CEPHALOCON APAC 2018

THE FUTURE OF STORAGE

22-23 March 2018 | BEIJING

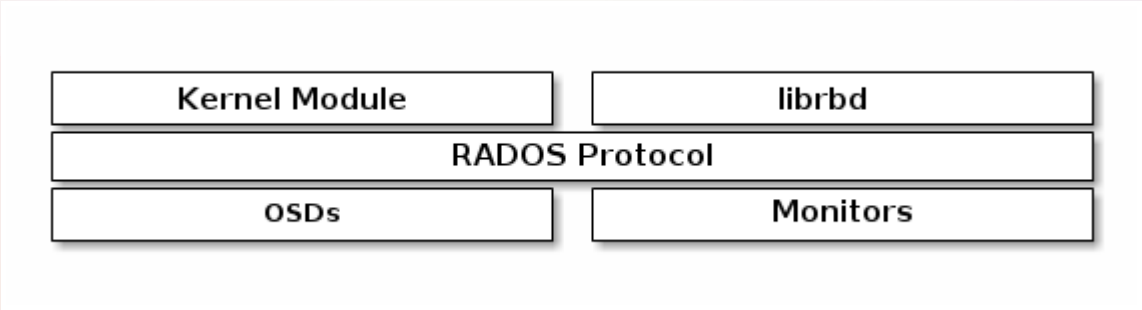
Ceph Client (librbd) Performance Analysis and Learnings

Mahati Chamarthy
Software Engineer, Intel

Agenda

- RBD
- Tracing/profiling tools
- Test environment setup
- Workload configs
- Performance analysis & conclusions

RBD



- RBD images – block devices striped over objects and stored in RADOS object store
- Stripes images across the entire cluster
- Common use-case: disks for VMs

RBD

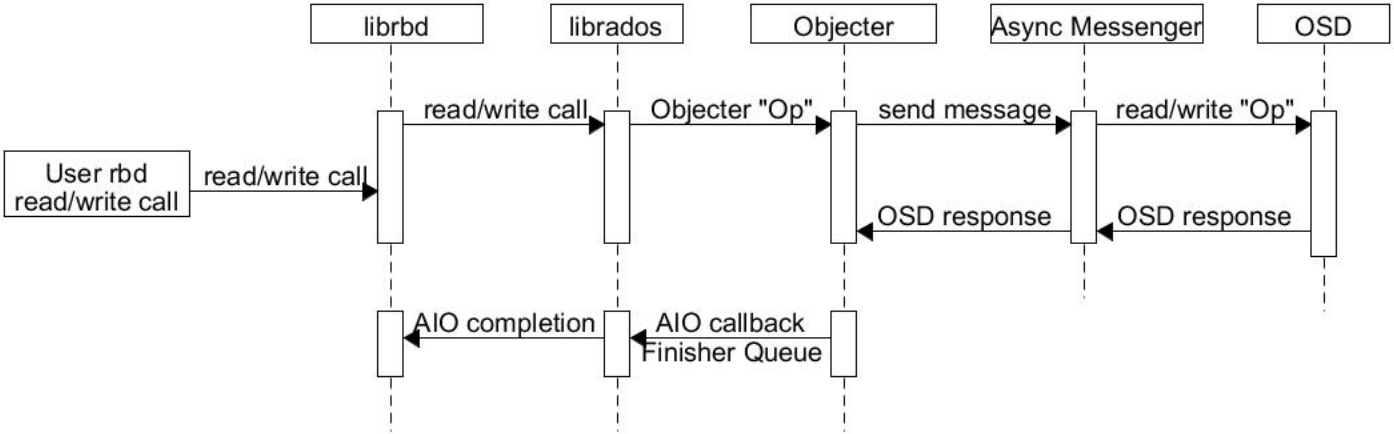
```
mahati@ubuntu:~$ rados -p hello_world_pool ls
rbd_data.1099643c9869.0000000000000000
rbd_directory
rbd_info
rbd_id.librbd_test
rbd_object_map.1099643c9869
rbd_header.1099643c9869
```

- rbd_data.$rbd_id.$fragment
- rbd_directory
- rbd_info
- rbd_id.$rbd_name
- rbd_header.$rbd_id
- rbd_object_map.$rbd_id

