Ceph - Bug #698

cosd memory usage with large number of pools

01/11/2011 10:49 AM - John Leach

Status:	Resolved	**% Done:**	0%
Priority:	Normal	**Spent time:**	0.00 hour
Assignee:	Greg Farnum		
Category:	OSD		
Target version:	v0.26		
Source:		**Reviewed:**	
Tags:		**Affected Versions:**	
Backport:		**ceph-qa-suite:**	
Regression:	No	**Pull request ID:**	
Severity:	3 - minor	**Crash signature:**	

Description

I reported this on the mailing list a week ago but never filed it here. Still present in 0.24.1.

I've got a 3 node test cluster (3 mons, 3 osds) with about 24,000,000 very small objects across 2400 pools (written directly with librados, this isn't a ceph filesystem).

The cosd processes have steadily grown in ram size and have finally exhausted ram and are getting killed by the oom killer (the nodes have 6gig RAM and no swap).

When I start them back up they just very quickly increase in ram size again and get killed.

I'm running ceph 0.24 (and 0.24.1) on 64bit Ubuntu Lucid servers. In case it's useful, I've just written these objects serially, no reading, no rewrites, updates or snapshots.

I haven't touched the pg_nums on this cluster that I recall (it's been up a couple of weeks but has nearly exclusively been used for writing this test data).

tcmalloc heap profiling report attached (last profile before cosd was killed).

cosd debug output from around the time of the final ram exhaustion:

```
2011-01-05 00:17:58.532524   mon e1: 3 mons at {0=10.135.211.78:6789/0,1=10.61.136.222:6789/0,2=10
.202.105.222:6789/0}
2011-01-05 00:22:53.325264   osd e10659: 3 osds: 3 up, 3 in
2011-01-05 00:22:53.383272    pg v151295: 20936 pgs: 1 creating, 2 peering, 10352 crashed+peering,
 3052 active+clean+degraded, 7053 degraded+peering, 476 crashed+degraded+peering; 24130 MB data, 2
66 GB used, 332 GB / 630 GB avail; 12489924/49420044 degraded (25.273%)
2011-01-05 00:22:53.422433   log 2011-01-05 00:22:53.325027 mon0 10.135.211.78:6789/0 4 : [INF] os
d0 10.135.211.78:6801/31836 boot
2011-01-05 00:24:47.301186    pg v151296: 20936 pgs: 1 creating, 2 peering, 10352 crashed+peering,
 3052 active+clean+degraded, 7053 degraded+peering, 476 crashed+degraded+peering; 24130 MB data, 2
66 GB used, 332 GB / 630 GB avail; 12489924/49420044 degraded (25.273%)

<cosd crashes here>

2011-01-05 00:25:52.422340   log 2011-01-05 00:25:52.189259 mon0 10.135.211.78:6789/0 5 : [INF] os
d0 10.135.211.78:6801/31836 failed (by osd2 10.61.136.222:6800/915)
2011-01-05 00:25:57.265635   log 2011-01-05 00:25:57.121870 mon0 10.135.211.78:6789/0 6 : [INF] os
d0 10.135.211.78:6801/31836 failed (by osd2 10.61.136.222:6800/915)
2011-01-05 00:26:02.341805   osd e10660: 3 osds: 2 up, 3 in
```

```
2011-01-05 00:26:02.362526   log 2011-01-05 00:26:02.127627 mon0 10.135.211.78:6789/0 7 : [INF] os
d0 10.135.211.78:6801/31836 failed (by osd2 10.61.136.222:6800/915)
2011-01-05 00:26:02.470942    pg v151297: 20936 pgs: 1 creating, 2 peering, 10352 crashed+peering,
 3052 active+clean+degraded, 7053 degraded+peering, 476 crashed+degraded+peering; 24130 MB data, 2
66 GB used, 332 GB / 630 GB avail; 12489924/49420044 degraded (25.273%)
2011-01-05 00:26:12.578266    pg v151298: 20936 pgs: 1 creating, 2 peering, 3393 crashed+peering,
3052 active+clean+degraded, 7053 degraded+peering, 7435 crashed+degraded+peering; 24130 MB data, 2
66 GB used, 332 GB / 630 GB avail; 20728862/49420044 degraded (41.944%)
```

History

#1 - 01/11/2011 10:51 AM - Sage Weil

- Category set to OSD

- Assignee set to Colin McCabe

- Target version set to v0.24.2

#2 - 01/28/2011 10:50 AM - Sage Weil

- Target version changed from v0.24.2 to v0.24.3

#3 - 01/31/2011 12:06 PM - Greg Farnum

- Target version changed from v0.24.3 to 19

#4 - 02/04/2011 02:15 PM - Sage Weil

- Target version changed from 19 to v0.26

#5 - 02/04/2011 02:15 PM - Sage Weil

- Assignee deleted (Colin McCabe)

#6 - 02/04/2011 02:16 PM - Sage Weil

- Subject changed from huge cosd memory usage with large number of objects to cosd memory usage with large number of pools

#7 - 02/09/2011 10:49 AM - Greg Farnum

- Status changed from New to In Progress

- Assignee set to Greg Farnum

Taking a look now!

