rbd - Bug #4531

ObjectCacher: read waiters for parent data during copyup get reordered, causing the write order assert to fail

03/22/2013 10:01 AM - Josh Durgin

Status:	Resolved	**% Done:**	0%
Priority:	Urgent	**Spent time:**	0.00 hour
Assignee:	Josh Durgin		
Category:			
Target version:	v0.61 - Cuttlefish		
Source:	Community (user)	**Reviewed:**	
Tags:		**Affected Versions:**	
Backport:	bobtail	**ceph-qa-suite:**	
Regression:	No	**Pull request ID:**	
Severity:	2 - major	**Crash signature:**	

Description

This assert was seen by several users using clones:

```
osdc/ObjectCacher.cc: In function 'void
ObjectCacher::bh_write_commit(int64_t, sobject_t, loff_t, uint64_t,
tid_t, int)' thread 7f376133f700 time 2013-03-21 14:17:47.517280
osdc/ObjectCacher.cc: 834: FAILED assert(ob->last_commit_tid < tid)
 ceph version 0.56.3-42-ga30903c (a30903c6adaa023587d3147179d6038ad37ca520)
 1: (ObjectCacher::bh_write_commit(long, sobject_t, long, unsigned
long, unsigned long, int)+0xd68) [0x7f376fdbda48]
 2: (ObjectCacher::C_WriteCommit::finish(int)+0x6b) [0x7f376fdc460b]
 3: (Context::complete(int)+0xa) [0x7f376fd7c9fa]
 4: (librbd::C_Request::finish(int)+0x85) [0x7f376fdac315]
 5: (Context::complete(int)+0xa) [0x7f376fd7c9fa]
 6: (librbd::rados_req_cb(void*, void*)+0x47) [0x7f376fd91387]
 7: (librados::C_AioSafe::finish(int)+0x1d) [0x7f376f14163d]
 8: (Finisher::finisher_thread_entry()+0x1c0) [0x7f376f1ac920]
 9: (()+0x7e9a) [0x7f376ca0fe9a]
 10: (clone()+0x6d) [0x7f376c73bcbd]
```

Travis Rhoden supplied detailed logs of this happening, and they show several writes to the same object that need to do copyup.

Originally everything tries to read from the parent and waits on the 0~6172672, since that's the only missing piece:

```
2013-03-21 14:17:46.956737 7f376133f700 20 librbd: aio_read 0x7f3772d59420 completion 0x7f370002a0
10 [134217728,8388608]
2013-03-21 14:17:46.956740 7f376133f700 20 librbd: ictx_check 0x7f3772d59420
2013-03-21 14:17:46.956752 7f376133f700 20 librbd:  oid rbd_data.1e3512c0381.0000000000000010 0~83
88608 from [0,8388608]
2013-03-21 14:17:46.956768 7f376133f700 10 objectcacher readx extent(rbd_data.1e3512c0381.00000000
00000010 (0) in @4 0~8388608 -> [0,8388608])
2013-03-21 14:17:46.956777 7f376133f700 10 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rbd_data.1e3512c0381.0000000000000010 0~8388608
2013-03-21 14:17:46.956782 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f3700047c40 0~6172672 0x7f370000b4e0 (0) v 0 rx] waiters = { 0->[0x7f370
0048630, 0x7f370003fbe0, 0x7f3700042040, 0x7f3700032c00, 0x7f3700049f90, ]}
2013-03-21 14:17:46.956795 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read hit bh[ 0x7f37000280f0 6172672~4096 0x7f370000b4e0 (4096) v 0 clean firstbyte=20]
waiters = {}
2013-03-21 14:17:46.956807 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
```

```
10/19) map_read rx bh[ 0x7f370003e1e0 6176768~2043904 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:46.956814 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read hit bh[ 0x7f3700019b30 8220672~4096 0x7f370000b4e0 (4096) v 0 clean firstbyte=8] w
aiters = {}
2013-03-21 14:17:46.956822 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370003e5e0 8224768~8192 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:46.956829 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read hit bh[ 0x7f3772d5dcc0 8232960~4096 0x7f370000b4e0 (4096) v 0 clean firstbyte=11]
waiters = {}
2013-03-21 14:17:46.956882 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370003e9e0 8237056~28672 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:46.956892 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read hit bh[ 0x7f3772ee6b00 8265728~4096 0x7f370000b4e0 (4096) v 0 clean firstbyte=20]
waiters = {}
2013-03-21 14:17:46.956901 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370003ee60 8269824~8192 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:46.956908 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read hit bh[ 0x7f37000142a0 8278016~4096 0x7f370000b4e0 (4096) v 0 clean firstbyte=23]
waiters = {}
2013-03-21 14:17:46.956916 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370002f770 8282112~106496 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:46.956922 7f376133f700 10 objectcacher readx missed, waiting on bh[ 0x7f3700047c4
0 0~6172672 0x7f370000b4e0 (0) v 0 rx] waiters = { 0->[0x7f3700048630, 0x7f370003fbe0, 0x7f3700042
040, 0x7f3700032c00, 0x7f3700049f90, ]} off 0
```

