Ceph - Bug #7915

./include/interval_set.h: 385: FAILED assert(_size >= 0)

03/31/2014 09:06 AM - Sage Weil

Status:	Duplicate	**% Done:**	0%
Priority:	Urgent	**Spent time:**	0.00 hour
Assignee:			
Category:	OSD		
Target version:			
Source:	Q/A	**Affected Versions:**	
Tags:		**ceph-qa-suite:**	
Backport:		**Pull request ID:**	
Regression:	No	**Crash signature (v1):**	
Severity:	3 - minor	**Crash signature (v2):**	
Reviewed:			

Description

ubuntu@teuthology:/a/teuthology-2014-03-30_02:30:11-rados-master-testing-basic-plana/154635

```
     0> 2014-03-30 12:24:35.293099 7f9901ee2700 -1 ./include/interval_set.h: In function 'void interval_set<T>::erase(T, T) [with T = snapid_t]' thread 7f9901ee2700 time 2014-03-30 12:24:35.271653
./include/interval_set.h: 385: FAILED assert(_size >= 0)

 ceph version 0.78-459-ge672c52 (e672c52b4f8b945a516f2eec006e33665a08f045)
 1: (interval_set<snapid_t>::subtract(interval_set<snapid_t> const&)+0x1f3) [0x795053]
 2: (PG::activate(ObjectStore::Transaction&, unsigned int, std::list<Context*, std::allocator<Context*> >&, std::map<int, std::map<spg_t, pg_query_t, std::less<spg_t>, std::allocator<std::pair<spg_t const, pg_query_t> > >, std::less<int>, std::allocator<std::pair<int const, std::map<spg_t, pg_query_t, std::less<spg_t>, std::allocator<std::pair<spg_t const, pg_query_t> > > > > >&, std::map<int, std::vector<std::pair<pg_notify_t, std::map<unsigned int, pg_interval_t, std::less<unsigned int>, std::allocator<std::pair<unsigned int const, pg_interval_t> > > >, std::allocator<std::pair<pg_notify_t, std::map<unsigned int, pg_interval_t, std::less<unsigned int>, std::allocator<std::pair<unsigned int const, pg_interval_t> > > > > >, std::less<int>, std::allocator<std::pair<int const, std::vector<std::pair<pg_notify_t, std::map<unsigned int, pg_interval_t, std::less<unsigned int> >, std::allocator<std::pair<unsigned int const, pg_interval_t> > > >, std::allocator<std::pair<pg_notify_t, std::map<unsigned int, pg_interval_t, std::less<unsigned int>, std::allocator<std::pair<unsigned int const, pg_interval_t> > > > > > > > > >*, PG::RecoveryCtx*)+0x2e2a) [0x76ce6a]
 3: (PG::RecoveryState::Active::Active(boost::statechart::state<PG::RecoveryState::Active, PG::RecoveryState::Primary, PG::RecoveryState::Activating, (boost::statechart::history_mode)0>::my_context)+0x350) [0x76d9b0]
 4: (boost::statechart::state<PG::RecoveryState::Active, PG::RecoveryState::Primary, PG::RecoveryState::Activating, (boost::statechart::history_mode)0>::shallow_construct(boost::intrusive_ptr<PG::RecoveryState::Primary> const&, boost::statechart::state_machine<PG::RecoveryState::RecoveryMachine, PG::RecoveryState::Initial, std::allocator<void>, boost::statechart::null_exception_translator>&)+0x5c) [0x7a37fc]
 5: (boost::statechart::detail::safe_reaction_result boost::statechart::simple_state<PG::RecoveryState::Peering, PG::RecoveryState::Primary, PG::RecoveryState::GetInfo, (boost::statechart::history_mode)0>::transit_impl<PG::RecoveryState::Active, PG::RecoveryState::RecoveryMachine, boost::statechart::detail::no_transition_function>(boost::statechart::detail::no_transition_function const&)+0x94) [0x7a45c4]
 6: (boost::statechart::simple_state<PG::RecoveryState::Peering, PG::RecoveryState::Primary, PG::RecoveryState::GetInfo, (boost::statechart::history_mode)0>::react_impl(boost::statechart::event_base const&, void const*)+0x192) [0x7a4862]
 7: (boost::statechart::simple_state<PG::RecoveryState::WaitUpThru, PG::RecoveryState::Peering, boost::mpl::list<mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na, mpl_::na>, (boost::statechart::history_mode)0>::react_impl(boost::statechart::event_base const&, void const*)+0xcb) [0x79f0ab]
 8: (boost::statechart::state_machine<PG::RecoveryState::RecoveryMachine, PG::RecoveryState::Initi
```

```
al, std::allocator<void>, boost::statechart::null_exception_translator>::process_queued_events()+0
xfb) [0x7855ab]
 9: (boost::statechart::state_machine<PG::RecoveryState::RecoveryMachine, PG::RecoveryState::Initi
al, std::allocator<void>, boost::statechart::null_exception_translator>::process_event(boost::stat
echart::event_base const&)+0x1e) [0x78572e]
 10: (PG::handle_activate_map(PG::RecoveryCtx*)+0x103) [0x73dfb3]
 11: (OSD::advance_pg(unsigned int, PG*, ThreadPool::TPHandle&, PG::RecoveryCtx*, std::set<boost::
intrusive_ptr<PG>, std::less<boost::intrusive_ptr<PG> >, std::allocator<boost::intrusive_ptr<PG> >
 >*)+0x61c) [0x65645c]
 12: (OSD::process_peering_events(std::list<PG*, std::allocator<PG*> > const&, ThreadPool::TPHandl
e&)+0x235) [0x656845]
 13: (OSD::PeeringWQ::_process(std::list<PG*, std::allocator<PG*> > const&, ThreadPool::TPHandle&)
+0x12) [0x6a4ed2]
 14: (ThreadPool::worker(ThreadPool::WorkThread*)+0x4e6) [0xa53476]
 15: (ThreadPool::WorkThread::entry()+0x10) [0xa55280]
 16: (()+0x7e9a) [0x7f9916f89e9a]
 17: (clone()+0x6d) [0x7f991554a3fd]
 NOTE: a copy of the executable, or `objdump -rdS <executable>` is needed to interpret this.
```

