Configuring Small Ceph Clusters for Optimal Performance

Josh Salomon
Senior Principal Software Engineer
19 May 2019
jsalomon@redhat.com
Motivation - What is Special in Small Clusters

- Ceph is built for scale and works great in large clusters
  - For small clusters default parameters are less than optimal
- Balancing the PGs is partially statistical
  - Works better for larger numbers, smaller clusters can be less than optimally balanced
- There are new use cases and workloads for smaller clusters in clouded environments
  - There is no need to compromise the performance!
  - Small clusters can be closely monitored - use it.
- In a small cluster each node takes big part of the cluster work
- This presentation is based on tests with the Mimic version
  - Environments included bare metal, virtualized and containerized Ceph deployments.
The diagram shows the default installation:

- 3 nodes
- 4 HDDs per node
- 40Gb NICs in the cluster
- Cosbench client connected to the cluster network
- 1 RGW handling the load

Tested bare metal and virtualized deployments
The Importance of Balancing

Distributed systems are vulnerable to the weakest-link-in-the-chain phenomena under some circumstances.

- Overloading a component has the same effect as slow component
- Examples (HDD performance terminology):
  - 99 devices with 100 IOPS, 1 device with 50 IOPS - total 5,000 IOPS
  - 99 devices with 100 IOPS, 1 device with 200 IOPS - total 10,000 IOPS
  - 99 devices with 400GB data, 1 device with 800 GB data - total 5,000 IOPS
  - 99 devices with 800GB data, 1 device with 400 GB data - total 9,950 IOPS
Balancing the Capacity

Placement Group (PG) balancing

- Capacity is distributed among the devices via the PGs
  - Capacity is divided to power-of-2 chunks and assigned to PGs
  - PGs distribution and balancing is performed per pool

- Balancing rule 1 - same number of PGs per OSD
- Balancing rule 2 - devices should have the same utilization
- Balancing rule 3 - same number of primary PGs per OSD

Conclusions:
  - Number of PGs should be a power of 2 (or middle between powers of 2)
  - Small number of larger device(s) in the pool is bad for the global pool performance
  - Small number of smaller devices have only local effect (one device is working less)
  - On heavy read loads, need to check primary OSD balancing

Contradicting!
(devices in different sizes)
Not implemented yet!
Ceph balancing tools
Balancing Tools in Ceph

- Balancer
- OSD CRUSH reweight
- PG autoscaler
- Change number of PGs
Ceph Balancer
Automatic Mode

- The *balancer* plugin can optimize the placement of PGs across OSDs in order to achieve a balanced distribution
- Enable the plugin (on by default)
- Enabling the balancer
  
  You must set the mode!

- Modes:
  - *crush-compat* - for clients older than Luminous
  - *upmap* - for newer clients. Should achieve “perfect” (± 1) PG mapping in all cases

```bash
  ceph mgr module enable balancer
  ceph balancer on
  ceph balancer mode <mode>
```
Ceph Balancer

Supervised Mode

- Evaluate current distribution
- Evaluate specific pool distribution
- Create a plan (using current mode)
- Evaluate the predicted distribution
- Execute the plan

```
ceph balancer eval
ceph balancer eval <pool-name>
ceph balancer optimize <plan> {<pool>}
ceph balancer eval <plan-name>
ceph balancer execute <plan-name>
```
CRUSH Reweight

Pre-Balancer mechanism

- Reweight is a mechanism that overrides the CRUSH default assignments of capacity to OSDs in cases where normal capacity distribution is not optimal
  - Good for changing very small number of over utilized ODSs, less practical for larger number of OSDs
  - Nautilus (or later revisions of Luminous and Mimic) with no pre-Luminous clients should use the balancer instead of this mechanism

- Single OSD reweight
- Reweight all the OSDs based on the existing OSD utilization

```
ceph osd reweight {osd-num} {weight}
ceph osd reweight-by-utilization {params}
ceph osd test-reevaluate-by-utilization {}  
```
PG Autoscaler
Nautilus feature

- A new tool for automatically set the number of PGs per pool
  - Picking the optimal number of PGs per pool is tricky, in the past this number could be increased but not decreased

- Nautilus new features
  - Pg_num can be decreased
  - Pg_num can be tuned in the background
    - Based on data per pool
    - Auto scale (on) and warning (warn) modes

```
ceph mgr module enable pg_autoscaler
ceph osd pool set {pool-name} pg_autoscale_mode {mode}
```

Tip: Be careful of the automatic mode when you need max performance
### PG Autoscaler

