

CL'SO

Mastering Ceph Operations

upmap and the balancer

Dan van der Ster

CTO - Clyso & Ceph Executive Council

Motivation



Motivation II

- This past year I've met several Ceph teams in that situation:
 - I send them an obscure tool with a weird name, and it saves the weekend.
 - Those very expert operators call it "magic" and wonder why they didn't know about this before.
- How could this happen? Perhaps it's because we decided that we shouldn't talk about Ceph internals.
 - "Ceph is too hard ... don't talk about PGs ..." – we stopped sharing experience in solving complex maintenance tasks.
 - We wrote a bunch of code to hide complexity beneath a layer of automation, and operators stopped learning about things like PGs.
- So today I'm going to talk about one of my favorite topics on how Ceph works - a remix of something I presented at [Ceph Day Berlin 2018](#)
 - Why we have PGs, what is the role of CRUSH, what is this thing called upmap, and how these things are all so important for optimizing Ceph maintenance at scale.

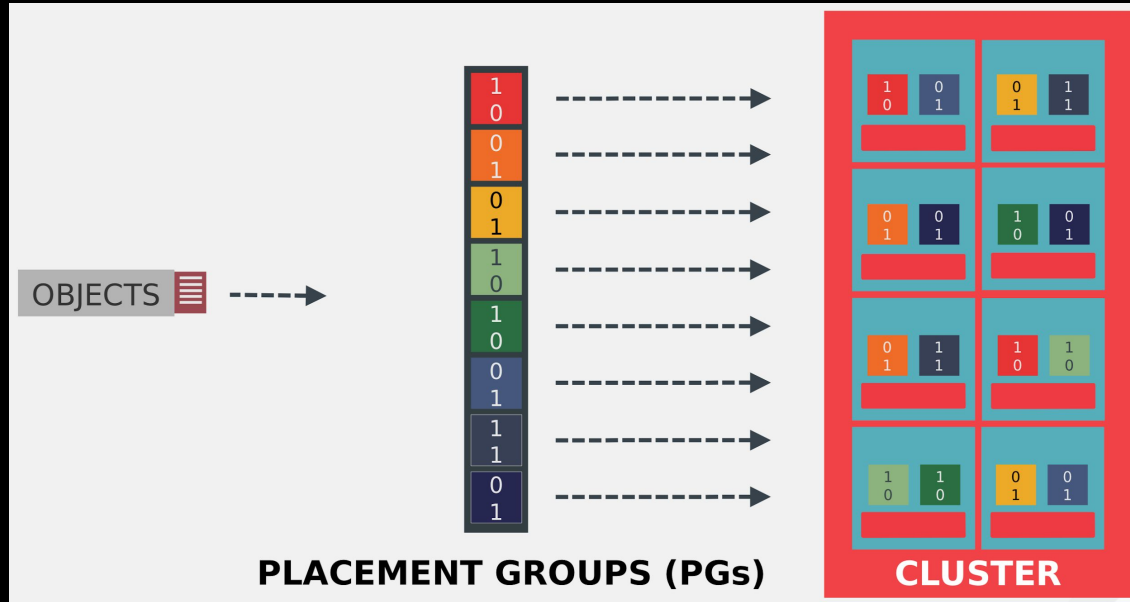
Ceph stores objects

How does it decide where to store them?
And how does it remember?

How to store objects on a cluster?