Related issues:

Related to Ceph - Bug #11493: mon: adding exsting pool as tier with --force-n...	**Resolved**	**04/28/2015**

Associated revisions

Revision 6bf46e23 - 04/02/2014 11:03 PM - Sage Weil

OSDMap: bump snap_epoch when adding a tier

When we make an existing pool a tier, we start copying the snap metadata
from the base tier. That includes removed_snaps. In order for the OSD
to recognize that this value is changing for the first time, we need to
set snap_epoch, or else the OSD doesn't update it's in-memory PGPool
with removed snaps and we eventually hit an assertion failure because
PGPool::cached_remove_snaps is incorrect (e.g., empty).

Fix this by bumping snap_epoch when we add the new tier.

Fixes: #7915
Signed-off-by: Sage Weil <sage@inktank.com>

History

#1 - 03/31/2014 09:07 AM - Sage Weil

ubuntu@teuthology:/a/teuthology-2014-03-30_02:30:11-rados-master-testing-basic-plana/154509

#2 - 03/31/2014 09:10 AM - Sage Weil

ubuntu@teuthology:/a/teuthology-2014-03-30_02:30:11-rados-master-testing-basic-plana/154663

#3 - 03/31/2014 02:03 PM - Sage Weil

Looking at the first failure:

- PGPool::cached_removed_snaps is empty
- info.purged_snaps = 1~1

hence the failure. I can't figure out why the cached_removed_snaps would be empty, though.