#8 - 02/10/2011 08:38 AM - Greg Farnum

I ran the OSD through massif and in one of my tests, creating 2000 pools (with default 8 PGs), 1/3 of the ending memory was used up by MPGStatsAck. Best guess is that the single pg_stat_queue_lock is slowing things down too much and the messages are stacking up.

Continuing to look for other potential issues.

#9 - 02/14/2011 09:23 AM - Greg Farnum

It seems the messages stacking up are just a result of massif slowing down operations too much.

According to massif, each PG takes ~14KB of memory when empty and without any peers.
There are some hash_maps taking up a bunch of unnecessary space, though -- converted a couple to maps and removed an unused one. This should be a significant decrease but haven't tested how much.

I'm probablying going to have to proceed testing with tcmalloc heap profiling rather than massif, since the massif performance is causing problems. Hopefully I can at least run tests via massif with peering to see how that impacts PG memory use, since massif is a bit more precise at profiling exactly where the memory use is going.

#10 - 02/16/2011 09:22 PM - Greg Farnum

Hmm, I just tried to run one of my tests using 2 OSDs instead of one for the first time. It looks like it's still possible for them to get backed up on PG messages from the monitor -- I tried to send heapdump commands and it was I think minutes between when the monitor "sent" it and the OSD "received" it (according to the dout logs). Will update tomorrow, probably with a new bug.

