Ceph - Bug #592

osd: rebind cluster_messenger when wrongly marked down

11/18/2010 05:03 PM - Colin McCabe

Status:	Resolved	**Start date:**	11/18/2010
Priority:	High	**Due date:**	
Assignee:	Sage Weil	**% Done:**	0%
Category:	OSD	**Estimated time:**	0.00 hour
Target version:	v0.24	**Spent time:**	3.00 hours
Source:		**Reviewed:**	
Tags:		**Affected Versions:**	
Backport:		**ceph-qa-suite:**	
Regression:	No	**Pull request ID:**	
Severity:	3 - minor	**Crash signature:**	

Description

This happened with commit:323565343071ce695f7d454ed29590688de64d5d on flab.ceph.dreamhost.com

While running test_unfound.sh, I noticed a situation where OSD peering never completed.

This happened in stray_test. Basically the sequence of events was like this:

1. brought up all OSDs with vstart.sh

2. Took out osd0 with

    ./ceph osd out 0

3. ran radostool with

    ./rados -p data put obj01 /tmp/tmp.UeAgBcigql/ver1

It blocked forever and never returned.

So what is going on on the backend?
Well, it seems that we are requesting something on PG 0.5. osd2 blocks forever with this message:

```
2010-11-18 15:05:33.161056 7fd48dc51710 osd2 14 _dispatch 0x2d24d80 osd_op(client4113.0:1 obj01 [w
ritefull 0~1000000] 0.92bd) v1
 2010-11-18 15:05:33.161069 7fd48dc51710 osd2 14 require_same_or_newer_map 9 (i am 14) 0x2d24d80
 2010-11-18 15:05:33.161079 7fd48dc51710 osd2 14 _share_map_incoming client4113 10.3.14.10:0/1691
9
 2010-11-18 15:05:33.161088 7fd48dc51710 osd2 14 client4113 has old map 9 < 14
 2010-11-18 15:05:33.161094 7fd48dc51710 osd2 14 send_incremental_map 9 - > 14 to client4113 10.3.
14.10:0/1691
 2010-11-18 15:05:33.161131 7fd48dc51710 filestore(dev/osd2) read dev/osd2/current/meta/inc_osdmap
.14_0 0~116 = 116
 2010-11-18 15:05:33.161152 7fd48dc51710 filestore(dev/osd2) read dev/osd2/current/meta/inc_osdmap
.13_0 0~116 = 116
 2010-11-18 15:05:33.161170 7fd48dc51710 filestore(dev/osd2) read dev/osd2/current/meta/inc_osdmap
.12_0 0~116 = 116
 2010-11-18 15:05:33.161187 7fd48dc51710 filestore(dev/osd2) read dev/osd2/current/meta/inc_osdmap
.11_0 0~116 = 116
 2010-11-18 15:05:33.161214 7fd48dc51710 filestore(dev/osd2) read dev/osd2/current/meta/inc_osdmap
```

```
.10_0 0~424 = 424
 2010-11-18 15:05:33.161224 7fd48dc51710 -- 10.3.14.10:6804/29550 --> client4113 10.3.14.10:0/1691
 -- osd_map(10,14) v1 -- ?+0 0x2e94000
 2010-11-18 15:05:33.161260 7fd48dc51710 osd2 14 hit non-existent pg 0.5, waiting
```

Further inspection reveals that pg 0.5 is stuck in the peering state (see pg_dump.txt).

For PG 0.5, the primary is osd1. The replica is osd2. The stray is osd0.

The primary (osd1) has requested PG info from osd2. However, for some reason, it never seems to get the PG info that it wanted. So the PG stays in its peering state and the user blocks forever.

Logs are under /home/cmccabe/log/osd_peering_doesnt_complete on flab.

Associated revisions

Revision 924b1fcb - 11/22/2010 05:49 PM - Sage Weil

osd: bind to new cluster address when wrongly marked down

If we come back up on the same address, there is a possible race. Other nodes will mark_down when they see us go down. If we go up first, queue some messages, and *then* they see that we're down and mark_down, the messages we queued will get lost. Since it's stateful on the cluster backend, we need to introduce an ordering so that closing out the *old* session doesn't break the new session. We do this by binding to a new address (just a different port, actually) before marking ourselves back up.

Fixes #592.

Signed-off-by: Sage Weil <sage@newdream.net>

History

#1 - 11/18/2010 05:04 PM - Colin McCabe

- File osd_dump.txt added

#2 - 11/18/2010 05:07 PM - Colin McCabe

I should also add that Greg Farnum helped me examine the logs for this bug.