#4 - 04/02/2014 11:55 AM - Sage Weil

- Status changed from New to Need More Info

we've added more debug to master/firefly

#5 - 04/02/2014 04:04 PM - Sage Weil

- Status changed from Need More Info to Fix Under Review

#6 - 04/02/2014 04:13 PM - Samuel Just

- Status changed from Fix Under Review to Resolved

#7 - 03/10/2015 08:40 AM - Yann Dupont

Hi, Seems I just hitted this bug, with 0.93 :/

Scenario :
Test cluster, ceph 0.93, witch Erasure coding (6+2) Pool and cache pool. 12 OSD. 2 or 3 TB per OSD.
Low Initial number of PG on Erasure Code pool. Quite populated Pool, with a high variance on data placement : 45% full to 92% full.

Swiched from Straw to Straw2 with no much luck (in term of data placement)

Decided to raise number of PG for EC pool. from 64 (don't remember , maybe lower) to 256. This operation gave 70% of data misplaced (which is normal).

This morning only 8 out of 12 OSD were still alive.

I can't restart those OSD - Not sure it's really #7915, but symptoms are quite similar. Attached is a log with debug on.

#8 - 03/10/2015 08:42 AM - Yann Dupont

- File ceph-osd.11.log added

#9 - 03/10/2015 08:57 AM - Loïc Dachary

- Status changed from Resolved to In Progress

```
lpr=9054 pi=9010-9049/3 crt=0'0 lcod 0'0 inactive] enter Started/ReplicaActive/RepNotRecovering
2015-03-10 09:11:18.065815 7f7957de9700 10 osd.11 pg_epoch: 9054 pg[22.11( v 8641'424393 (8385'421393,8641'424
393] lb 0//0//-1 local-les=9054 n=0 ec=5576 les/c 9038/9019 9050/9050/8874) [11,0]/[2] r=-1 lpr=9054 pi=9010-9
049/3 crt=0'0 lcod 0'0 inactive] state<Started/ReplicaActive>: In ReplicaActive, about to call activate
2015-03-10 09:11:18.065834 7f7957de9700 10 osd.11 pg_epoch: 9054 pg[22.11( v 8641'424393 (8385'421393,8641'424
393] lb 0//0//-1 local-les=9054 n=0 ec=5576 les/c 9038/9019 9050/9050/8874) [11,0]/[2] r=-1 lpr=9054 pi=9010-9
049/3 crt=0'0 lcod 0'0 inactive] activate - no missing, moving last_complete 8641'424393 -> 8641'424393
2015-03-10 09:11:18.065932 7f7957de9700 10 osd.11 pg_epoch: 9054 pg[22.11( v 8641'424393 (8385'421393,8641'424
393] lb 0//0//-1 local-les=9054 n=0 ec=5576 les/c 9038/9019 9050/9050/8874) [11,0]/[2] r=-1 lpr=9054 pi=9010-9
049/3 crt=0'0 lcod 0'0 inactive] state<Started/ReplicaActive>: Activate Finished
2015-03-10 09:11:18.067071 7f7957de9700 10 write_log with: dirty_to: 4294967295'18446744073709551615, dirty_fr
om: 4294967295'18446744073709551615, dirty_divergent_priors: 1, writeout_from: 4294967295'18446744073709551615
, trimmed:
2015-03-10 09:11:18.077513 7f79565e6700 -1 ./include/interval_set.h: In function 'void interval_set<T>::erase(
T, T) [with T = snapid_t]' thread 7f79565e6700 time 2015-03-10 09:11:18.065498
./include/interval_set.h: 385: FAILED assert(_size >= 0)
```

```
 ceph version 0.93 (bebf8e9a830d998eeaab55f86bb256d4360dd3c4)
 1: (ceph::__ceph_assert_fail(char const*, char const*, int, char const*)+0x72) [0xcd7342]
 2: (interval_set<snapid_t>::subtract(interval_set<snapid_t> const&)+0xa0) [0x93fba0]
 3: (PG::activate(ObjectStore::Transaction&, unsigned int, std::list<Context*, std::allocator<Context*> >&, st
d::map<int, std::map<spg_t, pg_query_t, std::less<spg_t>, std::allocator<std::pair<spg_t const, pg_query_t> >
>, std::less<int>, std::allocator<std::pair<int const, std::map<spg_t, pg_query_t, std::less<spg_t>, std::allo
cator<std::pair<spg_t const, pg_query_t> > > > > >&, std::map<int, std::vector<std::pair<pg_notify_t, std::map
<unsigned int, pg_interval_t, std::less<unsigned int>, std::allocator<std::pair<unsigned int const, pg_interva
l_t> > > >, std::allocator<std::pair<pg_notify_t, std::map<unsigned int, pg_interval_t, std::less<unsigned int
>, std::allocator<std::pair<unsigned int const, pg_interval_t> > > > > >, std::less<int>, std::allocator<std::
pair<int const, std::vector<std::pair<pg_notify_t, std::map<unsigned int, pg_interval_t, std::less<unsigned in
t>, std::allocator<std::pair<unsigned int const, pg_interval_t> > > >, std::allocator<std::pair<pg_notify_t, s
td::map<unsigned int, pg_interval_t, std::less<unsigned int>, std::allocator<std::pair<unsigned int const, pg_
interval_t> > > > > > > > >*, PG::RecoveryCtx*)+0xd63) [0x911543]
 4: (PG::RecoveryState::Active::Active(boost::statechart::state<PG::RecoveryState::Active, PG::RecoveryState::
Primary, PG::RecoveryState::Activating, (boost::statechart::history_mode)0>::my_context)+0x53a) [0x91415a]
```