RBD Features

- Interacts with OSD via kernel module or librbd
- RBD layering
- Copy-on-write
- Integration with QEMU, libvirt, Linux Kernel
- RBD mirroring

read/write request



Tracing/Profiling tools

- Ceph subsystem logging
- Wireshark/t-shark for analyzing network traffic
- Zipkin/Blkin
- Ceph perf counters along with rbd replay
- Perf tool & flame charts
- Eventtrace

Perf counters

$ rbd-replay-prep ~/lttng-traces/auto-20171122-143523/ust/uid/*/* replay.bin

$ rbd-replay replay.bin --dump-perf-counters

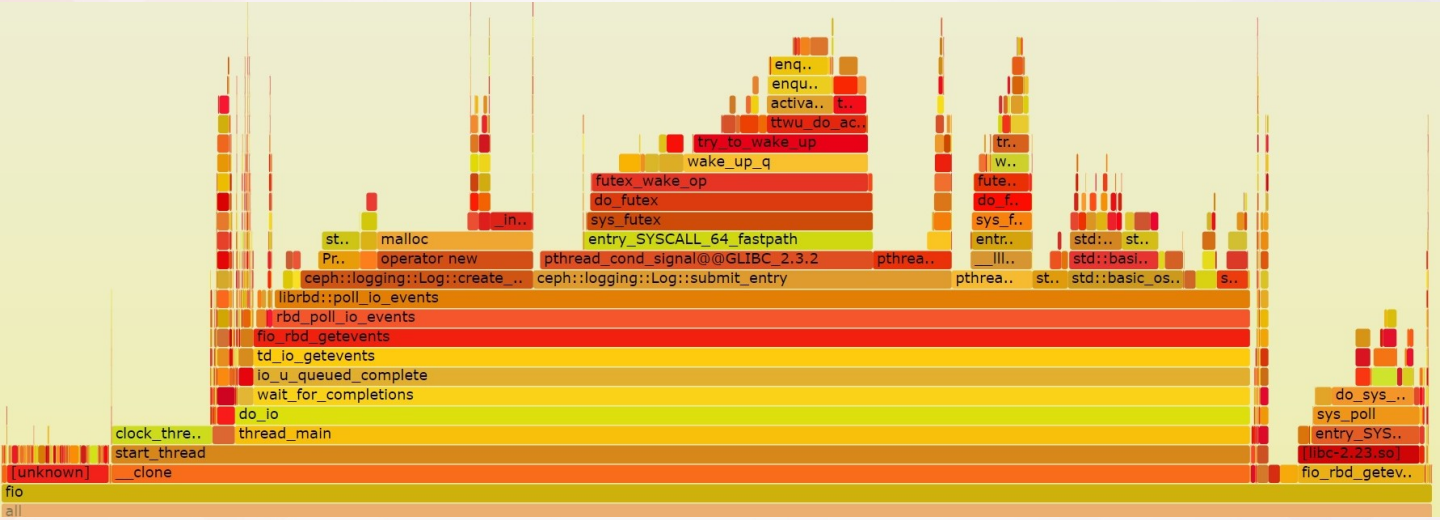
```
{
    "AsyncMessenger::Worker-0": {
        "msgr_recv_messages": 1292
        "msgr_send_messages": 129274,
        "msgr_recv_bytes": 1091071692,
        "msgr_send_bytes": 1100174365,
        "msgr_created_connections": 4,
        "msgr_active_connections": 3,
        "msgr_running_total_time": 16.177110921,
        "msgr_running_send_time": 7.117028038,
        "msgr_running_recv_time": 5.896193864,
        "msgr_running_fast_dispatch_time": 2.522238960
    },
    "finisher-librbd::TaskFinisher::m_finisher": {
        "queue_len": 0,
        "complete_latency": {
            "avgcount": 1,
            "sum": 0.000020664,
            "avgtime": 0.000020664
        }
    },
    "finisher-radosclient": {
        "queue_len": 0,
        "complete_latency": {
            "avgcount": 125096,
            "sum": 2.290013297,
            "avgtime": 0.000018306
        }
```

```
            "avgtime": 0.000000000
        },
        "snap_create": 0,
        "snap_remove": 0,
        "snap_rollback": 0,
        "snap_rename": 0,
        "notify": 0,
        "resize": 0,
        "readahead": 0,
        "readahead_bytes": 0,
        "invalidate_cache": 0,
        "opened_time": 1511362838.582844561,
        "lock_acquired_time": 0.000000000
    },
    "librbd-102a74b0dc51-rbd-fio_test-0x7fc2b805e230": {
        "rd": 0,
        "rd_bytes": 0,
        "rd_latency": {
            "avgcount": 0,
            "sum": 0.000000000,
            "avgtime": 0.000000000
        },
        "wr": 64609,
        "wr_bytes": 1058553856,
        "wr_latency": {
            "avgcount": 64609,
            "sum": 14793.272050858,
