkins
Michael Hackett

May 9, 2018

## Objectives

- High level understanding of Ceph Object Storage
- How to plan for a Ceph Object Storage cluster at scale
- How to address important success criteria--Do’s and Don’ts
- Understand hardware price/performance tradeoffs
- How to get the results you want for the time and capital you invest

## About Ceph

Enterprise-class cloud storage

Ceph delivers object, block and file storage on one platform, delivering:

- Scalability from petabytes to exabytes
- High Availability--hardware failure is an expectation, not just an exception
  - **Data durability:** replication or erasure coding
  - **Data distribution:** Data is evenly and pseudo-randomly distributed
  - **Fault tolerance:** Ability to confine failures. No single point of failure.
  - **Resilience:** Automatic recovery from a degraded state.

Ceph is a very appealing cloud storage solution for OpenStack.

## Ceph Object Storage

What is Ceph Object Storage?

Ceph Object Gateway (RGW) provides an object storage service with:

- Well-known RESTful S3 and Swift APIs
- User Management, Tenants, Users, Usage and Quotas
- Multi-site and Failover Capabilities
- Ability to integrate with OpenStack, LDAP/AD
- Multiple Performance Profiles Simultaneously
    - Throughput-optimized
    - Capacity-optimized
    - I/O-optimized

## Ceph Object Storage

Why Ceph Object Storage?

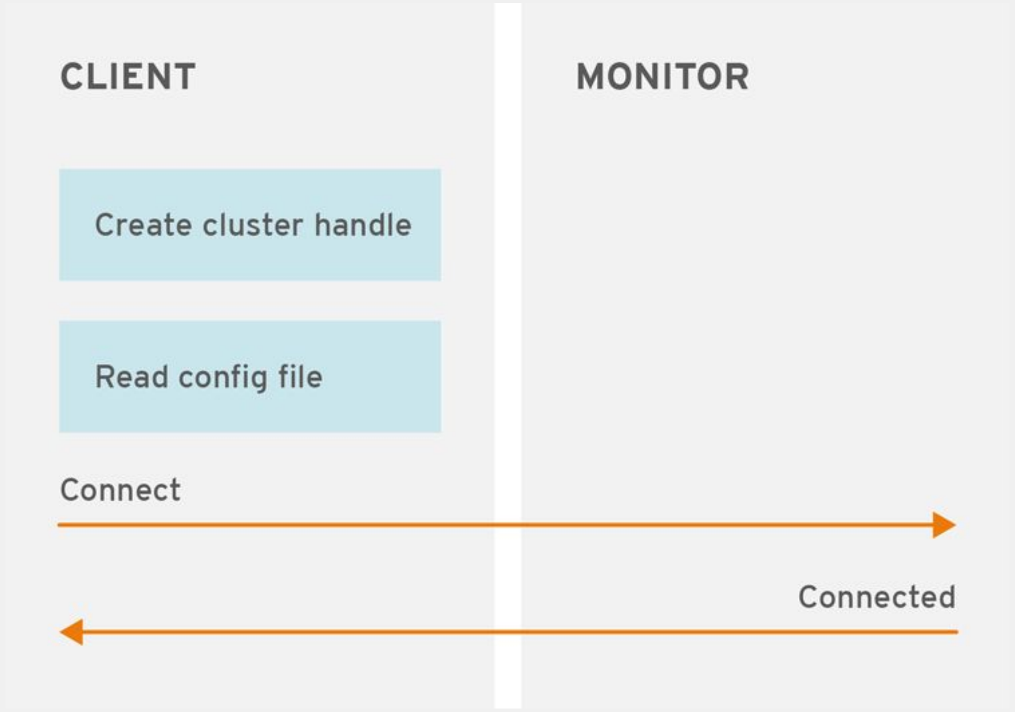
It’s an increasingly important cloud storage solution, because:

- It is ideally suited for unstructured data
  - Streaming Audio and Video, 4K
  - Graphics
  - Imaging
  - Analytics
- Over 60% of all data storage is unstructured and it is growing rapidly
- Emerging as a foundation for data lakes and analytics solutions
- The same skills required for other Ceph clients

