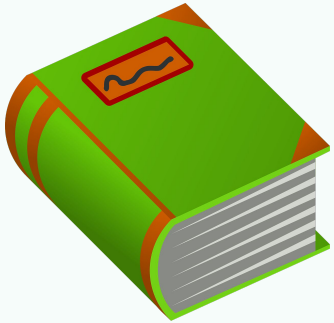

ceph
cephalocon
2023
New Read Balancer in Ceph
Laura Flores

Why is read balancing important for Ceph?

- In distributed storage systems like Ceph, it is important to balance **write** and **read** requests for optimal performance.
  - To balance “writes” → to ensure fast storage and replication of data in a cluster
  - To balance “reads” → to ensure quick access and retrieval of data in a cluster
- The existing capacity (upmap) balancer works well to balance write requests, but not read requests.

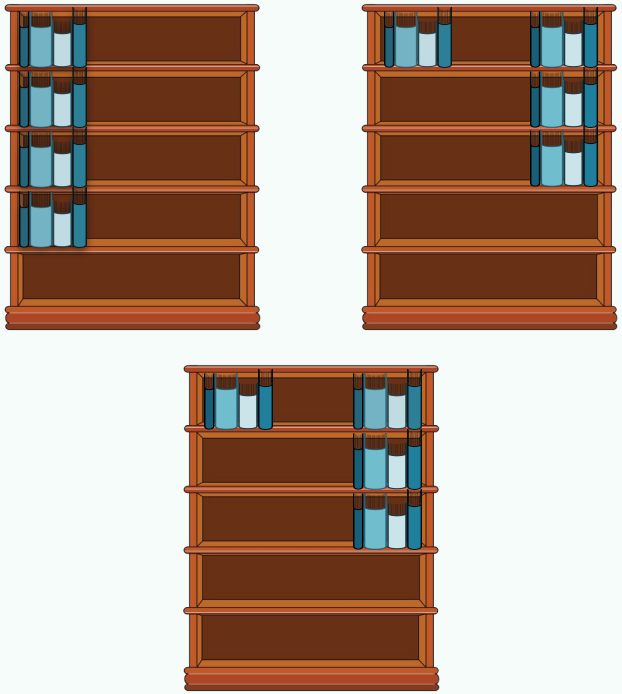


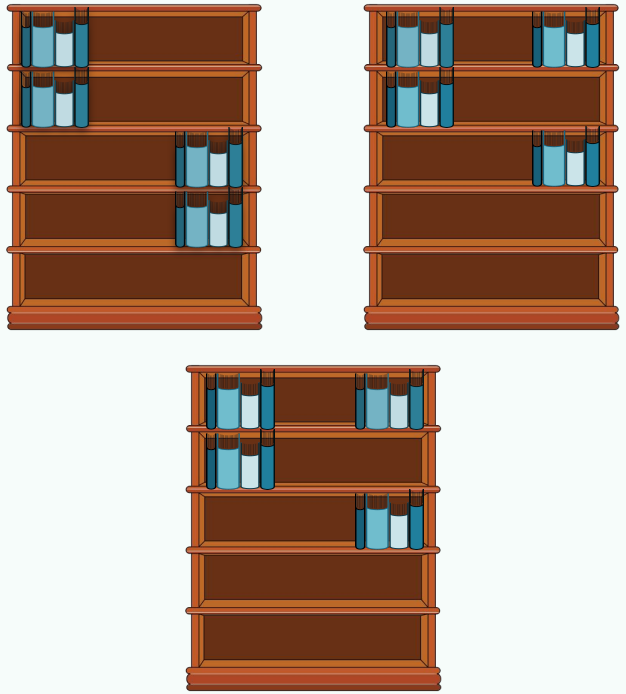
Capacity Balancing vs. Read Balancing

A functional requirement • A strict requirement for Ceph's functionality • If a device fills up, we lose capacity of the entire cluster	**A performance requirement** • Not a strict requirement for Ceph's functionality; helps the system "work" better • Unbalanced reads → reduced overall cluster bandwidth
Balancing is expensive • It requires data movement (by definition) • It takes time, and during this time the performance of the system is reduced.	**Balancing is cheap** • It is just a metadata operation, fast and involves no data movement • No impact on the cluster performance (except improved performance when the operation completes – almost immediately)
It balances the write performance … • … if all devices are homogeneous (same size and performance)	**It balances the read performance …** • … if all devices are homogeneous (same size and performance) • BUT - in future versions, the same mechanism can be used to improve overall cluster performance in heterogeneous systems

Let’s balance capacity in a library…

Let’s balance “primaries” in a library…





What did primary balancing look like before Reef?