#3 - 11/18/2010 08:42 PM - Sage Weil

i suspect 0.5 didn't get set up on osd1 or 2 before osd0 went down? do you have the full logs for the other instances?

#4 - 11/18/2010 09:22 PM - Sage Weil

- Category set to OSD

- Assignee changed from Colin McCabe to Sage Weil

- Target version set to v0.24

ok, this is a problem with how the osd is interacting with the messenger. looking at the history of 0.5, we see

```
grep -h 0\\.5\( osd.?| sort|less
```

that osd1 clearly queries osd2, but osd2 doesn't respond. comparing

```
grep 10.3.14.10:6804/29550 osd.1
grep 10.3.14.10:6802/29157 osd.2
```

we see that osd1 sends 9look at --> lines)

```
2010-11-18 15:04:59.743555 7fe83b066710 -- 10.3.14.10:6802/29157 --> osd2 10.3.14.10:6804/29550 -- PGq v1 -- ?
+0 0x181b8c0
2010-11-18 15:05:01.931516 7fe83b066710 -- 10.3.14.10:6802/29157 <== osd2 10.3.14.10:6804/29550 1 ==== PGq v1
==== 1472+0+0 (503857468 0 0) 0x181b8c0
2010-11-18 15:05:01.935455 7fe83b066710 -- 10.3.14.10:6802/29157 --> osd2 10.3.14.10:6804/29550 -- PGnot v1 --
 ?+0 0x17b1e00
2010-11-18 15:05:01.935593 7fe83b066710 -- 10.3.14.10:6802/29157 <== osd2 10.3.14.10:6804/29550 2 ==== PGnot v
1 ==== 1240+0+0 (733914456 0 0) 0x1392000
2010-11-18 15:05:12.130370 7fe83b066710 osd1 10 send_incremental_map 6 -> 10 to osd2 10.3.14.10:6804/29550
2010-11-18 15:05:12.130475 7fe83b066710 -- 10.3.14.10:6802/29157 --> osd2 10.3.14.10:6804/29550 -- osd_map(7,1
0) v1 -- ?+0 0x1717400
2010-11-18 15:05:12.130506 7fe83b066710 -- 10.3.14.10:6802/29157 --> osd2 10.3.14.10:6804/29550 -- PGnot v1 --
 ?+0 0x13fdc40
2010-11-18 15:05:12.130596 7fe83b066710 -- 10.3.14.10:6802/29157 --> osd2 10.3.14.10:6804/29550 -- PGq v1 -- ?
+0 0x13c4a80
2010-11-18 15:05:13.191322 7fe835b50710 -- 10.3.14.10:6802/29157 >> 10.3.14.10:6804/29550 pipe(0x181e000 sd=18
 pgs=7 cs=1 l=0).fault with nothing to send, going to standby
2010-11-18 15:05:16.053316 7fe836358710 -- 10.3.14.10:6802/29157 >> 10.3.14.10:6804/29550 pipe(0x181e780 sd=17
 pgs=0 cs=0 l=0).accept connect_seq 0 vs existing 1 state 3
2010-11-18 15:05:16.053340 7fe836358710 -- 10.3.14.10:6802/29157 >> 10.3.14.10:6804/29550 pipe(0x181e780 sd=17
 pgs=0 cs=0 l=0).accept peer reset, then tried to connect to us, replacing
2010-11-18 15:05:16.056617 7fe83b066710 -- 10.3.14.10:6802/29157 <== osd2 10.3.14.10:6804/29550 1 ==== PGnot v
1 ==== 1548+0+0 (3823422094 0 0) 0x182b380
2010-11-18 15:05:16.059076 7fe83b066710 -- 10.3.14.10:6802/29157 <== osd2 10.3.14.10:6804/29550 2 ==== PGq v1
==== 2713+0+0 (3673062997 0 0) 0x1835e00
2010-11-18 15:05:16.069644 7fe83b066710 -- 10.3.14.10:6802/29157 --> osd2 10.3.14.10:6804/29550 -- PGnot v1 --
 ?+0 0x18888c0
2010-11-18 15:05:19.726386 7fe83b066710 -- 10.3.14.10:6802/29157 <== osd2 10.3.14.10:6804/29550 3 ==== PGnot v
1 ==== 1548+0+0 (1870374711 0 0) 0x18888c0
2010-11-18 15:05:21.439128 7fe83b066710 osd1 11 send_incremental_map 10 -> 11 to osd2 10.3.14.10:6804/29550
2010-11-18 15:05:21.439170 7fe83b066710 -- 10.3.14.10:6802/29157 --> osd2 10.3.14.10:6804/29550 -- osd_map(11,
11) v1 -- ?+0 0x13fc400
2010-11-18 15:05:21.439205 7fe83b066710 -- 10.3.14.10:6802/29157 --> osd2 10.3.14.10:6804/29550 -- PGnot v1 --
 ?+0 0x17a6380
```