But then the cache is trimmed:

```
2013-03-21 14:17:47.066713 7f376133f700 10 objectcacher trim trimming bh[ 0x7f37000142a0 8278016~4
096 0x7f370000b4e0 (4096) v 0 clean firstbyte=23] waiters = {}
```

And the first waiter retries their read, and waits for this new gap to be filled:

```
2013-03-21 14:17:47.075919 7f376133f700 20 objectcacher finishing waiters 0x7f3700048630,0x7f37000
3fbe0,0x7f3700042040,0x7f3700032c00,0x7f3700049f90,0x7f3700026580
2013-03-21 14:17:47.075926 7f376133f700 10 objectcacher readx extent(rbd_data.1e3512c0381.00000000
00000010 (0) in @4 0~8388608 -> [0,8388608])
2013-03-21 14:17:47.075935 7f376133f700 10 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rbd_data.1e3512c0381.0000000000000010 0~8388608
2013-03-21 14:17:47.075942 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read hit bh[ 0x7f3700047c40 0~6176768 0x7f370000b4e0 (6176768) v 0 clean firstbyte=0] w
aiters =
{}
2013-03-21 14:17:47.075950 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370003e1e0 6176768~2043904 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:47.075956 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read hit bh[ 0x7f3700019b30 8220672~4096 0x7f370000b4e0 (4096) v 0 clean firstbyte=8] w
aiters =
{}
2013-03-21 14:17:47.075966 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370003e5e0 8224768~8192 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:47.075975 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read hit bh[ 0x7f3772d5dcc0 8232960~4096 0x7f370000b4e0 (4096) v 0 clean firstbyte=11]
waiters =
 {}
2013-03-21 14:17:47.075983 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370003e9e0 8237056~28672 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:47.075989 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read hit bh[ 0x7f3772ee6b00 8265728~4096 0x7f370000b4e0 (4096) v 0 clean firstbyte=20]
waiters = {}
2013-03-21 14:17:47.075997 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370003ee60 8269824~8192 0x7f370000b4e0 (0) v 0 rx] waiters = {}
```

```
2013-03-21 14:17:47.076007 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read gap bh[ 0x7f3700025b60 8278016~4096 0x7f370000b4e0 (0) v 0 missing] waiters = {}
2013-03-21 14:17:47.076013 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370002f770 8282112~106496 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:47.076020 7f376133f700  7 objectcacher bh_read on bh[ 0x7f3700025b60 8278016~4096
 0x7f370000b4e0 (0) v 0 missing] waiters = {}
2013-03-21 14:17:47.076062 7f376133f700 10 client.7872.objecter recalc_op_target tid 2766 pgid 4.e
51252a4 acting [15,38]
2013-03-21 14:17:47.076070 7f376133f700 20 client.7872.objecter  note: not requesting commit
2013-03-21 14:17:47.076075 7f376133f700 10 client.7872.objecter op_submit oid rbd_data.1e3512c0381
.0000000000000010 @4 [read 8278016~4096] tid 2766 osd.15
2013-03-21 14:17:47.076084 7f376133f700 15 client.7872.objecter send_op 2766 to osd.15
2013-03-21 14:17:47.076090 7f376133f700  1 -- 10.10.20.1:0/1060273 --> 10.10.30.2:6803/3748 -- osd
_op(client.7872.0:2766 rbd_data.1e3512c0381.000000000000010@19 [read 8278016~4096] 4.e51252a4) v4
 -- ?+0 0x7f370001c7e0 con 0x7f3772eb4be0
2013-03-21 14:17:47.077056 7f376133f700 20 -- 10.10.20.1:0/1060273 submit_message osd_op(client.78
72.0:2766 rbd_data.1e3512c0381.000000000000010@19 [read 8278016~4096] 4.e51252a4) v4 remote, 10.1
0.30.2:6803/3748, have pipe.
2013-03-21 14:17:47.077091 7f376133f700  5 client.7872.objecter 58 unacked, 11 uncommitted
2013-03-21 14:17:47.077102 7f376133f700 10 objectcacher readx missed, waiting on bh[ 0x7f3700025b6
0 8278016~4096 0x7f370000b4e0 (0) v 0 rx] waiters = {} off 8278016
2013-03-21 14:17:47.077115 7f376133f700 20 objectcacher readx defer 0x7f3700036300
```