- No active balancing code or dedicated mechanism
- On homogeneous systems:
  - CRUSH random distribution improves with number of PGs
  - In general, larger systems are quite balanced, smaller system may not be balanced
  - Extreme unbalanced situations occur more often in small, homogeneous clusters (ODF).
- On heterogeneous systems:
  - Larger systems tend to become heterogeneous over time
- Conclusion:
  - Read balancer (the Reef feature) is useful mostly for ODF clusters
  - Workload Balancer (planned for future versions, will be based on the read balancer infrastructure) will be useful for large, heterogeneous clusters

Stepping Stone #1: Read Balance Score

- Added a new metric to the output of: `ceph osd pool ls detail`
- (applicable only for replicated pools)

```
[jsalomon@Josh-laptop build$]./bin/ceph osd pool ls detail 2>/dev/null
pool 1 '.rgw.root' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 16 pgp_num 16 autoscale_mode on last_change 46 l
for 0/0/34 flags hashpspool stripe_width 0 application rgw read_balance_score 2.00
pool 2 'default.rgw.log' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 16 pgp_num 4 pgp_num_target 16 autoscale_m
ode on last_change 59 lfor 0/0/36 flags hashpspool stripe_width 0 application rgw read_balance_score 3.00
pool 3 'default.rgw.control' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 1 pgp_num 1 autoscale_mode on last_cha
nge 25 flags hashpspool stripe_width 0 application rgw read_balance_score 3.00
pool 4 'default.rgw.meta' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 16 pgp_num 4 pgp_num_target 16 autoscale_
mode on last_change 59 lfor 0/0/38 flags hashpspool stripe_width 0 pg_autoscale_bias 4 application rgw read_balance_score 1.00
pool 5 'default.rgw.buckets.index' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 16 pgp_num 16 autoscale_mode on
last_change 46 lfor 0/0/40 flags hashpspool stripe_width 0 pg_autoscale_bias 4 application rgw read_balance_score 1.25
pool 6 'default.rgw.buckets.data' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 16 pgp_num 16 autoscale_mode on l
ast_change 46 lfor 0/0/42 flags hashpspool stripe_width 0 application rgw read_balance_score 1.50
pool 7 'ecp' erasure profile default size 3 min_size 2 crush_rule 1 object_hash rjenkins pg_num 16 pgp_num 16 autoscale_mode on last_chan
ge 49 flags hashpspool stripe_width 8192
```

The Meaning of the Read Balance Score

- A score of 1.0 is optimal
  - In most cases we can't achieve this score
  - When all primaries are in the same OSD, the score is the replica count
- The score is in the range [1.0, replica_count]
- A score of 1 + x shows approximately (x*100)% degradation in pool read performance under load
  - Score of 1.2 is roughly 20% degradation.

More information in the json output…

```
[jsalomon@Josh-laptop build$]./bin/ceph osd pool ls detail -f json-pretty | jq '.[] | select(.pool == 6)'
```

… skipped most of the json output here …

```
  "application_metadata": {
    "rgw": {}
  },
  "read_balance": {
    "score_acting": 1.5,
    "score_stable": 1.5,
    "optimal_score": 1,
    "raw_score_acting": 1.5,
    "raw_score_stable": 1.5,
    "primary_affinity_weighted": 1,
    "average_primary_affinity": 1,
    "average_primary_affinity_weighted": 1
  }
```

⇐ used to be the last object in the json/xml file

⇐ A new object with more information on read balance score
⇐ scores calculated on primaries as well as acting primaries
⇐ optimal score may change when we have devices with low primary affinity

⇐ We have information about the score calculation before and after adjustments to the primary affinity and its weights

More information in the json output…

```
[jsalomon@Josh-laptop build$]./bin/ceph osd pool ls detail -f json-pretty | jq '.[] | select(.pool == 6)'
```

… skipped most of the json output here …


```
  "application_metadata": {
    "rgw": {}
  },
  "read_balance": {
    "score_acting": 1.5,
    "score_stable": 1.5,
    "optimal_score": 1,
    "raw_score_acting": 1.5,
    "raw_score_stable": 1.5,
    "primary_affinity_weighted": 1,
    "average_primary_affinity": 1,
    "average_primary_affinity_weighted": 1
  }
```


- Score = raw_score / optimal_score
- Weighted info takes into account the OSD sizes, in this case all OSDs are in the same size hence **`average_primary_affinity`** and **`average_primary_affinity_weighted`** are identical.