**Advanced commands**

- Set the PGs among the pools proportionally by pool size
- Number of PGs / OSD is a ceph configuration item (default to 100)
- Users can set the expected size of a pool in bytes or percentage to prevent many changes when the pool is first filled with data
- User can set the minimal number of PGs per pool
- User can set number of PGs / pool on pool creation time

```bash
ceph config set global mon_target_pg_per_osd <100>
ceph osd pool set <pool name> target_size_bytes <100T>
ceph osd pool set <pool name> target_size_ratio <.9>
ceph osd pool set <pool name> pg_num_min <256>
ceph osd pool create <pool name> pg_num <512>
```
PG Autoscaler

More information

- System is rather conservative and makes changes only when the ratio between number of PGs and what it thinks the number should be > 3
- More information can be found in: http://docs.ceph.com/docs/master/rados/operations/placement-groups/
Change Number of PGS

- Add more PGs
- As of Nautilus - reduce number of PGs!
- PG number should be a power of 2, or in the middle between powers of 2
- Moves a lot of data (up to 40%)
- Adding PGs usually improves the balancing
- Always set pg_num and pgp_num together

```
ceph osd pool set <pool> pg_num <num>
ceph osd pool set <pool> pgp_num <num>
```
Checking the cluster balance
Some commands show interesting balancing information
- `ceph -s` shows generic cluster health information including PGs status
- `ceph osd df` shows the standard deviation of the current balance distribution

No existing tool shows the primary PGs balancing

---

### Checking the Cluster Balance

#### Existing tools

<table>
<thead>
<tr>
<th>ID</th>
<th>CLASS</th>
<th>WEIGHT</th>
<th>REWEIGHT</th>
<th>SIZE</th>
<th>USE</th>
<th>AVAIL</th>
<th>%USE</th>
<th>VAR</th>
<th>PGS</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>hdd</td>
<td>1.81720</td>
<td>1.00000</td>
<td>1.82TiB</td>
<td>336GiB</td>
<td>1.49TiB</td>
<td>18.03</td>
<td>0.94</td>
<td>197</td>
</tr>
<tr>
<td>10</td>
<td>hdd</td>
<td>1.81720</td>
<td>1.00000</td>
<td>1.82TiB</td>
<td>268GiB</td>
<td>1.56TiB</td>
<td>14.41</td>
<td>0.75</td>
<td>191</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

MIN/MAX VAR: 0.72/1.28  STDDEV: 3.51
A useful script for showing the primary OSD balancing can be found in

- 512 PGs: (STDDEV: 0.84)
  # PRIMARY PGs number per OSD sorted descending:
  osd.11  PGs number:  53
  <... Lines removed here ...>
  osd.1   PGs number:  35

  # PRIMARY and Replicated/EC PGs number per OSD sorted by total descending:
  osd.5   PGs:  49  46  35 ,total= 138
  <... Lines removed here ...>
  osd.10  PGs:  40  37  42 ,total= 119

  Primary: Min=12 Max=31 Ratio=2.58333 Average=21.3333 Max/Ave=1.24219
Some test results
Performance results

Containerized ceph on 3 nodes, 4 HDDs per node, single RGW

100% read load, large objects (50% 200MB, 50% 1GB)

- 128 PGs (Stddev: 3.51, Max/Ave: 1.875)
  - Bandwidth: 820 MB/s
- 256 PGs (Stddev: 1.68, Max/Ave 1.45)
  - Bandwidth: 928 MB/s
- 512 PGs (Stddev 0.84, Max/Ave 1.25)
  - Bandwidth: 948 MB/s
- 1024 PGs (Stddev 0.13, Max/Ave 1.16)
  - Bandwidth: 925 MB/s !!

- Observation - we reached another bottleneck!
Next suspect was RGW, so what can we do?

- Add CPU quota to the container
- Add memory quota to the container
- Enlarge the objecter_* attributes (requires more CPU and memory)
  - In the [global] section of ceph.conf add:
    
    ```
    objecter_inflight_ops = 5120
    objecter_inflight_op_bytes = 524288000 (512 * 1,024,000)
    ## You can multiply / divide both with the same factor
    ```

- Result: 512 PGs (Stddev 0.75, Max/Ave 1.31) - Bandwidth 1250 MB/s
  - 31% improvement to 948 MB/s
  - Needed ≈7.5 GB memory for RGW to avoid swaps
For small clusters use relatively more PGs than for large clusters
Check your balance
Look at other bottlenecks (CPU, network)
Beware of swapping!
In some cases consider re-architecting the cluster for QoS reasons
If you use Nautilus server and clients - check the new tools!
Thank you Questions?

linkedin.com/company/red-hat
facebook.com/redhatinc
youtube.com/user/RedHatVideos
twitter.com/RedHat