Dynamic load change in software defined storage systems

How to make SDS systems run better in the cloud

Josh Salomon, Senior Principal Software Engineer, Red Hat
jsalomon@redhat.com

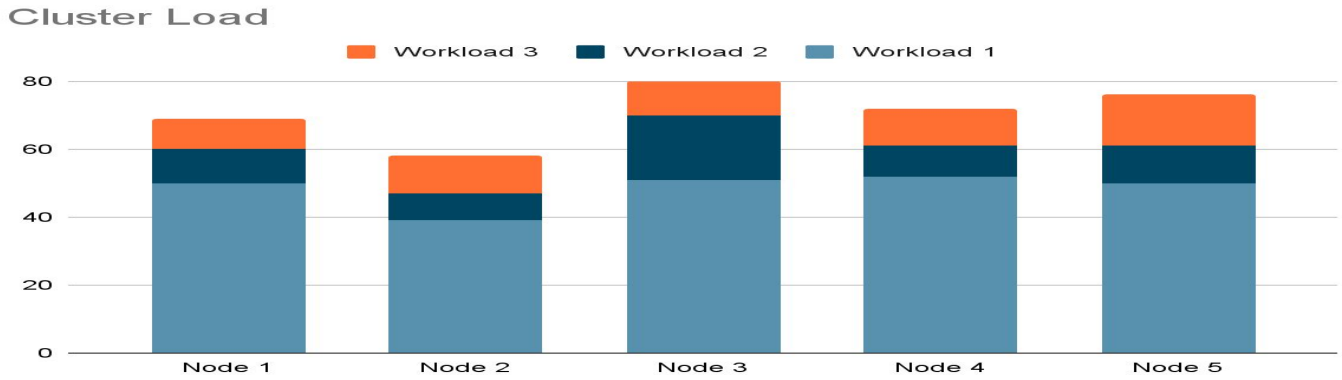
Download this presentation from https://github.com/JoshSalomon/FOSDEM-2023

Red Hat

Agenda

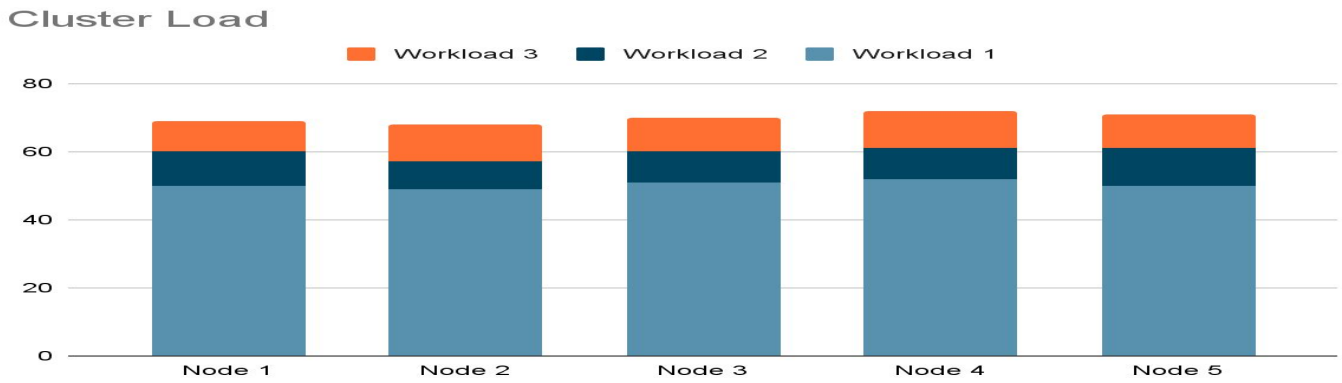
- **What is optimal cluster performance and why we need it**
- Ceph read balancer (added in Reef)
- Read balancer future plans
- How can we use read balancer infrastructure for dynamic load balancing

Real Cluster Performance (at some point in time)



What happens when one node reaches 100% load?

Optimal Cluster Performance



All nodes reach 100% load simultaneously

Cluster Performance - What we want

- Flexible structure with a fixed volume
  - Volume = performance

Cluster Performance - The Reality

- Flexible structure with a fixed volume
  - The balloon is made of lego bricks (nodes and workloads)
  - Much less flexibility ...
  - ... but some flexibility exists

From Real Cluster Performance to More Optimal Performance

Notes:

- Only workload 3 was changed
- Total amount of workload 3 is the same
- This is a presentation 😉 - in reality we may have some restrictions that prevent perfect balance.
- However today we will see how we can change one workload (ceph) to better fit the entire cluster workload.