#11 - 02/17/2011 09:24 AM - Greg Farnum

So, yeah. All the heapdump commands were sent **after** ceph -w reported the PGs were all active+clean, although I didn't check cosd CPU usage.

```
gregf@kai:~/testing/ceph/src/out$ grep heapdump mon.a | grep osd0 | less
2011-02-16 21:26:03.188484 7f6b53290710 mon.a@0(leader) e1 send_command osd0 10.0.1.205:6800/14185[heapdump]
2011-02-16 21:26:03.188499 7f6b53290710 mon.a@0(leader) e1 try_send_message mon_command(heapdump v 463) v1 to
osd0 10.0.1.205:6800/14185
2011-02-16 21:26:03.188519 7f6b53290710 -- 10.0.1.205:6789/0 --> osd0 10.0.1.205:6800/14185 -- mon_command(hea
pdump v 463) v1 -- ?+0 0x7f6b480432a0
2011-02-16 21:26:43.467015 7f6b53290710 mon.a@0(leader) e1 send_command osd0 10.0.1.205:6800/14185[heapdump]
2011-02-16 21:26:43.467036 7f6b53290710 mon.a@0(leader) e1 try_send_message mon_command(heapdump v 463) v1 to
osd0 10.0.1.205:6800/14185
2011-02-16 21:26:43.467057 7f6b53290710 -- 10.0.1.205:6789/0 --> osd0 10.0.1.205:6800/14185 -- mon_command(hea
pdump v 463) v1 -- ?+0 0x7f6b44062910
2011-02-16 21:27:17.594580 7f6b53290710 mon.a@0(leader) e1 send_command osd0 10.0.1.205:6800/14185[heapdump]
2011-02-16 21:27:17.594596 7f6b53290710 mon.a@0(leader) e1 try_send_message mon_command(heapdump v 463) v1 to
osd0 10.0.1.205:6800/14185
2011-02-16 21:27:17.594616 7f6b53290710 -- 10.0.1.205:6789/0 --> osd0 10.0.1.205:6800/14185 -- mon_command(hea
pdump v 463) v1 -- ?+0 0x7f6b44062910
2011-02-16 21:27:31.665497 7f6b53290710 mon.a@0(leader) e1 send_command osd0 10.0.1.205:6800/14185[heapdump]
2011-02-16 21:27:31.665512 7f6b53290710 mon.a@0(leader) e1 try_send_message mon_command(heapdump v 463) v1 to
osd0 10.0.1.205:6800/14185
2011-02-16 21:27:31.665533 7f6b53290710 -- 10.0.1.205:6789/0 --> osd0 10.0.1.205:6800/14185 -- mon_command(hea
pdump v 463) v1 -- ?+0 0x7f6b4c05d790
2011-02-16 21:29:31.038685 7f6b53290710 mon.a@0(leader) e1 send_command osd0 10.0.1.205:6800/14185[heapdump]
2011-02-16 21:29:31.038700 7f6b53290710 mon.a@0(leader) e1 try_send_message mon_command(heapdump v 463) v1 to
osd0 10.0.1.205:6800/14185
2011-02-16 21:29:31.038720 7f6b53290710 -- 10.0.1.205:6789/0 --> osd0 10.0.1.205:6800/14185 -- mon_command(hea
pdump v 463) v1 -- ?+0 0x7f6b48017550
```

```
gregf@kai:~/testing/ceph/src/out$ grep heapdump osd.0  | less
2011-02-16 21:37:59.379852 7f8999b57710 -- 10.0.1.205:6800/14185 <== mon0 10.0.1.205:6789/0 1163 ==== mon_comm
and(heapdump v 463) v1 ==== 50+0+0 (3078000875 0 0) 0x2c91700 con 0x19e1500
2011-02-16 21:37:59.379885 7f8999b57710 osd0 462 _dispatch 0x2c91700 mon_command(heapdump v 463) v1
2011-02-16 21:37:59.379907 7f8999b57710 osd0 462 handle_command args: [heapdump]
2011-02-16 21:37:59.478496 7f8999b57710 -- 10.0.1.205:6800/14185 <== mon0 10.0.1.205:6789/0 1171 ==== mon_comm
and(heapdump v 463) v1 ==== 50+0+0 (3078000875 0 0) 0x258f8c0 con 0x19e1500
2011-02-16 21:37:59.478528 7f8999b57710 osd0 462 _dispatch 0x258f8c0 mon_command(heapdump v 463) v1
2011-02-16 21:37:59.478551 7f8999b57710 osd0 462 handle_command args: [heapdump]
2011-02-16 21:37:59.578440 7f8999b57710 -- 10.0.1.205:6800/14185 <== mon0 10.0.1.205:6789/0 1179 ==== mon_comm
and(heapdump v 463) v1 ==== 50+0+0 (3078000875 0 0) 0x47ffa80 con 0x19e1500
2011-02-16 21:37:59.578472 7f8999b57710 osd0 462 _dispatch 0x47ffa80 mon_command(heapdump v 463) v1
2011-02-16 21:37:59.578494 7f8999b57710 osd0 462 handle_command args: [heapdump]
2011-02-16 21:37:59.611745 7f8999b57710 -- 10.0.1.205:6800/14185 <== mon0 10.0.1.205:6789/0 1182 ==== mon_comm
and(heapdump v 463) v1 ==== 50+0+0 (3078000875 0 0) 0x47ff540 con 0x19e1500
2011-02-16 21:37:59.611777 7f8999b57710 osd0 462 _dispatch 0x47ff540 mon_command(heapdump v 463) v1
2011-02-16 21:37:59.611799 7f8999b57710 osd0 462 handle_command args: [heapdump]
2011-02-16 21:38:02.505711 7f8999b57710 -- 10.0.1.205:6800/14185 <== mon0 10.0.1.205:6789/0 1219 ==== mon_comm
and(heapdump v 463) v1 ==== 50+0+0 (3078000875 0 0) 0x2cbc000 con 0x19e1500
2011-02-16 21:38:02.505751 7f8999b57710 osd0 463 _dispatch 0x2cbc000 mon_command(heapdump v 463) v1
2011-02-16 21:38:02.505774 7f8999b57710 osd0 463 handle_command args: [heapdump]
```

Presumably this is related to DongJin's issue #810, and maybe to Jim Schutt's trouble on the list with large clusters heartbeating.

Have we done something to make PGs and peering more expensive recently, or is the system just not well-tested with large numbers of PGs like this and others are seeing it because the system doesn't create intelligent default PG numbers for large clusters?

#12 - 02/18/2011 03:25 PM - Greg Farnum

I merged code which removes some hash_maps in a7929c5e265d1b6502733ee9525fd93bbcfc739e; this takes per-PG memory use from ~14KB to ~4KB in the case of empty PGs.

I will review this on an ongoing basis but for now issues with memory fragmentation and map churn are far more significant than the actual PG memory use.

#13 - 02/27/2011 07:42 PM - Greg Farnum

- Status changed from In Progress to Resolved

I'm closing this since it's become apparent that the actual memory use issues are less related to the in-memory objects and more about issues with message spam, peering, and general problems with odd memory use, all of which we're working on!

#14 - 03/13/2011 02:39 PM - Sage Weil

- translation missing: en.field_story_points set to 3

- translation missing: en.field_position set to 1

- translation missing: en.field_position changed from 1 to 543

Files

osd.0.0017.heap.pprof.txt.gz	6.35 KB	01/11/2011	John Leach