            "avgtime": 0.228966120
        },
        "discard": 0,
        "discard_bytes": 0,
        "discard_latency": {
            "avgcount": 0,
            "sum": 0.000000000,
            "avgtime": 0.000000000
        },
        "flush": 0,
        "flush_latency": {
            "avgcount": 0,
            "sum": 0.000000000,
            "avgtime": 0.000000000
        },
```


```
        },
        "discard": 0,
        "discard_bytes": 0,
        "discard_latency": {
            "avgcount": 0,
            "sum": 0.000000000,
            "avgtime": 0.000000000
        },
        "flush": 0,
        "flush_latency": {
            "avgcount": 0,
            "sum": 0.000000000,
            "avgtime": 0.000000000
        },
        "ws": 0,
        "ws_bytes": 0,
        "ws_latency": {
            "avgcount": 0,
            "sum": 0.000000000,
            "avgtime": 0.000000000
        },
        "cmp": 0,
        "cmp_bytes": 0,
        "cmp_latency": {
            "avgcount": 0,
            "sum": 0.000000000,
            "avgtime": 0.000000000
        },
        "snap_create": 0,
        "snap_remove": 0,
        "snap_rollback": 0,
        "snap_rename": 0,
        "notify": 0,
        "resize": 0,
        "readahead": 0,
        "readahead_bytes": 0,
        "invalidate_cache": 0,
        "opened_time": 1511362838.606351951,
        "lock_acquired_time": 1511362838.644545848
    },
    "objecter": {
        "op_active": 0,
        "op_laggy": 0,
```

```
        "op_send": 129248,
        "op_send_bytes": 0,
        "op_resend": 0,
        "op_reply": 129248,
        "op": 129248,
        "op_r": 64597,
        "op_w": 64651,
        "op_rmw": 0,
        "op_pg": 0,
        "osdop_stat": 2,
        "osdop_create": 0,
        "osdop_read": 64574,
        "osdop_write": 64609,
        "osdop_writefull": 0,
        "osdop_writesame": 0,
        "osdop_append": 0,
        "osdop_zero": 0,
        "osdop_truncate": 0,
        "osdop_delete": 0,
        "osdop_mapext": 0,
        "osdop_sparse_read": 0,
        "osdop_clonerange": 0,
        "osdop_getxattr": 0,
        "osdop_setxattr": 0,
        "osdop_cmpxattr": 0,
        "osdop_rmxattr": 0,
```