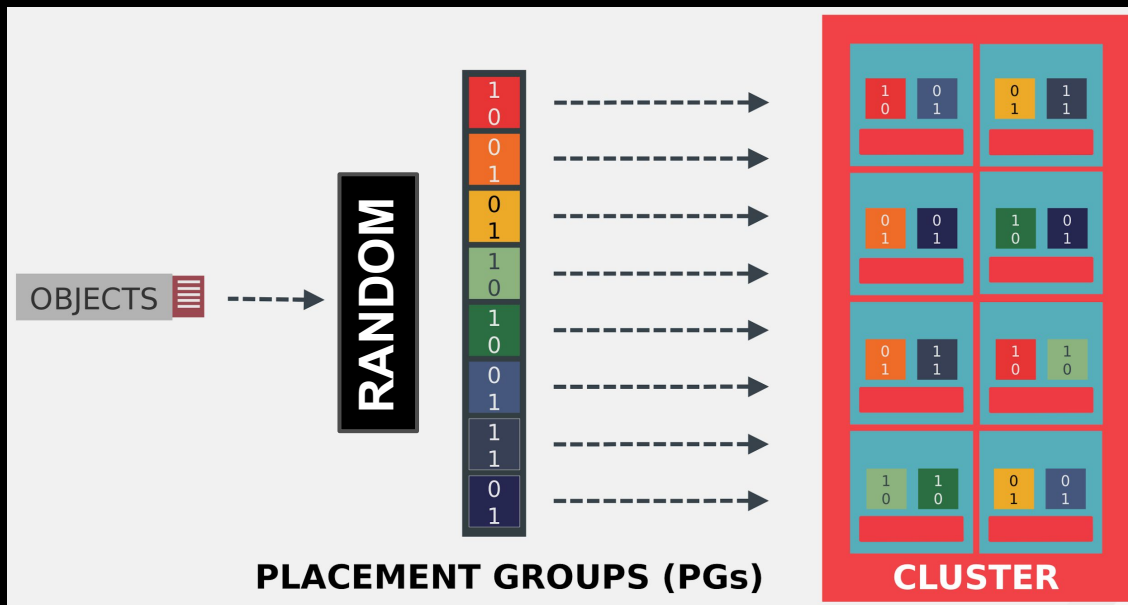


Two-Step Placement with PGs



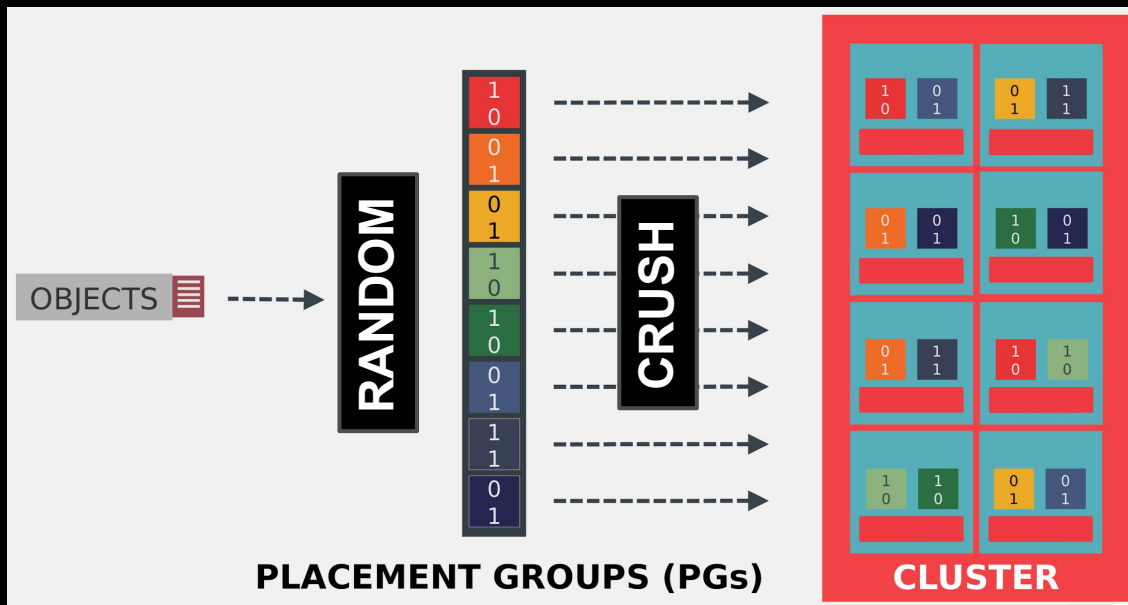
Placement Group: a collection of objects managed as a single unit

