Ceph中国社区

IT大咖说 知识共享平台

CEPHALOCON APAC 2018

THE FUTURE OF STORAGE

22-23 March 2018 | BEIJING

Common Support Issues and How to Troubleshoot Them

Michael Hackett and Vikhyat Umrao
PSMEs
Red Hat, Inc.

ceph

What are we going to cover today?

- Slow/Blocked Requests
  - What is a slow request?
  - Possible causes
  - Types of Slow Requests
  - Common Troubleshooting Techniques

- Flapping OSD's when RGW buckets have millions of objects
  - Where to start
  - Possible causes
  - Temporary solutions
  - Permanent Solutions

ceph

The Dreaded Slow Request!!!

```
HEALTH_WARN 30 requests are blocked > 32 sec; 3 osds have slow requests
30 ops are blocked > 268435 sec
1 ops are blocked > 268435 sec on osd.11
1 ops are blocked > 268435 sec on osd.18
28 ops are blocked > 268435 sec on osd.39
3 osds have slow requests
```

```
2015-08-24 13:18:10.024659 osd.1 127.0.0.1:6812/3032 9 : cluster [WRN] 6 slow
requests, 6 included below; oldest blocked for > 61.758455 secs
```

```
2016-07-25 03:44:06.510583 osd.50 [WRN] slow request 30.005692 seconds old,
received at {date-time}: osd_op(client.4240.0:8 benchmark_data_ceph-
1_39426_object7 [write 0~4194304] 0.69848840) v4 currently waiting for subops
from [610]
```

- What is a Slow request?
  - When Ceph detects a request that has taken too long to process it will get flagged as a slow request.
- A Slow request will log against an OSD when it has been unable to service the request in it's *op_queue* queue for 30 seconds or more (default).
  - Configurable via *osd_op_complaint_time*. Default is 30 seconds.
- Will be accompanied by blocked requests.
- Changing the *osd_op_complaint_time* is not recommended as can lead to false reporting issues.

ceph

Possible Causes

With the understanding of what a slow request is, what can cause these to occur?

- Problems with underlying hardware such as disk drives, controllers, hosts (kernel or configuration), racks or networking equipment
- System load
- Improper configurables set on OSD's (*op_threads* set to high)
- Improper balance in the cluster leading to 'hot' OSD's
- Cluster configuration issues (too many/not enough PG's per OSD)
- Cluster under backfill/recovery
- Deep scrubbing
- Compaction or splitting is occurring on the OSD node.

ceph

Types of Common Slow Requests

What types of Slow Requests can you expect to see?

- *Waiting for rw locks*: The op is blocked because there's an in-progress op on the object in question. This op needs to be completed before we can obtain a lock.
- *Waiting for sub ops*: This is triggered when the op is already being processed and we are awaiting replica OSD to commit the op, any local events such as committing to disk will move the op out of this state. Most commonly points to the OSD in question as being over utilized.
- *No flag points reached*: The output when the op has not even been queued for the PG yet. Tends to mean there is a large backup or lack of CPU time.
- *Waiting for degraded object*: This means the OSD knows an object exists but does not have it locally as recovery is still in progress.

ceph

Common Troubleshooting Techniques

How to determine what cause of my issue is?

- Verify the cluster status, that all OSD's are UP/IN state and that all PG's are active+clean state (*ceph -s* or *ceph health detail*).
- Determine if OSD's logging the slow requests share a common piece of HW. A log parsing tool such as the ceph log parser can be used to determine ratio of slow requests logging per host or rack crush leaf.
- Verify disk performance on OSD logging slow requests by utilizing a Linux utility such as *iostat*. Look for high utilization percentage and high r_wait and await values.
- Verify network performance/throughput or any errors seen on NICs or networking equipment. Linux utilities such as *netstat* and *iperf* can be used.
- If using Jumbo Frames verify MTU size matches throughout the environment (*ifconfig* and *ping*) .
- Determine if deep-scrubbing, compaction or splitting was occuring at the time of the slow requests. The OSD logs can be reviewed for these events and correlated to the slow requests.