Stepping Stone #2: New Commands

- Added two commands to change the primary OSD for a PG in a replicated pool.
- The new primary must be one of the OSDs which participate in this PG.

```
ceph osd pg-upmap-primary <pgid> <osd_id>
ceph osd rm-pg-upmap-primary <pgid>
```

```
[jsalomon@Josh-laptop build$]./bin/ceph pg dump pgs_brief | grep "6\.1"
dumped pgs_brief
6.1                  active+clean  [1,3,2]          1  [1,3,2]              1

[jsalomon@Josh-laptop build$]./bin/ceph osd pg-upmap-primary 6.1 3
change primary for pg 6.1 to osd.3

[jsalomon@Josh-laptop build$]./bin/ceph pg dump pgs_brief | grep "6\.1"
dumped pgs_brief
6.1                  active+clean  [3,1,2]          3  [3,1,2]              3
```

High Level Design of the Read Balancer

For one pool:

- Calculate the **desired primary distribution** that should be on each OSD
  - For each OSD, the desired # of primary pgs =

```
(num_pgs / replica_count) * primary_affinity(OSD)
```

- Swap which OSD is primary on select pgs to achieve the desired distribution.
- No data movement is involved; swapping only occurs with OSDs already involved in a pg.

Example: Equal Primary Affinity (step 1)

Check ceph status

```
$ ./bin/ceph -s

  cluster:
    id:     9db145d4-1239-4a68-982b-293d8af14229
    health: HEALTH_OK

  services:
    mon: 3 daemons, quorum a,b,c (age 32m)
    mgr: x(active, since 32m)
    mds: 1/1 daemons up, 2 standby
    osd: 4 osds: 4 up (since 32m), 4 in (since 32m)
    rgw: 2 daemons active (1 hosts, 1 zones)

  data:
    volumes: 1/1 healthy
    pools:   7 pools, 184 pgs
    objects: 232 objects, 458 KiB
    usage:   4.0 GiB used, 400 GiB / 404 GiB avail
    pgs:     184 active+clean
```

Example: Equal Primary Affinity (step 2)

Check pool details for current read balancer scores.
We'll try to improve the score of pool 6, which has ~63% read performance degradation.

```
$ ./bin/ceph osd pool ls detail

pool 1 '.mgr' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 8 pgp_num 8 autoscale_mode on last_change 34 lfor 0/
0/32 flags hashpspool stripe_width 0 pg_num_max 32 pg_num_min 8 application mgr read_balance_score 1.50
pool 2 'cephfs.a.meta' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 16 pgp_num 16 autoscale_mode on last_change
 107 lfor 0/0/38 flags hashpspool stripe_width 0 pg_autoscale_bias 4 pg_num_min 16 recovery_priority 5 application cephfs read_balance_s
core 1.50
pool 3 'cephfs.a.data' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 32 pgp_num 32 autoscale_mode on last_change
 44 lfor 0/0/38 flags hashpspool stripe_width 0 application cephfs read_balance_score 1.63
pool 4 '.rgw.root' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 32 pgp_num 32 autoscale_mode on last_change 44
lfor 0/0/40 flags hashpspool stripe_width 0 application rgw read_balance_score 1.25
pool 5 'default.rgw.log' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 32 pgp_num 32 autoscale_mode on last_chan
ge 146 lfor 0/0/40 flags hashpspool stripe_width 0 application rgw read_balance_score 1.25
pool 6 'default.rgw.control' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 32 pgp_num 32 autoscale_mode on last_
change 44 lfor 0/0/42 flags hashpspool stripe_width 0 application rgw read_balance_score 1.63
pool 7 'default.rgw.meta' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 32 pgp_num 32 autoscale_mode on last_cha
nge 44 lfor 0/0/42 flags hashpspool stripe_width 0 pg_autoscale_bias 4 application rgw read_balance_score 1.25
```

Example: Equal Primary Affinity (step 3)

Get the latest copy of the osd map

```
$ ./bin/ceph osd getmap -o om

got osdmap epoch 150
```

Example: Equal Primary Affinity (step 4)

Run the **upmap balancer** first to make sure writes are balanced. In this case, the upmap balancer was unable to optimize further.

```
$ ./bin/osdmaptool om --upmap out.txt

./bin/osdmaptool: osdmap file 'om'
writing upmap command output to: out.txt
checking for upmap cleanups
upmap, max-count 10, max deviation 5
pools default.rgw.meta default.rgw.control default.rgw.log .rgw.root cephfs.a.meta .mgr cephfs.a.data
prepared 0/10 changes
Unable to find further optimization, or distribution is already perfect
```