Two-Step Placement with PGs



1. Map each **object** to a **PG** uniformly randomly.

Two-Step Placement with PGs

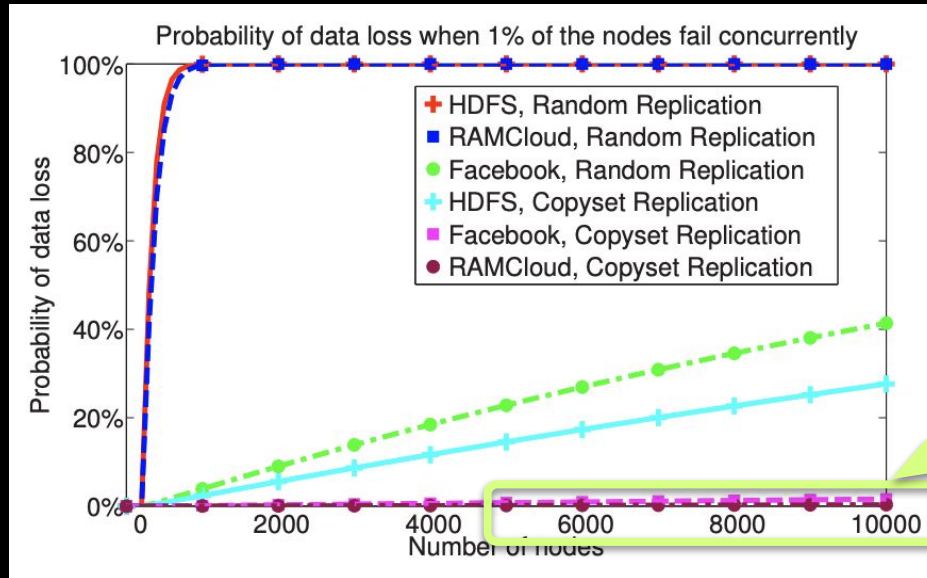


1. Map each **object** to a **PG** uniformly randomly.
2. Map each PG to a set of OSDs using **CRUSH**

What is the purpose of PGs?

- *We map objects to PGs, then map those PGs to OSDs.
Why don't we just map the objects to directly to OSDs using CRUSH?*
- In that scenario we'd be assigning each object to an OSD, one by one.
- The effect would be that all OSDs would be peered with all others.
 - *This would be a disaster for high availability.*
 - For example, *any* concurrent failure of 3+ disks would lead to data loss.
 - Consider a 5000 OSD Ceph cluster:
 - There are 125 billion failure combinations that would cause data loss.
 - We peer each OSD with ~100 others, so the vast majority of such failures are harmless.

Related Concept: Copysets



Ceph is down here too

[Copysets: Reducing the Frequency of Data Loss in Cloud Storage](#), USENIX 2013

Note: Ceph had already used this approach back in 2006

What is the purpose of CRUSH?

- Since the beginning, CRUSH has been Ceph's key feature, its *secret sauce*.
- **CRUSH provides a language** to describe data placement rules according to your infrastructure.
 - E.g. place 3 copies in different servers/racks/rooms/...
 - E.g. place 4+2 erasure coded shards with the first 4 on NVMe in room A and the 2 parity shards in room A on hdds.
- **CRUSH is really fast**: it computes object locations quickly.
- **CRUSH maps are tiny**: can be efficiently shared between servers and clients.
- Together, this means **Ceph doesn't need a large file location DB** like other legacy storage systems. So Ceph is "exascale".

But CRUSH is imperfect :(

- Placing data with CRUSH causes imbalanced usage of the individual OSDs.
- Why is that a problem?
 - A Ceph cluster is full when just one OSD is full.
 - E.g. if one osd is 90% full and the others are 70% full, you have 20% unusable capacity.
 - OSDs with more data do more work, and become a performance bottleneck.
- Why does this happen?
 - We don't have enough PGs to make statistically uniform distributions:
 - Expected stdev = $\sqrt{\text{\#PGs per OSD}}$, e.g. 100 PGs/OSD \Rightarrow 10% stdev \Rightarrow 20% diff between the most and least full OSDs.
 - The “CRUSH multipick anomaly”
 - Using the same crush weight to place 2nd and subsequent copies leads to small OSDs getting more objects than they should.

Summary up to now

1. PGs are great for availability, scalability, and other reasons.
2. CRUSH is a great language and fast algorithm to place data reliably.
3. But CRUSH leads to imperfect usage of the OSDs.