## Understanding the Object Gateway and Storage Pools

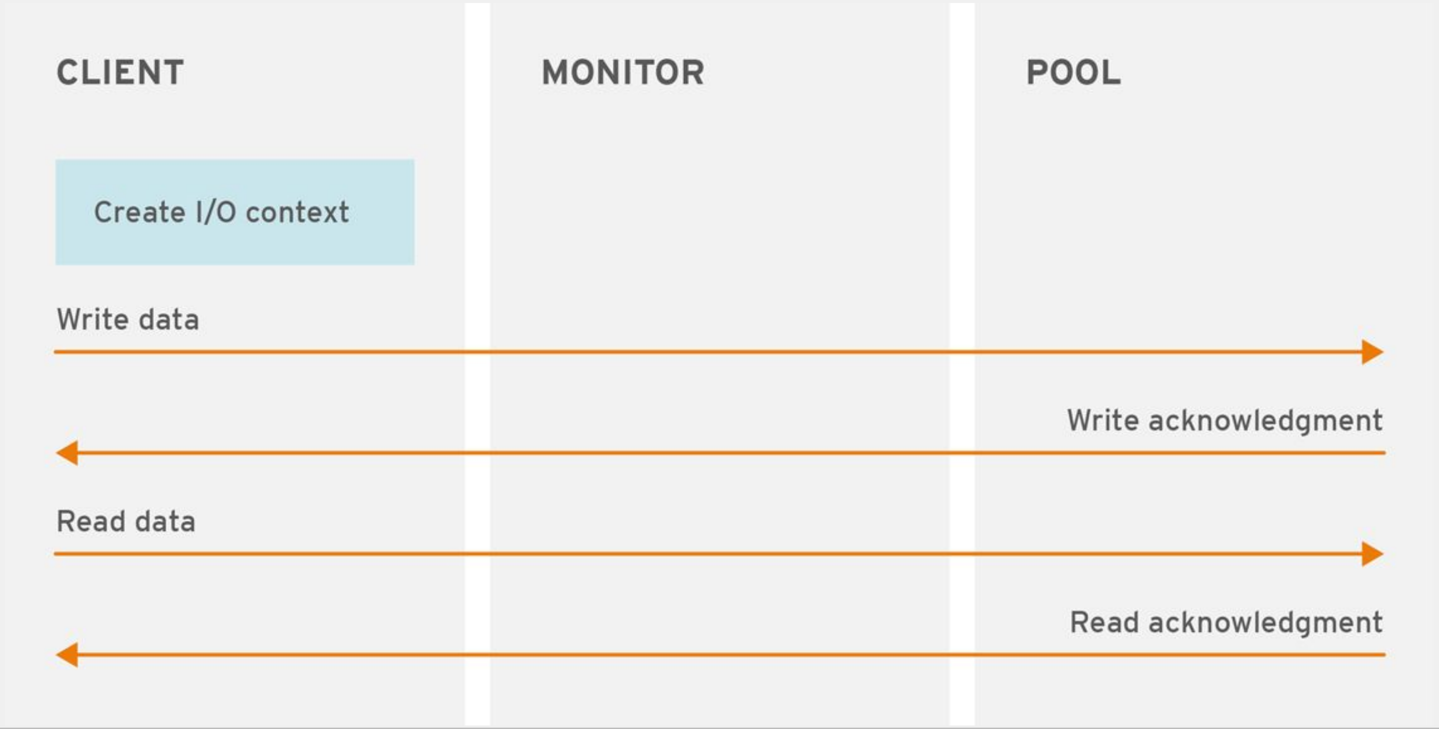
# All Ceph clients retrieve the cluster map

Ceph Object Gateway is a client, and so it retrieves a cluster map first



# Clients bind to a pool

Ceph Object Gateway binds to pools and reads and writes data



## Service Pools

There are 10 Pools just for the Service Layer

Service pools are completely separate from the pools that store client data.

- **Root:** Realm, zone group and zone configuration.
- **Control**
- **Garbage Collection**
- **Logs:** Logging and Intent Logging
- **User Management:** S3 UIDs, Swift UIDs, access keys, user emails, quotas/usage.