perf flame graph



Eventtrace

- Lttng based
- Ease of use with minimal code changes
- Eventtrace support exists in Ceph code
- Generates large amount of data – needs to be parsed

Usage

```
#include “common/EventTrace.h”

……
FUNCTRACE(cct);
……
```

- cct being CephContext value

Test environment setup

- Ceph cluster with single OSD, single Mon and a manger [1]
- Bluestore as backend
- Separate partitions for wal and db
- SSD P3700 NVMe/PCIe 400GB

[1]https://github.com/MahatiC/ceph-single-osd-mon

Config - I

- Workloads collected on an rbd image of size 1024MB, in a pool with 100 pgs, and **cache disabled**
- Data collected using eventtrace; the values are averaged out using a python script
- Config:
  - random write
  - blocksize: 4k
  - iodepth: 1
  - runtime: 6sec

	A	
7	handle_acquire_lock	
8	process(WRITE_PRE)	169.14
9	send(ImageRequest.cc)	162.71
10	send_request	158.5
11	file_to_extents	10.5
12	set_request_count	1.54
13	send_object_requests	133.91
14	create_object_request	16.19
15	extent_to_file	1.62
16	compute_parent_extents	4.37
17	prune_parent_extents	1.09
18	send(ObjectRequest.cc)	112.58
19	object_may_exist	2.36
20	send_write	105.08
21	send_write	102.49
22	send_pre_object_map_update	99.48
23	detained_aio_update	92.73
24	aio_update	88.71
25	update_object_map	83.21
26	handle_update_object_map	84.6
27	update_in_memory_object_map	3.85
28	should_complete (WRITE_PRE)	67.82
29	send_write_op	64.21

	A	
30	add_write_ops	
31	process(WRITE_FLAT)	139.73
32	send(ImageRequest.cc)	133.09
33	send_request	129.06
34	file_to_extents	9.84
35	set_request_count	1.45
36	send_object_requests	105.51
37	create_object_request	16.28
38	extent_to_file	1.5
39	compute_parent_extents	4.15
40	prune_parent_extents	1.09
41	send(ObjectRequest.cc)	84.33
42	object_may_exist	2.26
43	send_write	77.12
44	send_write	74.65
45	send_pre_object_map_update	71.81
46	send_write_op	66.89
47	add_write_ops	3.58
48	should_complete (WRITE_FLAT)	8.11
49	send_post_object_map_update	2.69
50	complete_request	20.23
51	finalize	1.97
52	complete	8.89

Config - II

- Workloads collected on an rbd image of size 1024MB, in a pool with 100 pgs, and **cache disabled**
- Data collected using eventtrace; the values are averaged out using a python script
- Config:
  - random read
  - blocksize: 4k
  - iodepth: 1
  - runtime: 6sec

	func	
2	send_message	
3	write_message	26.8
4	ms_dispatch	8.5
5	aio_read	3.8
6	queue	2.16
7	process(WRITE_PRE)	61.92
8	send(ImageRequest.cc)	58.65
9	send_request	55.77
10	file_to_extents	4.57
11	set_request_count	1.05
12	extent_to_file	1.11
13	compute_parent_extents	3.32
14	prune_parent_extents	0.88
15	send(ObjectRequest.cc)	34.04
16	should_complete	1.26
17	finish	17.55
18	add_partial_sparse_result	1.47
19	complete_request	11.91
20	finalize	1.61
21	complete	4.91