#10 - 04/08/2015 12:38 PM - Yann Dupont

Quick follow-up. AFAICS, I still have the issue with hammer.

on my test cluster, on a total of 12 OSD, 9 are starting ok, 3 exhibits the same crash at start.

Will try to dig this a little when time permit. Not urgent, I don't have valuable data here, but I'd like to understand what's going on.

#11 - 05/07/2015 04:44 PM - Florent MONTHEL

Hi,

I'm experiencing this bug I think on 0.80.4. Do you know resolve situation and start OSD ?

```
 -6> 2015-05-07 18:40:28.687257 7fa6095ed700  5 osd.26 pg_epoch: 1200 pg[25.d50( empty local-les=1090 n=0 ec=9
16 les/c 1090/1090 1089/1089/1070) [15,26,24] r=1 lpr=1090 pi=916-1088/4 crt=0'0 inactive NOTIFY] enter Starte
d/Stray
    -5> 2015-05-07 18:40:28.687288 7fa6095ed700  5 filestore(/var/lib/ceph/osd/ceph-26) queue_transactions new
 osr(25.d50 0x33b01b0)/0x33b01b0
    -4> 2015-05-07 18:40:28.687296 7fa6095ed700  5 filestore(/var/lib/ceph/osd/ceph-26) queue_transactions (wr
iteahead) 79326 0x3e75a00
    -3> 2015-05-07 18:40:28.687462 7fa617330700  5 filestore(/var/lib/ceph/osd/ceph-26) _journaled_ahead 0x367
ef90 seq 79326 osr(25.d50 0x33b01b0) 0x3e75a00
    -2> 2015-05-07 18:40:28.687483 7fa617330700  5 filestore(/var/lib/ceph/osd/ceph-26) queue_op 0x367ef90 seq
 79326 osr(25.d50 0x33b01b0) 299 bytes   (queue has 1 ops and 299 bytes)
    -1> 2015-05-07 18:40:28.687527 7fa615f2e700  5 filestore(/var/lib/ceph/osd/ceph-26) _do_op 0x367ef90 seq 7
9326 osr(25.d50 0x33b01b0)/0x33b01b0 start
     0> 2015-05-07 18:40:28.688171 7fa608bec700 -1 ./include/interval_set.h: In function 'void interval_set<T>
::erase(T, T) [with T = snapid_t]' thread 7fa608bec700 time 2015-05-07 18:40:28.684824
./include/interval_set.h: 385: FAILED assert(_size >= 0)
```

```
 ceph version 0.80.4 (7c241cfaa6c8c068bc9da8578ca00b9f4fc7567f)
 1: (interval_set<snapid_t>::subtract(interval_set<snapid_t> const&)+0x21a) [0x7dc64a]
 2: (PGPool::update(std::tr1::shared_ptr<OSDMap const>)+0x3f6) [0x7b77d6]
 3: (PG::handle_advance_map(std::tr1::shared_ptr<OSDMap const>, std::tr1::shared_ptr<OSDMap const>, std::vecto
r<int, std::allocator<int> >&, int, std::vector<int, std::allocator<int> >&, int, PG::RecoveryCtx*)+0x289) [0x
```

```
7b7b39]
 4: (OSD::advance_pg(unsigned int, PG*, ThreadPool::TPHandle&, PG::RecoveryCtx*, std::set<boost::intrusive_ptr
<PG>, std::less<boost::intrusive_ptr<PG> >, std::allocator<boost::intrusive_ptr<PG> > >*)+0x324) [0x63fa14]
 5: (OSD::process_peering_events(std::list<PG*, std::allocator<PG*> > const&, ThreadPool::TPHandle&)+0x241) [0
x647461]
 6: (OSD::PeeringWQ::_process(std::list<PG*, std::allocator<PG*> > const&, ThreadPool::TPHandle&)+0x16) [0x69e
5d6]
 7: (ThreadPool::worker(ThreadPool::WorkThread*)+0x551) [0x9dc6c1]
 8: (ThreadPool::WorkThread::entry()+0x10) [0x9df700]
 9: /lib64/libpthread.so.0() [0x31252079d1]
 10: (clone()+0x6d) [0x3124ee88fd]
 NOTE: a copy of the executable, or `objdump -rdS <executable>` is needed to interpret this.
```