Service pools can use the same CRUSH hierarchy and ***must*** use replication for data durability. May use the same CRUSH hierarchy just for the service pools.

# Data Placement Pools

There are 3 Pools Per Placement Policy

Placement policies allow you to have different hardware configurations for different performance characteristics. There are three pools per placement policy.

- **Bucket Index:** Stores an index of objects per bucket. ***Use SSD/NVMe and Sharding!***
- **Bucket:** Stores the data objects of a bucket. ***Use erasure coding!***
- **Extras:** Stores other data, such as multipart uploads. ***Use SSD/NVMe!***

## Hardware Planning

# Hardware Defines Performance

...even though it is software-defined storage

Sometimes the term “software-defined storage” leads to misconceptions.

- “Commodity hardware” doesn’t mean old or low performance hardware
- <u>**Monitors**</u> are lightweight, but they <u>**are mission critical.**</u>
- The gateway supports multiple performance profiles--with different hardware
- Ceph (like all distributed systems) relies on networking.
- Saving money on hardware may increase expenses elsewhere

Hardware plays a significant role in achieving your objectives. **The RHCS Hardware Guide and the cited reference architectures are <u>*an invaluable source of information!*</u>**

## Hardware Planning

Identifying your general use case(s)

- Throughput-optimized (HDD/SSD)
- Capacity-optimized (HDD)
- IOPS-optimized (All Flash)

Identifying workload(s)

- Write (and delete), read intensive?
- What are the typical object sizes?
- What is a typical client load?

Estimating scale and growth requirements

- Data buckets use erasure coding. 1.5x
- Other buckets use replication. 3x
- Factor for multi-site requirements
- Consider compression too! RHCS 3+
- How fast is the cluster expected to grow?

Estimating storage density

- How many OSDs per host?
- Flash disks per host? Journal and Index
- Hot swappable drives
- Bandwidth requirements
- Gateway to OSD ratio? 1:50-1:100 OSDs

## Networking

Ceph High Availability depends on Network High Availability

Performance problems often begin with the network, not Ceph.

- High availability
  - Rack/leaf switches can be a single point of failure
  - A switch failure in a single rack cluster can bring down the cluster.
  - Nodes should always have ***at least*** two NICs
  - Nodes should use ***at least*** 10GB
  - Inter-rack bandwidth should be ***at least*** 40GB (or LCAP Mode 4 bonded 10GB)
- **Always** provision a cluster network to use more bandwidth than the public network
- Consider separate 1G networks for IPMI and management

## Networking

Additional considerations

Performance problems often begin with the network, not Ceph.

- Benchmark test your throughput. Bent cat-6 cables can reduce throughput.
- Ceph loves jumbo frames (MTU 9000)
- ***10Gbps is the bare minimum*** for a production cluster.
- Flash (SSD, NVMe, etc) easily saturates a 10Gbps network. Bottleneck!
- Consider serving public and cluster networks with the same rack switch (and make them redundant), so that a switch failure doesn't create a more significant problem for either the public or cluster network.
- Bandwidth between multiple sites must be sufficient to achieve timely consistency

## Storage Drives

Carefully Evaluate Storage Media

Storage drives play an important performance role, especially under heavy workloads

- HDD capacity is increasing faster than sustained transfer rates.
    - Generally faster RPMs provide higher sustained transfer
    - For higher capacity drives, require higher sustained transfer (300Mbps)
- HBA vs on-board controller. UPS required if enabling write back cache!
- Disable drive cache
- Prefer JBOD over RAID
- Identify the bottlenecks: Drive? Controller? Network?

## Storage Drives

Carefully Evaluate Storage Media

SSD drives play an important performance role

- For flash drives, always review the specifications and use enterprise-grade drives.
- Use SSDs for monitors to address synchronous write latency. Ensure enough capacity.
- Use SSDs for journals. Consider OSD to journal ratio.
- Use SSDs for bucket indexes
- Use SSDs for extras (multipart uploads)

When purchasing hosts, consider how many SSD drives you will need too

## Storage Density

Don’t be too dense!

Don’t assume everyone understands your high availability requirements.