Agenda

- ✔ What is optimal cluster performance and why we need it
- **Ceph read balancer (added in Reef)**
- Read balancer future plans
- How can we use read balancer infrastructure for dynamic load balancing

Ceph Read Balancer - What is it?

- A mechanism to change the primary OSDs for PGs in replicated pools
  - A score that shows how reads are split among the OSDs

```
$ ./bin/ceph osd pool ls detail
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 47 lfor 0/0/45 flags hashpspool stripe_width 0 application rgw read_balance_score 1.41
```

  - 2 new commands :
    - *ceph osd pg-upmap-primary* and *ceph osd rm-pg-upmap-primary*
    - Metadata only commands - almost immediate execution
  - A new *osdmaptool* subcommand that calculates the changes needed for optimal read balance configuration
    - *osdmaptool <in-file>* ***--read*** *<out-file>* ***--read-pool*** *<pool>*

Ceph Read Balancer - Example

```
$ ./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: 11
 osd.1 | primary affinity: 1 | number of prims: 6
 osd.3 | primary affinity: 1 | number of prims: 15

read_balance_score of 'default.rgw.control': 1.40625


---------- AFTER ------------
 osd.0 | primary affinity: 1 | number of prims: 10
 osd.1 | primary affinity: 1 | number of prims: 11
 osd.3 | primary affinity: 1 | number of prims: 11

read_balance_score of 'default.rgw.control': 1.03125


num changes: 6
```

```
$ cat out.txt

./bin/ceph osd pg-upmap-primary 6.5 1
./bin/ceph osd pg-upmap-primary 6.7 1
./bin/ceph osd pg-upmap-primary 6.9 1
./bin/ceph osd pg-upmap-primary 6.b 0
./bin/ceph osd pg-upmap-primary 6.c 1
./bin/ceph osd pg-upmap-primary 6.f 1
```

Ceph Read Balancer - Implementation

- Composed of 2 functions
  - `OSDMap::calc_desired_primary_distribution`
    - A policy function that can be changed
    - Initial implementation - in every OSD 1/replica_count of the PGs are primaries, adjusted to the OSD primary_affinity values.
  - `OSDMap::balance_primaries`
    - The overall balancing algorithm - bring the cluster configuration as close as possible to the output of `calc_desired_primary_distribution`
    - An infrastructure method that can be used as is.

Agenda

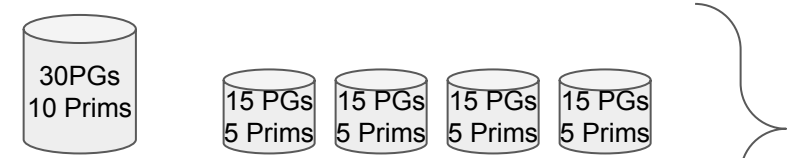
- ✓ What is optimal cluster performance and why we need it
- ✓ Ceph read balancer (added in Reef)
- **Read balancer future plans**
- How can we use read balancer infrastructure for dynamic load balancing

Ceph Read Balancer - What we can do more?

- What does the framework provide?
  - A mechanism for deterministically controlling the number of primary OSDs across the cluster.
- How can we use it?
- First usecase: Load balance better on heterogeneous systems.
  - Example: a system with devices with same technology and different capacity.
  - Assume we have a system with 4 OSDs of 1TB, 1 OSD of 2 TB and replica 3 - all the devices have the same bandwidth and IOPs
  - Assume also each RADOS object is kept on the large OSD with 2 replicas on small OSDs
  - For convenience let's assume that each device can support 100 IOPs

Can We Improve?