So, why don't we just maintain a lookup table mapping PGs to OSDs?

- It would be quite compact – we only have a few thousand PGs even for multi-PB clusters.
- We could optimize PGs placement according to any rules we want, perfectly.

Summary up to now... upmap!

1. PGs are great for availability, scalability, and other reasons.
2. CRUSH is a great language and fast algorithm to place data reliably.
3. But CRUSH leads to imperfect usage of the OSDs.

So, why don't we just maintain a lookup table mapping PGs to OSDs?

- It would be quite compact – we only have a few thousand PGs even for multi-PB clusters.
- We could optimize PGs placement according to any rules we want, perfectly.

That's exactly what **upmap** is! It's a DB of PG locations.

```
# ceph osd --help | grep upmap
osd pg-upmap <pgid> <osdname (id|osd.id)> [<osdname (id|osd.id)>...]
osd pg-upmap-items <pgid> <osdname (id|osd.id)> [<osdname (id|osd.id)>...]
osd rm-pg-upmap <pgid>
osd rm-pg-upmap-items <pgid>
#
```

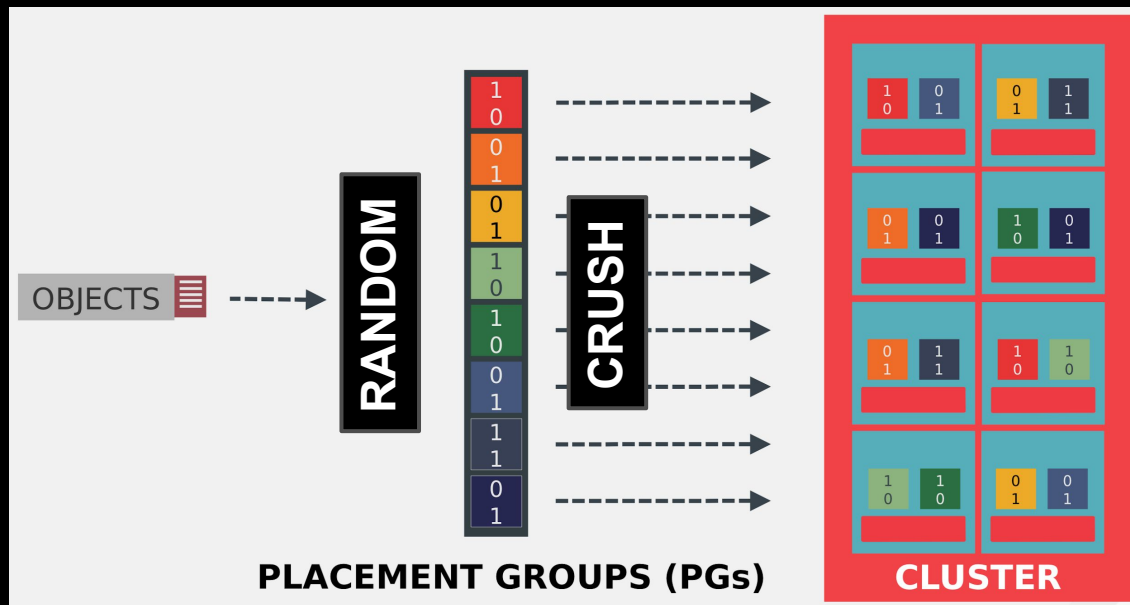
For this PG, if CRUSH says use osd.X, use osd.Y instead.