- All drives ***should*** be hot-swappable.
- Dense storage may require you to pull operating drives to fix broken ones.
- Excessive density have other impacts
  - Dense storage servers require much more bandwidth. OSD:NIC ratio.,
  - Dense storage servers are a bigger point of failure.
  - Backfilling and recovery requires substantial bandwidth in the cluster network.
  - Erasure-coding requires more nodes, not less. K=8; m=4 == 12

# Hardware Planning

Monitor nodes

- 3 monitor nodes (minimum)
- Use SSDs
- Use uninterrupted power supply (UPS)

OSD Nodes

- ~12 OSDs per node
- Always use flash (SSD) for journals
- Always use flash (SSD) for bucket indexes
- Prefer hot-swappable OSDs
- Identical hardware in a CRUSH hierarchy

OSD Nodes (continued)

- 10 Gbps+ NIC for public network
- 25 Gbps+ NIC for cluster network
- 12 nodes minimum for best erasure code durability (K=8; m=4 requires 12 nodes).
- Replication should use 3 copies

RGW Nodes

- At least two per zone.
- 1 Gateway to 50-100 OSDs
- HAProxy/keepalived load balancing

## Kernel and Cluster Tuning

## Kernel Tuning

Tune the kernel for optimal performance

Software performance tuning begins with the kernel.

- Consider Ceph Ansible. Some kernel tuning gets done automatically!
- Adjust TCMalloc for heavy multi-threaded memory allocation workloads
  - Edit `/etc/sysconfig/ceph`
  - `TCMALLOC_MAX_TOTAL_THREAD_CACHE_BYTES = 128`
  - To free unused memory: `# ceph tell osd.* heap release`
- Adjust the ulimit for the ceph admin user to unlimited. `/etc/security/limits.d`
  `<admin-username> soft nproc unlimited`
- Increase file descriptors on RGW nodes.
  - `vim /etc/security/limits.conf`
    `ceph soft nproc unlimited`

# Kernel Tuning

Tune the kernel for optimal performance

Software performance tuning begins with the kernel.

- Adjust PID count for clusters with a large number of OSDs
  - `/etc/sysctl.conf`
    `kernel.pid_max = 4194303`
- Avoid insufficient memory errors by reserving free RAM: `vm.min_free_kbytes=?`
  - For 64GB reserve 1GB
  - For 128GB reserve 2GB
  - For 256GB reserve 3GB
- Consider turning off NUMA
- Turn off energy saving

## Cluster Tuning

Adjusting the Cluster Map Size

Large RHCS 2 clusters benefit from adjusting the cluster map size.

```
[global]
osd_map_message_max=10
[osd]
osd_map_cache_size=20
osd_map_max_advance=10
osd_map_share_max_epochs=10
osd_pg_epoch_persisted_max_stale=10
```

This isn't necessary in RHCS 3, because the `ceph-manager` daemon offloads pg operations.

# Cluster Tuning

Adjust Default Scrubbing Settings

Scrubbing during high loads can significantly impact I/O leading some to turn off scrubbing. Consider scrubbing during low-load time windows or low load thresholds. For example:

```
[osd]
#night time scrubbing
osd_scrub_begin_hour = 23
osd_scrub_end_hour = 6
osd_scrub_load_threshold = 0.25 #low load scrubbing
osd_scrub_during_recovery = false #scrub during recovery
```