The Ceph log parser

Where do I get this handy tool?

https://github.com/linuxkidd/ceph-log-parsers

Example output of ceph log parser in spreadsheet:

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
535	Pool ID	Deep-Scrub: Average	Deep-Scrub: Count	Deep-Scrub: Max	Deep-Scrub: Min	Deep-Scrub: Total	Slow Primary	Slow Primary: commit_sent	Slow Primary: no flag points reached	Slow Primary: reached_pg	Slow Primary: started	Slow Primary: waiting for degraded object	Slow Primary: waiting for missing object	Slow Primary: waiting for ondisk	Slow Primary: waiting for rw locks	Slow Primary: waiting for scrub	Slow SubOp	Slow Total
536	1	289.468405	40	507.328	242.775	11578.7												
537	2	0.137212	3	0.216364	0.035713	0.411637												
538	3	0.036633	3	0.059088	0.0110841	0.1099												
539	4	0.080645	3	0.115252	0.0404911	0.241935	221		2	211	1	7						221
540	5	0.159889	2	0.2046	0.115178	0.319778												
541	6	0.065334	4	0.109835	0.029923	0.261337												
542	7	0.029934	6	0.061202	0.00389504	0.179606												
543	8	0.054197	3	0.0712051	0.02034	0.162592												
544	9	209.420344	5061	1365.56	11.3129	1059880	455401	4806	50870	274801	441	987	246	2	123226	22	69519	524920
545	11	16.549503	2698	100.63	0.613281	44650.6	2	2									16	18
546	12	0.119071	2694	27.8345	0.00160193	320.777	3		3									3
547	13	0.130721	3	0.267788	0.0144391	0.392163												
548																		
549																		

ceph

```
# ceph -w | grep osds
2017-04-05 06:27:20.810535 mon.0 [INF] osdmap e609: 9 osds: 8 up, 9 in
2017-04-05 06:27:24.120611 mon.0 [INF] osdmap e611: 9 osds: 7 up, 9 in
2017-04-05 06:27:25.975622 mon.0 [INF] HEALTH_WARN; 118 pgs stale; 2/9 in osds
are down
2017-04-05 06:27:27.489790 mon.0 [INF] osdmap e614: 9 osds: 6 up, 9 in
2017-04-05 06:27:36.540000 mon.0 [INF] osdmap e616: 9 osds: 7 up, 9 in
2017-04-05 06:27:39.681913 mon.0 [INF] osdmap e618: 9 osds: 8 up, 9 in
2017-04-05 06:27:43.269401 mon.0 [INF] osdmap e620: 9 osds: 9 up, 9 in
2017-04-05 06:27:54.884426 mon.0 [INF] osdmap e622: 9 osds: 8 up, 9 in
2017-04-05 06:27:57.398706 mon.0 [INF] osdmap e624: 9 osds: 7 up, 9 in
2017-04-05 06:27:59.669841 mon.0 [INF] osdmap e625: 9 osds: 6 up, 9 in
2017-04-05 06:28:07.043677 mon.0 [INF] osdmap e628: 9 osds: 7 up, 9 in
2017-04-05 06:28:10.512331 mon.0 [INF] osdmap e630: 9 osds: 8 up, 9 in
2017-04-05 06:28:12.670923 mon.0 [INF] osdmap e631: 9 osds: 9 up, 9 in
```

Flapping OSD's

- OSD's use the cluster network to send heartbeats to each-other to indicate status (UP/IN). If an OSD is unable to receive this heartbeat a peer OSD will flag as down to the monitor.
- By default, two Ceph OSD Daemons from different hosts must report to the Ceph Monitors that another Ceph OSD Daemon is down before the Ceph Monitors acknowledge that the reported Ceph OSD Daemon is down. This is controlled by this tunable - mon_osd_min_down_reporters.

Flapping OSD's when RGW buckets have millions of objects

- Where to start
  - Identifying type of workload.
    - Millions of objects it means definitely loads of PUTs but if in parallel good amount of DELETEs this is a workload which can cause this issue.
  - 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.

Flapping OSD's when RGW buckets have millions of objects

- Possible causes
  - The first issue here is when RGW buckets have millions of objects their bucket index shard RADOS objects become very large with high number OMAP keys stored in leveldb. Then operations like deep-scrub, bucket index listing etc takes a lot of time to complete and this triggers OSD's to flap. If sharding is not used this issue become worse because then only one RADOS index objects will be holding all the OMAP keys.

  - The second issue is when you have good amount of DELETEs it causes loads of stale data in OMAP and this triggers leveldb compaction all the time which is single threaded and non optimal with this kind of workload and causes osd_op_threads to suicide because it is always compacting hence OSD’s starts flapping.

Flapping OSD's when RGW buckets have millions of objects

- Possible causes contd ...
  - OMAP backend is leveldb in jewel and older clusters. Any luminous clusters which were upgraded from older releases have leveldb as OMAP backend.
  - All new luminous clusters have default OMAP backend as rocksdb which is great because rocksdb has multithreaded compaction and in Ceph we use 8 compaction thread by default and many other enhanced features as compare to leveldb.