E.g.:

```
ceph osd pg-upmap-items 1.ff 1 2 3 4
```

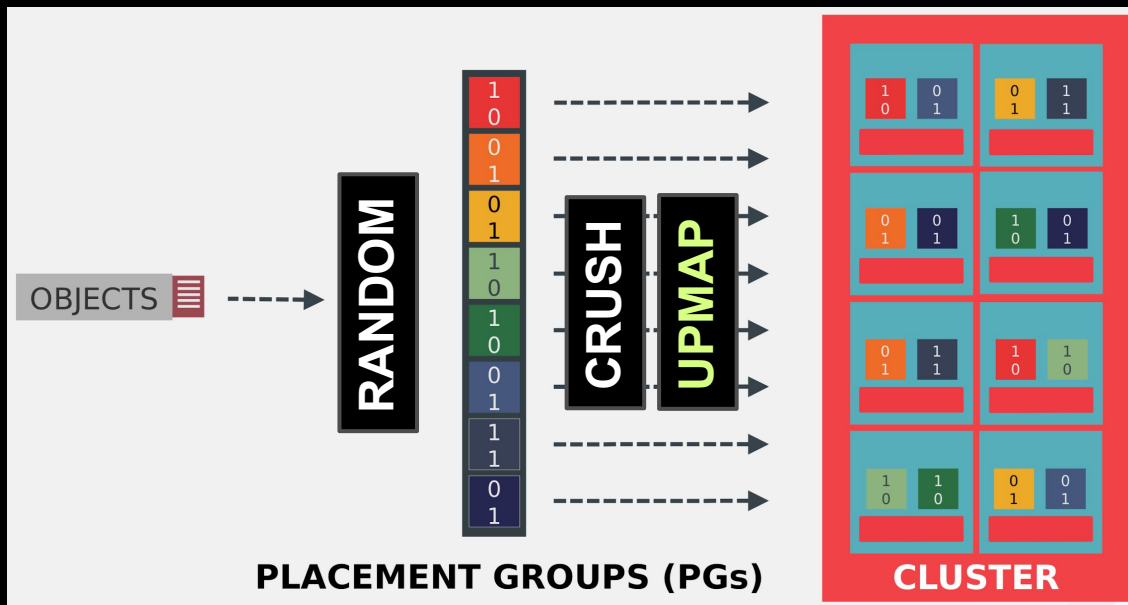
for pg 1.ff, replace osd.1 with osd.2 and replace osd.3 with osd.4

2.1-Step Placement



1. Map each **object** to a **PG** uniformly randomly.
2. Map each PG to a set of OSDs using **CRUSH**

2.1-Step Placement



1. Map each **object** to a **PG** uniformly randomly.
2. Map each PG to a set of OSDs using **CRUSH**
3. Use upmap to improve the output of CRUSH

How is upmap used?

Using upmap to balance data optimally

- *ceph df* says my cluster is 50% full overall, but the *cephfs_data* pool is 75% full. How do I fix that imbalance?
- Ceph has an automatic *balancer* that uses upmap to move PGs from the most full OSDs to the least full OSDs.
- tl;dr: just enable it:

```
ceph osd set-require-min-compat-client luminous

ceph config set mgr/balancer/begin_time 0830
ceph config set mgr/balancer/end_time 1800
ceph config set mgr/balancer/max_misplaced 0.005
ceph config set mgr/balancer/upmap_max_deviation 1

ceph balancer mode upmap
ceph balancer on
```

Large clusters will recover
several petabytes of capacity
using the upmap balancer.

The upmap balancer in action!

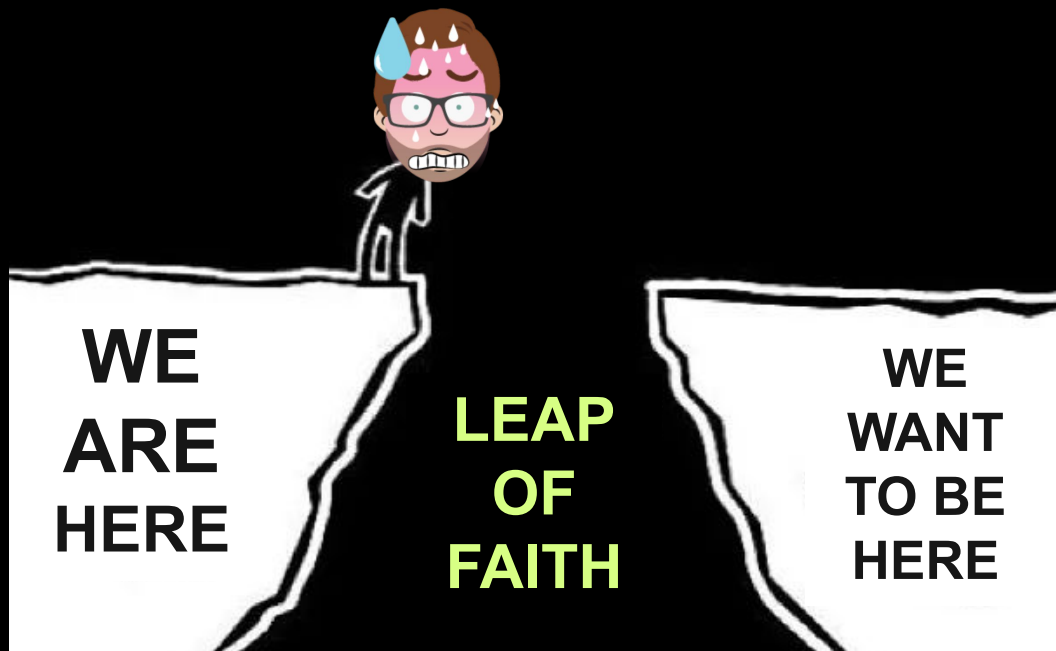
But wait, there's more...