The second and subsequent waiters see no gaps to fill, so they just wait on the first rx buffer, which was sent before the first waiter's new request:

```
2013-03-21 14:17:47.077122 7f376133f700 10 objectcacher readx extent(rbd_data.1e3512c0381.00000000
00000010 (0) in @4 0~8388608 -> [0,8388608])
2013-03-21 14:17:47.077130 7f376133f700 10 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rbd_data.1e3512c0381.0000000000000010 0~8388608
2013-03-21 14:17:47.077138 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read hit bh[ 0x7f3700047c40 0~6176768 0x7f370000b4e0 (6176768) v 0 clean firstbyte=0] w
aiters = {}
2013-03-21 14:17:47.077196 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370003e1e0 6176768~2043904 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:47.077208 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read hit bh[ 0x7f3700019b30 8220672~4096 0x7f370000b4e0 (4096) v 0 clean firstbyte=8] w
aiters = {}
2013-03-21 14:17:47.077223 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370003e5e0 8224768~8192 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:47.077235 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read hit bh[ 0x7f3772d5dcc0 8232960~4096 0x7f370000b4e0 (4096) v 0 clean firstbyte=11]
waiters = {}
2013-03-21 14:17:47.077245 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370003e9e0 8237056~28672 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:47.077252 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read hit bh[ 0x7f3772ee6b00 8265728~4096 0x7f370000b4e0 (4096) v 0 clean firstbyte=20]
waiters = {}
2013-03-21 14:17:47.077259 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370003ee60 8269824~8192 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:47.077266 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f3700025b60 8278016~4096 0x7f370000b4e0 (0) v 0 rx] waiters = { 8278016-
>[0x7f370002eb40, ]}
2013-03-21 14:17:47.077275 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.00000000000000
10/19) map_read rx bh[ 0x7f370002f770 8282112~106496 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:47.077281 7f376133f700 10 objectcacher readx missed, waiting on bh[ 0x7f370003e1e
0 6176768~2043904 0x7f370000b4e0 (0) v 0 rx] waiters = {} off 6176768
2013-03-21 14:17:47.077290 7f376133f700 20 objectcacher readx defer 0x7f3700046470
```