## Cluster Tuning

Adjust Default Backfill Settings

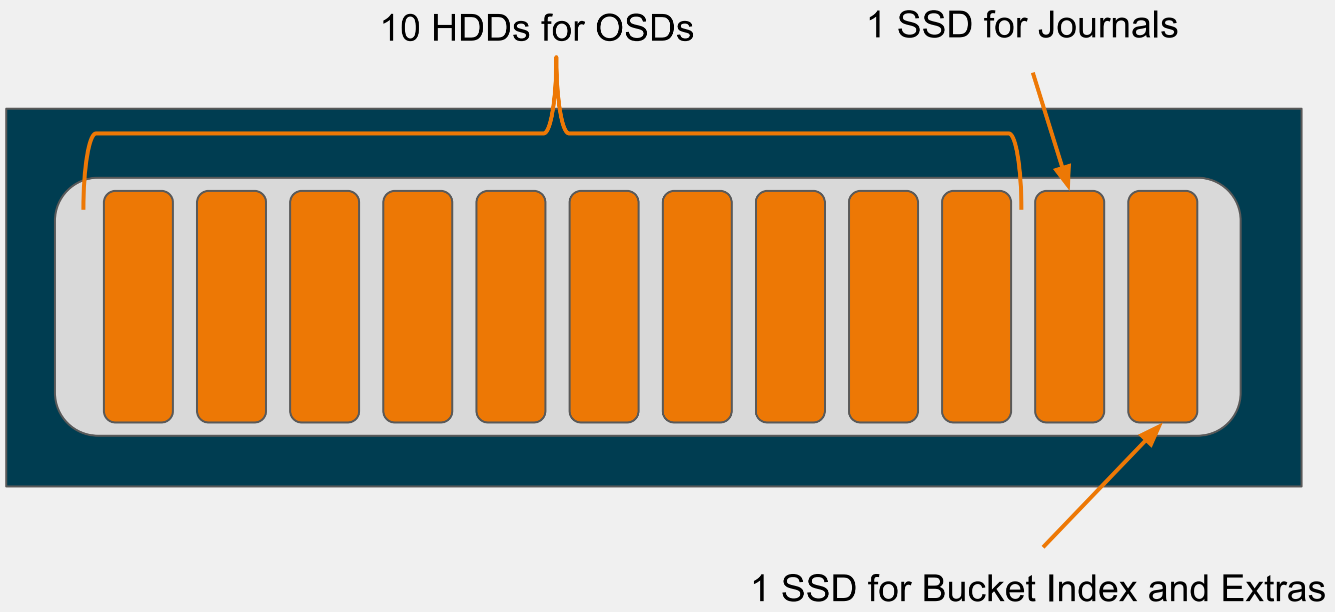
Adjust backfill settings so that backfill doesn’t impact I/O significantly. These can be temporarily increased during low loads if backfill is ongoing.

```
[osd]
osd_max_backfills = 1
osd_recovery_max_active = 1
osd_recovery_op_priority = 1
```

The Object Gateway for Production guide is an invaluable source of information!

## Developing Storage Strategies

# Example Server



# Create CRUSH Hierarchies and Rules

Underlying hardware and rules get mapped to pools

For each use case, create CRUSH hierarchies and for:

- The data pool. The CRUSH rule will use erasure-coding.
- The service pools. ***May*** use the same hierarchy as data, but will use replication.
- Index and extras pools. Separate SSD hierarchy using replication.

# Create CRUSH Hierarchies and Rules

## CRUSH Hierarchy

```
# ceph osd crush add-bucket t-put root
# ceph osd crush add-bucket rack1 rack
# ceph osd crush add-bucket node-a host
# ceph osd crush add-bucket node-b host
# ceph osd crush add-bucket rack2 rack
# ceph osd crush add-bucket node-c host
# ceph osd crush add-bucket node-d host
# ceph osd crush move rack1 root=t-put
# ceph osd crush move node-a rack=rack1
# ceph osd crush move node-b rack=rack1
# ceph osd crush move rack2 root=t-put
# ceph osd crush move node-c rack=rack2
# ceph osd crush move node-d rack=rack2
```

## CRUSH Rules

```
# ceph osd erasure-code-profile set data-profile \
  k=8 \
  m=4 \
  crush-failure-domain=host
  crush-root=throughput
  crush-device-class=hdd

# ceph osd crush rule create-replicated service t-put host hdd
# ceph osd crush rule create-replicated bucket-index t-put host ssd
# ceph osd crush rule create-erasure data data-profile
```

## Create Pools

Pools are the Object Gateway's interface to RADOS

For each data placement policy, create pools for:

- The data bucket. This pool will use erasure coding.
- The services. ***May*** use the same hierarchy as data, but must use replication.
- Index and extras pools. Will use a separate SSD hierarchy with replication.