Cluster Maintenance at Scale

- Need to add capacity
- Need to replace OSD servers
- Need to change the CRUSH placement rules
- Need to change the OSD weights
- Need to change the CRUSH tunables

- This type of work involves:
 - moving large amounts of data
 - which last several days or weeks
 - having an unpredictable impact on users
 - with no simple undo button.

Cluster Maintenance at Scale



Cluster Maintenance

- Step 1:
 - Add Capacity: `ceph orch apply osd ..`
 - Drain Hosts: `ceph osd crush reweight-subtree ..`
 - Change tunables: `ceph osd crush tunables ..`
- Step 2:
 - 4518 active+remapped+backfill_wait
 - 2.1B (28%) objects misplaced
 - Progress: Global Recovery Event [.....] (6w)
- Step 3:
 - Wait 6 weeks, or ..
 - Run a magic script.
- Step 4:
 - HEALTH_OK

Cluster Maintenance

```
cluster:  
  id:      e7681812-f2b2-41d1-9009-48b00e614153  
  health: HEALTH_WARN  
          norebalance flag(s) set  
          23828335/102852096 objects misplaced (23.168%)  
  
data:  
  pools:  2 pools, 8448 pgs  
  objects: 34.28M objects, 130TiB  
  usage:   391TiB used, 812TiB / 1.18PiB avail  
  pgs:     23828335/102852096 objects misplaced (23.168%)  
          4053 active+clean  
          3211 active+remapped+backfill_wait  
          1179 active+remapped+backfilling  
          3   active+clean+scrubbing  
          2   active+clean+scrubbing+deep
```

Cluster Maintenance

```
cluster:
  id: e7681812-f2b2-41d1-9009-48b00e614153
  health: HEALTH_WARN
        norebalance flag(s) set
        23828335/102852096 objects misplaced (23.168%)

data:
  pools: 2 pools, 8448 pgs
  objects: 34.28M objects, 130TiB
  usage: 391TiB used, 812TiB / 1.18PiB avail
  pgs: 23828335/102852096 objects misplaced (23.168%)
       4053 active+clean
       3211 active+remapped+backfill_wait
       1179 active+remapped+backfilling
       3 active+clean+scrubbing
       2 active+clean+scrubbing+deep
```

What if we could
“upmap” those
remapped PGs
back to where the
data is now?

Cluster Maintenance with upmap

```
set 64.80 pg_upmap_items mapping to [58->288,291->1322,1351->37]
set 5.45d pg_upmap_items mapping to [659->191]
set 4.1042 pg_upmap_items mapping to [291->662]
set 5.774 pg_upmap_items mapping to [218->1424,439->51]
set 64.e8 pg_upmap_items mapping to [1313->48,555->1248,659->253]
set 4.f5c pg_upmap_items mapping to [199->322]
set 4.112f pg_upmap_items mapping to [1352->1329,593->470,646->606]
set 64.20c pg_upmap_items mapping to [291->64,317->219,1390->1467]
set 5.2d2 pg_upmap_items mapping to [291->176,35->314,335->383]
set 4.fe1 pg_upmap_items mapping to [646->1467]
set 64.203 pg_upmap_items mapping to [1425->1470,659->1455]
clear 4.f08 pg_upmap_items mapping
set 5.90 pg_upmap_items mapping to [401->265,291->147]
```

Runs magic script...