#12 - 05/12/2015 09:46 AM - Achim Ledermüller

Ceph version 0.94.1 (e4bfad3a3c51054df7e537a724c8d0bf9be972ff)
Ubuntu 14.04 Trusty with 3.13.0 Kernel

Hi,

it seems that we hit the same issue. For testing purpose we added a cache pool (cache_test) to an existing pool (test) and then we created a snapshot (named test2). Almost immediately all OSDs in the cache pool dropped out. Actually there was no existing snapshot on this rbd, but in the past we already created and removed a snapshot with the name (test2). We were not able to start he OSDs again, even worse, we had to zap the affected osd.

```
# rados df
pool name                 KB      objects       clones     degraded      unfound           rd        rd KB
       wr        wr KB
cache_test          10385729         3061            0            0           0     10117093     39438844
 7105087     13963985
test                10398765         2561            0            0           0       440607      1343283
  210835    232040725

# rbd snap ls fio_test_rep -p test
# rbd snap create test/fio_test_rep@test2
2015-05-12 10:03:11.449464 7f046c160700  0 -- 10.10.0.146:0/1022176 >> 10.10.0.134:6800/14663 pipe(0x7f0464082
b90 sd=7 :0 s=1 pgs=0 cs=0 l=1 c=0x7f0464086e30).fault
2015-05-12 10:03:11.461046 7f046c665700  0 -- 10.10.0.146:0/1022176 >> 10.10.0.134:6804/14678 pipe(0x7f0464088
570 sd=7 :0 s=1 pgs=0 cs=0 l=1 c=0x7f046408c810).fault
2015-05-12 10:03:11.469816 7f046c463700  0 -- 10.10.0.146:0/1022176 >> 10.10.0.134:6808/14771 pipe(0x7f046408e
c80 sd=7 :0 s=1 pgs=0 cs=0 l=1 c=0x7f0464092f20).fault
^C

# rbd snap ls fio_test_rep -p test
2015-05-12 10:03:20.358823 7f23cc777700  0 -- 10.10.0.146:0/1024685 >> 10.10.0.134:6800/14663 pipe(0x421c680 s
d=4 :0 s=1 pgs=0 cs=0 l=1 c=0x4220920).fault

# ceph status
    cluster 4fc7754a-5ca0-491f-a5a7-6230d12ca8c6
     osdmap e173920: 105 osds: 105 up, 105 in

# ceph status
    cluster 4fc7754a-5ca0-491f-a5a7-6230d12ca8c6
     health HEALTH_WARN
            1165 pgs stale
            11/105 in osds are down
     osdmap e173925: 105 osds: 94 up, 105 in
                1164 stale+active+clean
  client io 7338 kB/s rd, 3971 kB/s wr, 1134 op/s
```