Under full load:



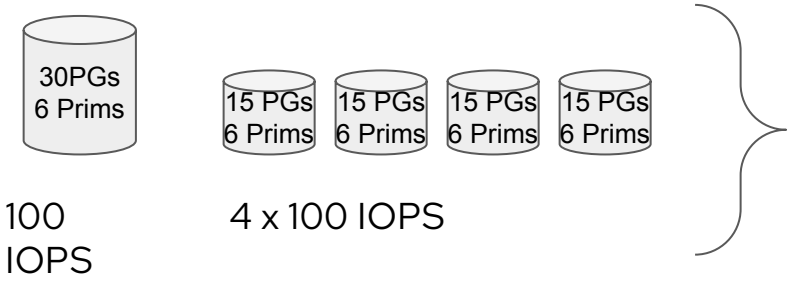
100 IOPS

4 x 50 IOPS

Total (server side): 300 IOPS, 30 Primary PGs

Yes we Can!

Under full read-only load:



- We can improve from 300 IOPs to 500 IOPs by just a small change in the workload distribution so it is evenly distributed across all OSDs.
- Can we improve on other workloads?

We Can Do Even More! (1/2)

- **Yes** - we can improve the performance of more workloads on heterogeneous systems under some conditions:
  - We need to have a reasonable assumption on the read/write ratio.
    - Since the read balancer works per pool - we should be able to assume this ratio or get it from the user.
- **But** - there is a limitation to what we can do:
  - E.g. if we work with replica-3 and some devices with 5x the capacity of other devices in the pool
  - We can improve the performance but we can't get to the optimal performance (all devices in the pool work in full capacity)
  - This method does not help for write only worklads

We Can Do Even More! (2/2)

- Second usecase: If we have mixed technology devices (such as SSD and HDD) in the same pool:
  - If we can make all the primaries on the SSD
    - Then we get the effect of 100% flash read cache over the HDDs (all reads are from SSD, writes are limited by HDD performance, no cache misses)
  - If making all primaries on SSDs is not possible:
    - Then make as much primaries as possible on SSDs
    - **But** the pool is incorrectly configured
      - Don't mix technologies if you can't always read from the faster devices

Agenda

- ✓ What is optimal cluster performance and why we need it
- ✓ Ceph read balancer (added in Reef)
- ✓ Read balancer future plans
- • **How can we use read balancer infrastructure for dynamic load balancing**

Why Dynamic Load Balancing?

- Systems do not always perform perfectly
  - Hardware problems, network problems and more can cause fluctuations in the performance of nodes or OSDs
  - In hyper converged deployments we may find noisy neighbors that temporarily reduce the performance of some components

What We Suggest

- Monitor Ceph IO performance from OSD level and up
- Identify performance discrepancies in the systems
- Tune the system for optimal performance under the new conditions
- Revert back to normal when performance discrepancies are gone.

This flow is correct for many cloud applications, it is simpler to implement the less state applications have - it is more challenging for storage systems which are stateful by definition.

Dynamic Load Balancing - Option 1

- A well known solution: "the power of 2"
  - Before you send a request to a cluster, select 2 random targets for the request, and send to the less loaded target
    - This can be implemented in Ceph with the read-from-replica feature
    - Requires changes in the clients, in the data path
    - The clients should be updated with data about the load on each OSD in the cluster
    - A similar approach was discussed during the design of the read balancer and was rejected due to the higher risk in implementation vs the selected solution.

Dynamic Load Balancing - Option 2

- Monitor the performance of the system (OSDs, Nodes, racks) centrally
- Create a policy that reduces the load on the less performant units
  - The policy function is a small function that calculates configuration - ~50 lines of code
- When the performance changes over some threshold:
  - Notify the operator about the performance change (in case this is non temporary phenomena that should be fixed, e.g. hardware problem)
  - Change the primary settings according to the policy
- Continue monitoring the system, when performance irregularity disappears system can be recovered to previous configuration.

Conclusion

Power of 2	Read balancer
✗ Added code in the data path	✔ outside of the data path
✗ Need to sync load data with clients continuously	✔ External metadata configuration, based on existing infrastructure
✗ Requires client change (for all clients)	✔ Server side change, client agnostic
✗ **Complex and risky**	✔ **Simpler, less risky, controlled**

Acknowledgements:

Orit Wasserman, distinguished engineer, IBM

Laura Flores, software engineer, IBM

Red Hat

Thank you
Questions?

linkedin.com/company/red-hat

youtube.com/user/RedHatVideos

facebook.com/redhatinc

twitter.com/RedHat

Download this presentation from https://github.com/JoshSalomon/FOSDEM-2023

Red Hat