(where that PGq is the query in question). but osd2 doesn't get all that (look at <== lines)

```
2010-11-18 15:05:01.931076 7fd48dc51710 -- 10.3.14.10:6804/29550 --> osd1 10.3.14.10:6802/29157 -- PGq v1 -- ?
+0 0x2d75a80
2010-11-18 15:05:01.931549 7fd48dc51710 -- 10.3.14.10:6804/29550 <== osd1 10.3.14.10:6802/29157 1 ==== PGq v1
==== 304+0+0 (1391314930 0 0) 0x2a12000
2010-11-18 15:05:01.934263 7fd48dc51710 -- 10.3.14.10:6804/29550 --> osd1 10.3.14.10:6802/29157 -- PGnot v1 --
 ?+0 0x2dafe00
2010-11-18 15:05:01.943923 7fd48dc51710 -- 10.3.14.10:6804/29550 <== osd1 10.3.14.10:6802/29157 2 ==== PGnot v
1 ==== 6168+0+0 (2334785073 0 0) 0x2dafe00
2010-11-18 15:05:13.191211 7fd48dc51710 -- 10.3.14.10:6804/29550 mark_down 10.3.14.10:6802/29157 -- 0x2a11780
2010-11-18 15:05:16.052676 7fd48dc51710 -- 10.3.14.10:6804/29550 --> osd1 10.3.14.10:6802/29157 -- PGnot v1 --
 ?+0 0x2e871c0
2010-11-18 15:05:16.052891 7fd48dc51710 -- 10.3.14.10:6804/29550 --> osd1 10.3.14.10:6802/29157 -- PGq v1 -- ?
+0 0x2a121c0
2010-11-18 15:05:19.726086 7fd48dc51710 -- 10.3.14.10:6804/29550 --> osd1 10.3.14.10:6802/29157 -- PGnot v1 --
 ?+0 0x2e99a80
2010-11-18 15:05:24.101482 7fd48dc51710 -- 10.3.14.10:6804/29550 <== osd1 10.3.14.10:6802/29157 1 ==== PGnot v
1 ==== 11404+0+0 (1765133000 0 0) 0x2a121c0
2010-11-18 15:05:24.111139 7fd48dc51710 -- 10.3.14.10:6804/29550 <== osd1 10.3.14.10:6802/29157 2 ==== osd_map
(11,11) v1 ==== 148+0+0 (40936685 0 0) 0x29fe600
2010-11-18 15:05:24.111377 7fd48dc51710 -- 10.3.14.10:6804/29550 <== osd1 10.3.14.10:6802/29157 3 ==== PGnot v
1 ==== 3088+0+0 (1681852847 0 0) 0x2e99a80
```

The PGq in question and a few other messages are lost, because of that mark_down. Whoops. The osd does that when a peer goes down (it closes out the connection).. in this case, osd1 went down in epoch 8. But in this case osd1 has already moved on to epoch 10 and sent new messages, but osd2 is behind and is losing those as a result.

The interaction is tricky. It's not immediately obvious what the osd should be doing here, need to think about it some more.

#5 - 11/18/2010 09:50 PM - Colin McCabe

- File notes.txt added

Ah. Looks like you got it figured out.
I wasn't aware of what mark_down did.

Just in case anyone finds it useful, here are some notes I made about the situation.

#6 - 11/21/2010 03:50 PM - Sage Weil

- Subject changed from OSD peering doesn't complete to osd: rebind cluster_messenger when wrongly marked down

- Priority changed from Normal to High

I think the cleanest solution here is to re-bind the cluster_messenger to a new port when we are marked down and go back up again. Because the intra-cluster msgr is stateful, I don't think there is an easy way around this. Probably we should *always* instantiate a separate cluster msgr just to handle this case, so that we don't have to tear down our monclient session too.

#7 - 11/22/2010 09:52 AM - Sage Weil

- Status changed from New to Resolved

53d0650a42cbfd2f02db2c708a570b6d9e116bb4

Files

test_log.txt	21.8 KB	11/18/2010	Colin McCabe
pg_dump.txt	10.1 KB	11/18/2010	Colin McCabe
osd_dump.txt	1.83 KB	11/18/2010	Colin McCabe
notes.txt	20.4 KB	11/18/2010	Colin McCabe