## Create Service Pools

Service pools may use the same CRUSH hierarchy and rule

Use fewer PGs per pool, because many pools may use the same CRUSH hierarchy.

```
# ceph osd pool create .<zone>.rgw.root <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.control <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.gc <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.log <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.intent-log <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.usage <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.users.keys <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.users.email <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.users.uid <pg> <pgp> replicated service
# ceph osd pool create .<zone>.rgw.users.swift <pg> <pgp> replicated service
```

# Create Data Placement Pools

Service pools may use the same CRUSH hierarchy and rule

Use fewer PGs per pool, because many pools may use the same CRUSH hierarchy.

- The data pool uses the erasure code profile.

```
# ceph osd pool create .<zone>.buckets <pg> <pgp> erasure data
```

- The bucket index profile uses SSDs

```
# ceph osd pool create .<zone>.buckets.index <pg> <pgp> replicated bucket-index
```

- The extras uses SSDs

```
# ceph osd pool create .<zone>.buckets.extra <pg> <pgp> replicated bucket-index
```

## Configure Bucket Index Sharding

Buckets with 100k or more objects benefit from sharding

In RHCS 3 and later releases, bucket index sharding is dynamic!

```
rgw_dynamic_resharding = true
rgw_max_objects_per_shard = 100000
```

## Issues and Solutions

# Flapping OSD’s and Buckets w/ 1M+ Objects

How to identify if this is my issue?
https://access.redhat.com/solutions/2971581

- Where to start
  - Identifying type of workload
    - Having millions of objects means a large amount of PUT operations, but if in parallel a large amount of  DELETE operations are also occurring this can lead to OSD’s flapping.

  - Check if bucket index sharding is used or not?
    - radosgw-admin metadata get bucket.instance:<bucket-name>:<bucket-id> | grep num_shards
    - If value comes as 0 it means no bucket index sharding.
    - If this has some value then need to verify if it is configured as recommended value 100K objects/shard.

  - Check if bucket index RADOS pool is backed by SSD or NVME OSD’s.

## I understand this is my issue...

What can cause this issue?

- Possible causes
    - The first issue is when RGW buckets have millions of objects. This leads to their RADOS objects (bucket index shard) becoming very large, with a high number OMAP keys stored in leveldb. Operations like deep-scrub, bucket index listing, recovery/backfill...etc taking an extended of time to complete, triggering OSD's to flap. If proper bucket sharding is not used this issue can become worse, because then only a minimal (with no sharding only a single one!) RADOS index objects will be holding all the OMAP keys.

    - The second issue is when you have a large amount of DELETE operations, which can lead to a large amount of stale data in OMAP.  This triggers leveldb compaction all the time, which is single threaded and non optimal with this kind of workload. This leads to osd_op_threads suiciding because it is always compacting and the OSD’s begin flapping.

# What’s the Temporary Solution?

What can I do to alleviate this issue?

- Temporary solutions
  - The **first** temporary action should be setting nodeep-scrub flag either globally (ceph osd set nodeep-scrub) or only on the RGW index pool (ceph osd pool set <pool-name> nodeep-scrub 1).
  - Then the **second** temporary step could be taken if OSD's continually are hitting suicide timeout. You would need to increase the OSD op threads normal timeout and suicide timeout values. If filestore op threads are also hitting timeout, then increase normal and suicide timeout for filestore op threads.

## What’s the Temporary Solution?

Temporary Solutions Continued...

- Add these options in [osd.id] section or in [osd] section to make them permanent while troubleshooting this issue is ongoing. You can use ceph tell injectargs command to inject them at run time (if the daemon restarts you will lose the change though!)

```
osd_op_thread_timeout = 90 #default is 15
osd_op_thread_suicide_timeout = 2000 #default is 150
```

If filestore op threads are hitting timeout:

```
filestore_op_thread_timeout = 180 #default is 60
filestore_op_thread_suicide_timeout = 2000 #default is 180
```

Same can be done for recovery thread also.

```
osd_recovery_thread_timeout = 120 #default is 30
osd_recovery_thread_suicide_timeout = 2000 #default is 300
```

## What's the Temporary Solution?

Temporary Solutions Continued...

- The **third** temporary step would be taken if OSD's have very large (above 10GB) OMAP directories. This can be verified with command:

  ```
  du -sh /var/lib/ceph/osd/ceph-$id/current/omap
  ```

- Once validated manual leveldb compaction for OSD's can be done.