By the time this read with the last 5 waiters completes, more sections of the object have been trimmed:

```
2013-03-21 14:17:47.107025 7f376133f700 10 objectcacher.object(rbd_data.1e3512c0381.0000000000000
10/19) map_read rbd_data.1e3512c0381.0000000000000010 0~8388608
2013-03-21 14:17:47.107032 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.0000000000000
10/19) map_read hit bh[ 0x7f3700047c40 0~8220672 0x7f370000b4e0 (8220672) v 0 clean firstbyte=0] w
aiters = {}
2013-03-21 14:17:47.107041 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.0000000000000
10/19) map_read gap bh[ 0x7f370005ed10 8220672~4096 0x7f370000b4e0 (0) v 0 missing] waiters = {}
2013-03-21 14:17:47.107048 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.0000000000000
10/19) map_read rx bh[ 0x7f370003e5e0 8224768~8192 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:47.107056 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.0000000000000
10/19) map_read gap bh[ 0x7f37000169c0 8232960~4096 0x7f370000b4e0 (0) v 0 missing] waiters = {}
2013-03-21 14:17:47.107063 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.0000000000000
10/19) map_read rx bh[ 0x7f370003e9e0 8237056~28672 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:47.107071 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.0000000000000
10/19) map_read gap bh[ 0x7f3700033130 8265728~4096 0x7f370000b4e0 (0) v 0 missing] waiters = {}
2013-03-21 14:17:47.107079 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.0000000000000
10/19) map_read rx bh[ 0x7f370003ee60 8269824~8192 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:47.107089 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.0000000000000
10/19) map_read rx bh[ 0x7f3700025b60 8278016~4096 0x7f370000b4e0 (0) v 0 rx] waiters = { 8278016-
>[0x7f370002eb40, ]}
2013-03-21 14:17:47.107098 7f376133f700 20 objectcacher.object(rbd_data.1e3512c0381.0000000000000
10/19) map_read rx bh[ 0x7f370002f770 8282112~106496 0x7f370000b4e0 (0) v 0 rx] waiters = {}
2013-03-21 14:17:47.107108 7f376133f700  7 objectcacher bh_read on bh[ 0x7f370005ed10 8220672~4096
 0x7f370000b4e0 (0) v 0 missing] waiters = {}
```

Since the first gap is before the first rx buffer this time all the waiters wait for the buffer for 8220672~4096. Next the read the first original waiter was waiting for finished, and it waits on the same buffer as the rest of them, but since it's appending to the list of waiters, it is now last instead of first.

Eventually this leads to the original waiters 2-6 completing first, thereby sending their copyup-and-write operations first, while the first original waiter writes after them, triggering the ordering assert when it completes.

Related issues:

Duplicated by rbd - Bug #4739: Failed assert in librbd with rbd cache enabled	**Duplicate**	**04/16/2013**

History

#1 - 03/22/2013 10:02 AM - Josh Durgin

- Source changed from Development to Community (user)

- Backport set to bobtail

- Severity changed from 3 - minor to 2 - major

#2 - 04/01/2013 11:42 AM - Ian Colle

- Target version set to v0.61 - Cuttlefish

#3 - 04/01/2013 11:56 AM - Josh Durgin

- Status changed from 12 to In Progress

#4 - 04/10/2013 06:29 PM - Josh Durgin

- Status changed from In Progress to 7

wip-objectcacher-handler-ordered

#5 - 04/11/2013 12:33 PM - Josh Durgin

- Status changed from 7 to Fix Under Review

- Assignee changed from Josh Durgin to Dan Mick

Dan, can you look this over? https://github.com/ceph/ceph/pull/214

#6 - 04/12/2013 01:49 PM - Dan Mick

- Assignee deleted (Dan Mick)

I tried, but I just don't know enough about the cacher or the completion framework to understand this, I'm afraid.

#7 - 04/12/2013 03:14 PM - Josh Durgin

- Assignee set to Sage Weil

#8 - 04/16/2013 03:48 PM - Sage Weil

- Status changed from Fix Under Review to Resolved

#9 - 04/16/2013 03:49 PM - Sage Weil

- Status changed from Resolved to Pending Backport

- Assignee changed from Sage Weil to Josh Durgin

#10 - 04/23/2013 12:05 PM - Josh Durgin

- Status changed from Pending Backport to Resolved