Log from a dropped out OSD:

```
2015-05-12 10:03:10.070016 7f96b42b1700  0 log_channel(cluster) log [INF] : 27.31b scrub ok
2015-05-12 10:03:11.328695 7f9699f33700  0 -- 10.10.9.102:6809/14763 >> 10.10.9.103:6811/1227 pipe(0x24156000
sd=506 :37318 s=2 pgs=4051 cs=497 l=0 c=0x1729da20).fault with nothing to send,
going to standby
2015-05-12 10:03:11.332428 7f967dab7700  0 -- 10.10.9.102:0/14763 >> 10.10.9.103:6812/1227 pipe(0xe74d000 sd=3
78 :0 s=1 pgs=0 cs=0 l=1 c=0x95d52e0).fault
2015-05-12 10:03:11.332512 7f967efcc700  0 -- 10.10.9.102:0/14763 >> 10.10.0.140:6810/1227 pipe(0x29860000 sd=
379 :0 s=1 pgs=0 cs=0 l=1 c=0x95d5860).fault
2015-05-12 10:03:11.342014 7f96baabe700 -1 ./include/interval_set.h: In function 'void interval_set<T>::erase(
T, T) [with T = snapid_t]' thread 7f96baabe700 time 2015-05-12 10:03:11.336429
./include/interval_set.h: 385: FAILED assert(_size >= 0)

 ceph version 0.94.1 (e4bfad3a3c51054df7e537a724c8d0bf9be972ff)
 1: (ceph::__ceph_assert_fail(char const*, char const*, int, char const*)+0x8b) [0xbc271b]
 2: (interval_set<snapid_t>::subtract(interval_set<snapid_t> const&)+0xb0) [0x82cd50]
 3: (PGPool::update(std::tr1::shared_ptr<OSDMap const>)+0x52e) [0x80113e]
 4: (PG::handle_advance_map(std::tr1::shared_ptr<OSDMap const>, std::tr1::shared_ptr<OSDMap const>, std::vecto
r<int, std::allocator<int> >&, int, std::vector<int, std::allocator<int> >&, int
, PG::RecoveryCtx*)+0x282) [0x801652]
 5: (OSD::advance_pg(unsigned int, PG*, ThreadPool::TPHandle&, PG::RecoveryCtx*, std::set<boost::intrusive_ptr
<PG>, std::less<boost::intrusive_ptr<PG> >, std::allocator<boost::intrusive_ptr<
PG> > >*)+0x2c3) [0x6b0e43]
 6: (OSD::process_peering_events(std::list<PG*, std::allocator<PG*> > const&, ThreadPool::TPHandle&)+0x21c) [0
x6b191c]
 7: (OSD::PeeringWQ::_process(std::list<PG*, std::allocator<PG*> > const&, ThreadPool::TPHandle&)+0x18) [0x709
278]
 8: (ThreadPool::worker(ThreadPool::WorkThread*)+0xa5e) [0xbb38ae]
 9: (ThreadPool::WorkThread::entry()+0x10) [0xbb4950]
 10: (()+0x8182) [0x7f96d80cd182]
 11: (clone()+0x6d) [0x7f96d663847d]
```

What's going on? Is it possible to start the OSDs again?

#13 - 05/26/2015 05:21 PM - Loïc Dachary

- Status changed from In Progress to 12

- Regression set to No

#14 - 06/02/2015 08:19 PM - Sage Weil

- Status changed from 12 to Duplicate

Files

ceph-osd.11.log	31.5 KB	03/10/2015	Yann Dupont