  - `ceph tell osd.$id compact` or
  - `ceph daemon osd.$id compact` or
  - Add `leveldb_compact_on_mount = true` in `[osd.$id]` or `[osd]` section and restart the OSD.
  - This makes sure that it compacts the leveldb on startup and then brings the OSD back `up/in` the cluster.

# What’s the permanent resolution?

Permanent resolution steps...

- Permanent Solutions
  - Calculate the bucket index shard RADOS object size
    - Count the OMAP keys in index shard object
      - `rados -p <idx-pool> listomapkeys <idx-shard-obj> | wc -l`
    - Each OMAP key is of 200 bytes for getting the size of object
      - `<count from above command> * 200 = <value in bytes>`
  - If the index shard object is large (above 20 MB), consider resharding the bucket. This is because bucket index sharding is off or it is misconfigured.
    - `radosgw-admin bucket reshard` is the command for resharding. More details can be found in Red Hat documentation. Please note that this is offline reshard tool!
    - Because of these RHCS 3.0 has introduced dynamic resharding.
      - https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/3/html-single/object_gateway_guide_for_red_hat_enterprise_linux/#bucket_sharding

## What’s a permanent resolution to the issue?

Permanent resolution continued...

- Permanent resolution continued
  - If your RGW index pool is not backed by SSD or NVME OSD’s and the OSD’s are running above 80% disk util(Disk bound) during leveldb compaction then consider migrating Index pool to new CRUSH ruleset which is backed by SSD or NVME SSD’s.
    - https://access.redhat.com/documentation/en-us/red_hat_ceph_storage/3/html/storage_strategies_guide/crush_administration
  - If RGW index pool OSD’s are always using above 100% CPU(CPU bound) during leveldb compaction then consider converting omap backend to rocksdb from leveldb.
    - https://access.redhat.com/solutions/3210951
  - RHCS 2.5 introduces support for rocksdb as a backend for omap.

## What’s a permanent resolution to the issue?

Permanent resolution continued...

- Permanent Solutions contd ...
  - With rocksdb support in RHCS 2.5 and RHCS 3.0 these commands can be utilized to convert omap bakend to rocksdb from leveldb:
    - Stop the OSD
    - mv /var/lib/ceph/osd/ceph-<id>/current/omap /var/lib/ceph/osd/ceph-<id>/omap.orig
    - *ulimit -n 65535*
    - ceph-kvstore-tool leveldb /var/lib/ceph/osd/ceph-<id>/omap.orig store-copy /var/lib/ceph/osd/ceph-<id>/current/omap 10000 rocksdb
    - ceph-osdomap-tool --omap-path /var/lib/ceph/osd/ceph-<id>/current/omap --command check
    - sed -i s/leveldb/rocksdb/g /var/lib/ceph/osd/ceph-<id>/superblock
    - chown ceph.ceph /var/lib/ceph/osd/ceph-<id>/current/omap -R
    - cd /var/lib/ceph/osd/ceph-<id>; rm -rf omap.orig
    - Start the OSD
  - If you do not want to follow the above steps (or are uncomfortable doing so) then you can rebuild the OSD with filestore_omap_backend = "rocksdb".

## So that is a lot of information...

The highlights to succeed...

- In summary:
  - Have your RGW index pool backed by SSD or NVME!!!
  - Have proper bucket index shard count value set on your buckets from the start and consider future growth of the bucket
  - Have your OSD's backing the RGW index pool using rocksdb with 8 compaction threads, rocksdb compression disabled and `rocksdb_cache_size` tuned properly, as per your workload, starting point should be 1G and can be increased as your HW allows.
  - If you still continue to see index pool OSD's flapping during deep-scrub operations then you can keep `nodeep-scrub` flag set on the index pool. BZ will fix this issue and you can unset `nodeep-scrub` after upgrading to fixed Luminous version.

RED HAT
SUMMIT

# THANK YOU

plus.google.com/+RedHat

facebook.com/redhatinc

linkedin.com/company/red-hat

twitter.com/RedHat

youtube.com/user/RedHatVideos