Config - III

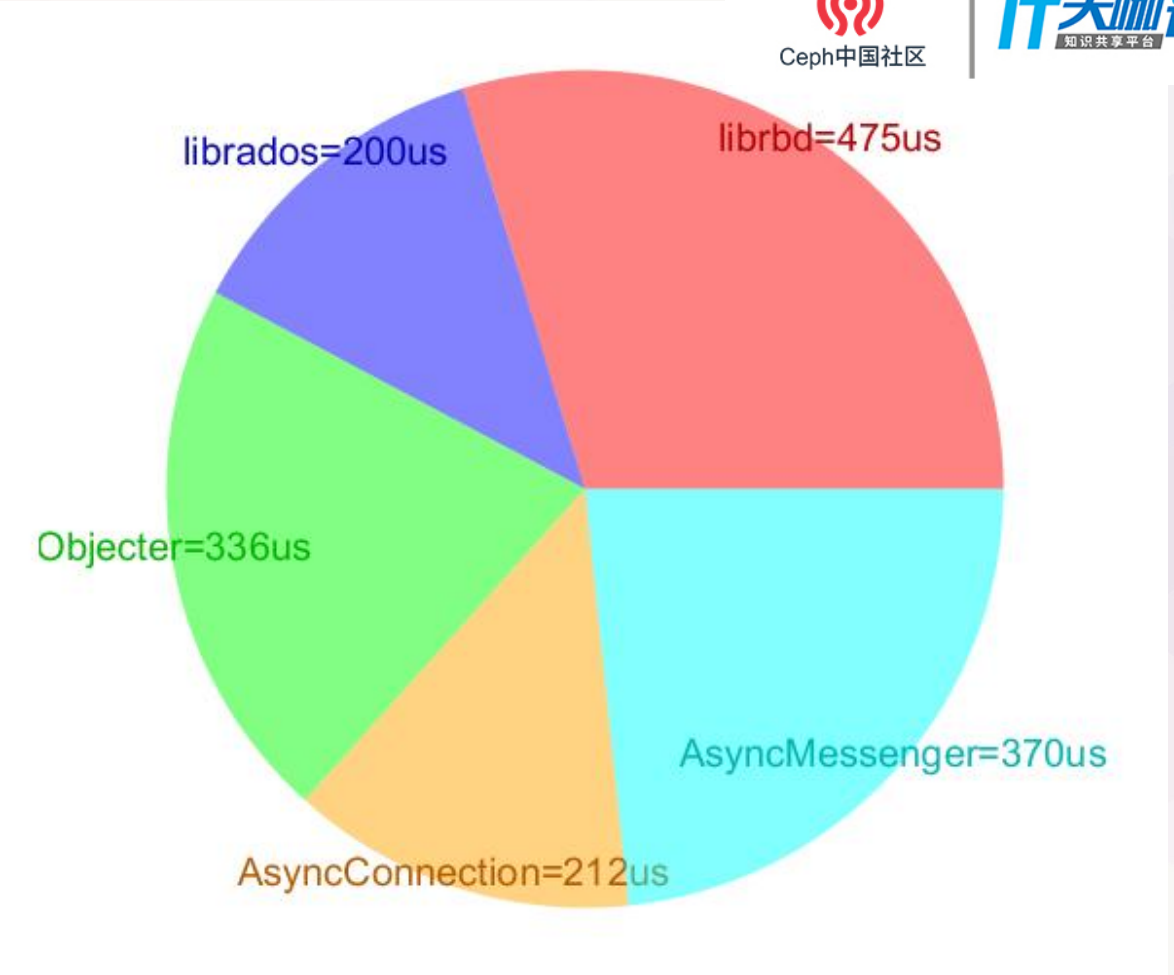
- Workloads collected on an rbd image of size 1024MB, in a pool with 100 pgs, and **cache disabled**
- Data collected using eventtrace; the values are averaged out using a python script
- Config:
  - random write
  - blocksize: 16k
  - iodepth: 1
  - runtime: 6sec

	A	
29	handle_acquire_lock	
30	process(WRITE_PRE)	163.11
31	send(ImageRequest.cc)	156.75
32	send_request	153.03
33	file_to_extents	10
34	set_request_count	1.57
35	send_object_requests	128.98
36	create_object_request	15.27
37	extent_to_file	1.92
38	compute_parent_extents	4.09
39	prune_parent_extents	1.06
40	send(ObjectRequest.cc)	108.45
41	object_may_exist	2.23
42	send_write	101.2
43	send_write	98.62
44	send_pre_object_map_update	95.68
45	detained_aio_update	89.47
46	aio_update	85.76
47	update_object_map	80.57
48	process(WRITE_FLAT)	138.68
49	send(ImageRequest.cc)	132.42
58	send(ObjectRequest.cc)	84.65
65	handle_update_object_map	98.88

Config - IV

- Workloads collected on an rbd image of size 1024MB, in a pool with 100 pgs, and **cache disabled**
- Data collected using eventtrace; the values are averaged out using a python script
- Config:
  - random read
  - blocksize: 16k
  - iodepth: 1
  - runtime: 6sec

	A	
21	aio_operate	
22	op_submit	78.75
23	_calc_target	13.5
24	_prepare_osd_op	6.75
25	_send_op	41
26	send_message	37
27	aio_read	4.37
28	queue	2.1
29	process	61.33
30	send(ImageRequest.cc)	58.03
31	send_request	55.17
32	file_to_extents(Striper)	4.43
33	set_request_count	0.98
34	extent_to_file	1.08
35	compute_parent_extents	3.32
36	prune_parent_extents	0.88
37	send(ObjectRequest.cc)	33.68
38	should_complete	1.53
39	finish	17.72
40	add_partial_sparse_result	1.93
41	complete_request	12.19
42	finalize	2.32
43	complete	4.56


librados=200us
librbd=475us
Objecter=336us
AsyncMessenger=370us
AsyncConnection=212us

Thank you!