Cluster Maintenance with upmap

```
cluster:  
  id:      e7681812-f2b2-41d1-9009-48b00e614153  
  health: HEALTH_OK  
  
data:  
  pools:   2 pools, 8448 pgs  
  objects: 34.29M objects, 130TiB  
  usage:   391TiB used, 812TiB / 1.18PiB avail  
  pgs:     8444 active+clean  
           2      active+clean+scrubbing+deep  
           2      active+clean+scrubbing
```

Cluster Maintenance with upmap

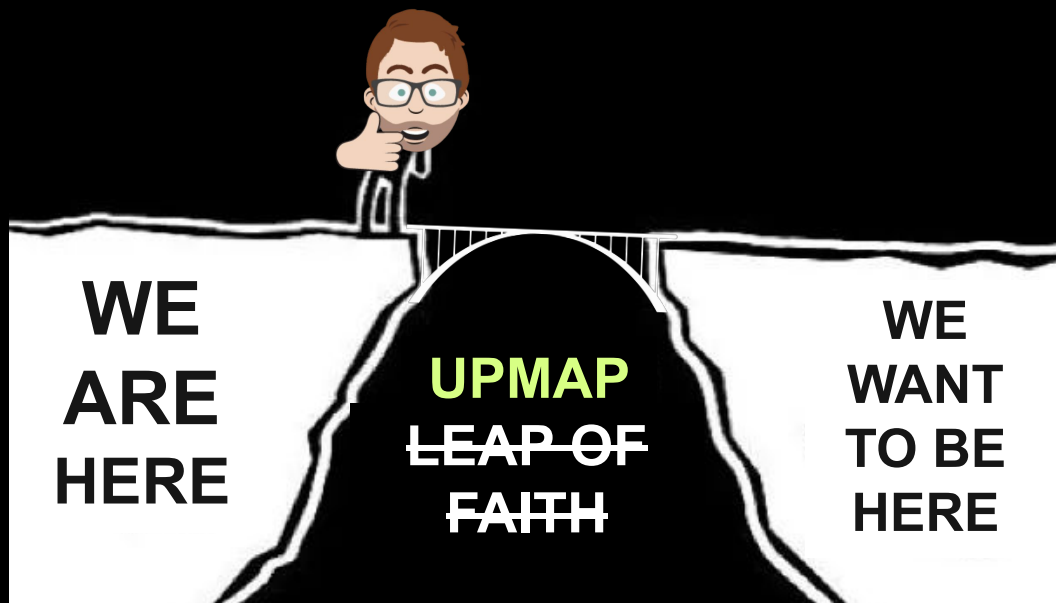
```
cluster:  
  id:      e7681812-f2b2-41d1-9009-48b00e614153  
  health: HEALTH_OK
```

```
data:  
  pools:   2 pools, 8448 pgs  
  objects: 34.29M objects, 130TiB  
  usage:   391TiB used, 812TiB / 1.18PiB avail  
  pgs:     8444 active+clean  
           2      active+clean+scrubbing+deep  
           2      active+clean+scrubbing
```

HEALTH_OK allows the operator sleep at night.

The automatic balancer works behind the scenes to balance data gradually.

No leap of faith required



The Magic Script

- Usage: upmap-remappedTM | sh -x

- (Digital Ocean maintains a similar tool pgremapper)

- Planning the contribution as a core “it-just-works” feature of the balancer.

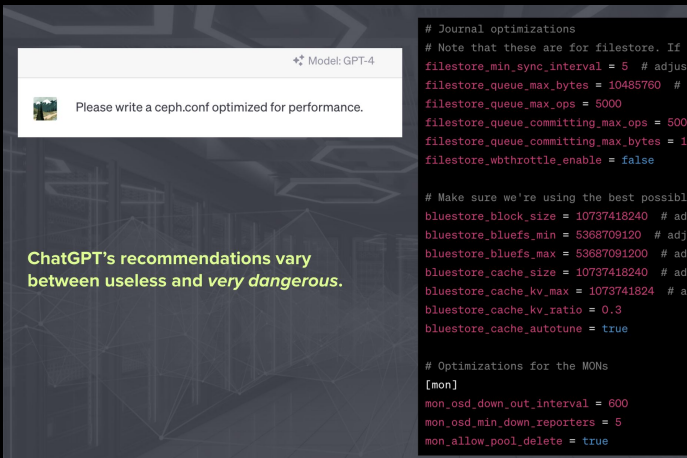


One more thing...