Flapping OSD's when RGW buckets have millions of objects

- Temporary solutions
  - The first temporary action should be setting nodeep-scrub flag either global in the cluster with ceph osd set nodeep-scrub or only to the RGW index pool with - ceph osd pool set <pool-name> nodeep-scrub 1.
  - Then the second temporary step could be taken if OSD's are not stopping from hitting suicide timeout. Increase the OSD op threads normal timeout and suicide timeout values and if filestore op threads are also hitting timeout then increase normal and suicide timeout for filestore op threads.

Flapping OSD's when RGW buckets have millions of objects

- Temporary solutions contd ...
  - Add these options in [osd.id] section or in [osd] section to make them permanent till the time troubleshooting of this issue is going on and use ceph tell injectargs command to inject them at run time.

    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

Flapping OSD's when RGW buckets have millions of objects

- Temporary solutions contd ...
  - The third temporary step could be taken if OSD's have very large OMAP directories you can verify it with command: du -sh /var/lib/ceph/osd/ceph-$id/current/omap, then do manual leveldb compaction for OSD's.
    - ceph tell osd.$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 and then bring the OSD back up/in which really helps.

Flapping OSD's when RGW buckets have millions of objects

- Permanent Solutions
  - Calculate the bucket index shard RADOS object size
    - Count the OMAP keys in index shard object
      - rados -p <rgw index pool name> listomapkeys <index-shard-object-name> | 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 very big like above 20 MB consider resharding because shard count is not set as per recommendation or sharding is not used at all.
    - radosgw-admin bucket reshard is the command more details can be found in upstream documentation. This is offline reshard tool.
    - Because of these issues now Luminous has dynamic resharding.
      - http://docs.ceph.com/docs/master/radosgw/dynamicresharding

Flapping OSD's when RGW buckets have millions of objects

- Permanent Solutions contd ...
  - If RGW index pool is not backed by SSD or NVME OSD's and OSD's are running above 80% disk util(Disk bound) during leveldb compaction consider migrating Index pool to new CRUSH ruleset which is backed by SSD or NVME SSD's.
  - If RGW index pool OSD's are always using above 100% CPU(CPU bound) during leveldb compaction consider converting omap backend to rocksdb from leveldb.
  - Jewel still do not support omap backend as rocksdb - this jewel pull request 18010 will bring the rocksdb support in jewel.

Flapping OSD's when RGW buckets have millions of objects

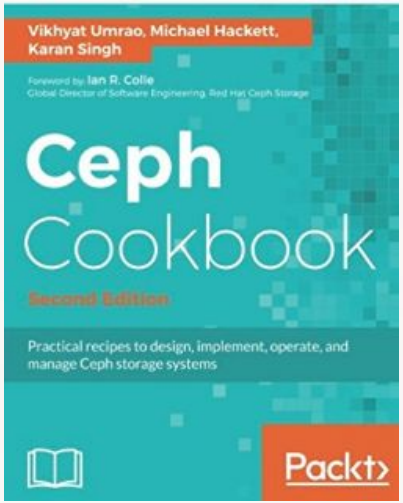
- Permanent Solutions contd ...
  - After rocksdb support in jewel and luminous already has it these commands can be used to convert omap bakend to rocksdb from leveldb:
    - Stop the OSD
    - mv /var/lib/ceph/osd/ceph-<id>/current/omap /var/lib/ceph/osd/ceph-<id>/omap.orig
    - ulimit -n 65535
    - 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 go with above steps then you can rebuild the OSD with filestore_omap_backend = "rocksdb".

Flapping OSD's when RGW buckets have millions of objects

- In summary:
  - Have RGW index pool backed by SSD or NVME.
  - Have proper bucket index shard count set to a nice value from starting considering future growth.
  - Have RGW index pool OSD’s using rocksdb with 8 compaction threads, rocksdb compression disabled and rocksdb_cache_size tuned properly as per your workload starting point 1G and can be increased more.
  - If you still see index pool OSD’s flapping during deep-scrub you can keep nodeep-scrub flag set on the index pool and this luminous pull request luminous: osd: deep-scrub preemption will fix this issue and you can unset nodeep-scrub after upgrading to fixed luminous version.

ceph

Shameless book plug!



[Amazon Link to book]

[Packt Publishing Site]

Ceph中国社区

IT大咖说 知识共享平台

CEPH

CEPHALOCON APAC 2018

THE FUTURE OF STORAGE

22-23 March 2018 | BEIJING

Thank you.