Example: Equal Primary Affinity (step 5)

Run the **read balancer**, focusing on the unbalanced pool from earlier, “default.rgw.control”...

```
$ ./bin/osdmaptool om --vstart --read out.txt --read-pool default.rgw.control

./bin/osdmaptool: osdmap file 'om'
writing upmap command output to: out.txt

---------- BEFORE ------------
 osd.0 | primary affinity: 1 | number of prims: 10
 osd.1 | primary affinity: 1 | number of prims: 6
 osd.2 | primary affinity: 1 | number of prims: 3
 osd.3 | primary affinity: 1 | number of prims: 13

read_balance_score of 'default.rgw.control': 1.63


---------- AFTER ------------
 osd.0 | primary affinity: 1 | number of prims: 8
 osd.1 | primary affinity: 1 | number of prims: 8
 osd.2 | primary affinity: 1 | number of prims: 8
 osd.3 | primary affinity: 1 | number of prims: 8

read_balance_score of 'default.rgw.control': 1


num changes: 9
```

Example: Equal Primary Affinity (step 6)

```
$ cat out.txt

./bin/ceph osd pg-upmap-primary 6.0 2
./bin/ceph osd pg-upmap-primary 6.2 2
./bin/ceph osd pg-upmap-primary 6.3 2
./bin/ceph osd pg-upmap-primary 6.5 1
./bin/ceph osd pg-upmap-primary 6.6 2
./bin/ceph osd pg-upmap-primary 6.7 1
./bin/ceph osd pg-upmap-primary 6.8 0
./bin/ceph osd pg-upmap-primary 6.9 2
./bin/ceph osd pg-upmap-primary 6.b 0
```

We can check to see what the balancer suggests in the “out.txt” file…

Example: Equal Primary Affinity (step 7)

We can apply the file to a live system, using `source` or by issuing the ceph commands by hand.

```
$ source out.txt

change primary for pg 6.0 to osd.2
change primary for pg 6.2 to osd.2
change primary for pg 6.3 to osd.2
change primary for pg 6.5 to osd.1
change primary for pg 6.6 to osd.2
change primary for pg 6.7 to osd.1
change primary for pg 6.8 to osd.0
change primary for pg 6.9 to osd.2
change primary for pg 6.b to osd.0
```

Example: Equal Primary Affinity (step 8)

Notice how the score has improved to **1.00** (0% degradation) for “default.rgw.control”, which previously had a score of **1.63** (63% degradation).

```
$ ./bin/ceph osd pool ls detail

pool 1 '.mgr' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 8 pgp_num 8 autoscale_mode on last_change 34 lfor 0/
0/32 flags hashpspool stripe_width 0 pg_num_max 32 pg_num_min 8 application mgr read_balance_score 1.50
pool 2 'cephfs.a.meta' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 16 pgp_num 16 autoscale_mode on last_change
 107 lfor 0/0/38 flags hashpspool stripe_width 0 pg_autoscale_bias 4 pg_num_min 16 recovery_priority 5 application cephfs read_balance_s
core 1.50
pool 3 'cephfs.a.data' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 32 pgp_num 32 autoscale_mode on last_change
 44 lfor 0/0/38 flags hashpspool stripe_width 0 application cephfs read_balance_score 1.63
pool 4 '.rgw.root' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 32 pgp_num 32 autoscale_mode on last_change 44
lfor 0/0/40 flags hashpspool stripe_width 0 application rgw read_balance_score 1.25
pool 5 'default.rgw.log' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 32 pgp_num 32 autoscale_mode on last_chan
ge 146 lfor 0/0/40 flags hashpspool stripe_width 0 application rgw read_balance_score 1.25
pool 6 'default.rgw.control' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 32 pgp_num 32 autoscale_mode on last_
change 44 lfor 0/0/42 flags hashpspool stripe_width 0 application rgw read_balance_score 1.00
pool 7 'default.rgw.meta' replicated size 3 min_size 1 crush_rule 0 object_hash rjenkins pg_num 32 pgp_num 32 autoscale_mode on last_cha
nge 44 lfor 0/0/42 flags hashpspool stripe_width 0 pg_autoscale_bias 4 application rgw read_balance_score 1.25
```

Future Plans and Improvements

- Conduct performance tests
- Add read balancer integration tests to the teuthology rados suite
- Turn the read balancer on by default as part of the balancer manager module
- Account for devices of different sizes

All materials and scripts used in this presentation are available at:
https://github.com/ljflores/ceph_read_balancer_2023

ceph

cephalocon 2023