Towards a Ceph Copilot

Clyso's Ceph Copilot Assistant:

- Best Practices compliance with [Ceph Analyzer](#)
- Collecting planning, maintenance, debugging utils, ...
- Chatbot...
- Check in our Clyso [github](#) later this quarter



Model: GPT-4

Please write a ceph.conf optimized for performance.

ChatGPT's recommendations vary between useless and very dangerous.

```
# Journal optimizations
# Note that these are for filestore. If :
filestore_min_sync_interval = 5 # adjus
filestore_queue_max_bytes = 10485760 # :
filestore_queue_max_ops = 5000
filestore_queue_committing_max_ops = 5000
filestore_queue_committing_max_bytes = 1
filestore_wbthrottle_enable = false

# Make sure we're using the best possible
bluestore_block_size = 10737418240 # ad:
bluestore_bluefs_min = 5368709120 # adj:
bluestore_bluefs_max = 53687091200 # ad:
bluestore_cache_size = 10737418240 # ad:
bluestore_cache_kv_max = 1073741824 # a:
bluestore_cache_kv_ratio = 0.3
bluestore_cache_autotune = true

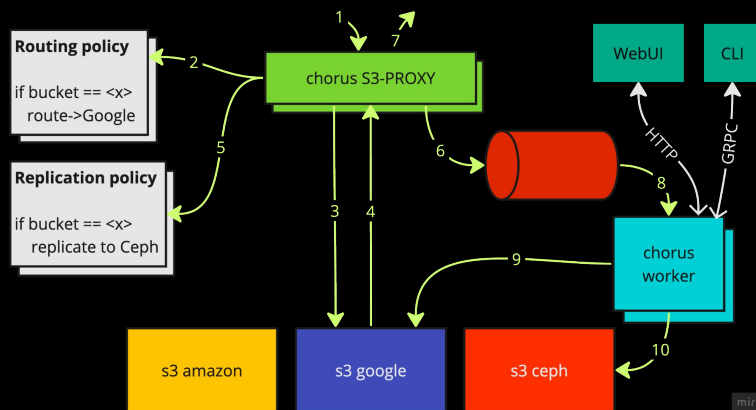
# Optimizations for the MONs
[mon]
mon_osd_down_out_interval = 600
mon_osd_min_down_reporters = 5
mon_allow_pool_delete = true
```



And since there's time another one last thing...

Chorus: An S3 Frontend Swiss Army Knife

- My colleagues Artem and Sirisha couldn't be here to present:
- Clyso recently released [Chorus](#) at FOSDEM 2024:
 - Apache 2.0 licensed S3 frontend for transparent traffic routing, data migration, bucket replication, etc. between Ceph/S3-compliant backends



Upmap-Remapped

1. Configure the cluster for upmap.
 - a. `ceph osd set-require-min-compat-client luminous`
 - b. `ceph config set mgr/balancer/begin_time 0830`
 - c. `ceph config set mgr/balancer/end_time 1800`
 - d. `ceph config set mgr/balancer/max_misplaced 0.005`
 - e. `ceph config set mgr/balancer/upmap_max_deviation 1`
2. Removing the crush-compat → upmap-remapped after
3. Putting old hosts into “fake” racks with one host in each rack, then change the crush rule to rack failure domain → upmap-remapped after
4. Add new hosts, in “real” racks, crush weight 0. No data should move.
5. ``ceph osd crush reweight-subtree`` the new racks/hosts (e.g. to 8TB) → run upmap-remapped after. Turn on the balancer, in mode “upmap” so it starts moving the data.
6. Primary affinity of the old osds to 0.
7. Drain the old hosts (reweight-subtree) to 0.1, then run upmap-remapped.
8. Then finally set crush weights to 0 and remove from the cluster.