rgw - Bug #27219

lock in resharding may expires before the dynamic resharding completes

08/24/2018 01:14 PM - Jeegn Chen

Status:	Resolved	**Start date:**	08/24/2018
Priority:	Normal	**Due date:**	
Assignee:	Eric Ivancich	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:		**Spent time:**	0.00 hour
Source:		**Reviewed:**	
Tags:		**Affected Versions:**	v12.2.2
Backport:	mimic,luminous	**ceph-qa-suite:**	
Regression:	No	**Pull request ID:**	
Severity:	3 - minor		

Description

Orit Wasserman <owasserm@redhat.com> □2018□8□19□□□ □□5:38□□□

> Hi,
> On Thu, Aug 16, 2018 at 8:11 PM Yehuda Sadeh-Weinraub <yehuda@redhat.com> wrote:
>
> > Hi,
> >
> > I don't remember the exact details right now, but we might be renewing it periodically as reshard happens. Orit?
>
> Looking at the code we do not renew the lock and we should have, Jeegn please open a tracker issue for this.
> This won't cause corruption as we complete the resharding regardless of the lock and a new thread will use a different bucket instance so it won't corrupt this one. This still wastes lots of system resources for nothing. At the moment we our difualt is a one thread so it could be only a different Radosgw instance but still possible.
>
> Regards,
> Orit
>
> > On Wed, Aug 15, 2018, 6:17 AM Jeegn Chen <jeegnchen@gmail.com> wrote:
> >
> > > Hi Yehuda,
> > >
> > > I just took a look at the code related to Dynamic Resharding in
> > > Luminous. Not sure whether I'm correct but my impression is that
> > > Dynamic Resharding logic does not address buckets with large number of
> > > objects properly, especially when there are multiple RGW processes.
> > >
> > > My major concern comes from the short lock expiration. For example,
> > > the expiration time is just 60 seconds in
> > > RGWReshard::process_single_logshard(). If RGWBucketReshard::execute()
> > > following the lock acquisition takes long time to deal with large
> > > buckets, the 60-seconds will not be enough (If a bucket is large
> > > enough, hours may even not be enough). As a result, another Dynamic
> > > Resharding thread in another RGW process may grab the log shard and
> > > work on the same buckets at the same time, which potentially cause
> > > corruption.
> > >
> > > Did I misunderstand something or is my concern valid?
> > >
> > > int RGWReshard::process_single_logshard(int logshard_num) {
> > > string marker;
> > > bool truncated = true;

```
CephContext *cct = store->ctx();
int max_entries = 1000;
int max_secs = 60;<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

rados::cls::lock::Lock l(reshard_lock_name);

utime_t time(max_secs, 0);<<<<<<<<<<<<<<<<<<<<<<<<<
l.set_duration(time);<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

char cookie_buf[COOKIE_LEN + 1];
gen_rand_alphanumeric(store->ctx(), cookie_buf, sizeof(cookie_buf) - 1);
cookie_buf[COOKIE_LEN] = '\0';

l.set_cookie(cookie_buf);

string logshard_oid;
get_logshard_oid(logshard_num, &logshard_oid);

int ret = l.lock_exclusive(&store->reshard_pool_ctx, logshard_oid);
```

Thanks,
Jeegn

Related issues:

Related to rgw - Bug #24551: RGW Dynamic bucket index resharding keeps reshar...	**Resolved**	**06/18/2018**
Related to rgw - Bug #24937: [rgw] Very high cache misses with automatic buck...	**Resolved**	**07/16/2018**
Copied to rgw - Backport #36687: mimic: lock in resharding may expires before...	**Resolved**	
Copied to rgw - Backport #36688: luminous: lock in resharding may expires bef...	**Resolved**	

History

#1 - 09/12/2018 11:58 AM - Orit Wasserman

- Assignee set to Orit Wasserman

#2 - 09/12/2018 12:29 PM - Orit Wasserman

The renew of the lock logic is run in process_single_shard but a single buckert resharding can take a long time.
The renew code needs to move into do_reshard and executed before and after any long operation like bi_list and bi_put.
In case the lock has expired the resharding operation should be aborted and all objects created should be deleted.

#3 - 09/12/2018 11:18 PM - Jorge Campos

- File ceph.conf added

- File ceph-client.rgw.baydat02.log.zip added

- File ceph-client.rgw.baydat04.log.zip added

#4 - 09/13/2018 06:44 AM - Orit Wasserman

- File deleted (ceph.conf)

#5 - 09/13/2018 06:44 AM - Orit Wasserman

- File deleted (ceph-client.rgw.baydat02.log.zip)

#6 - 09/13/2018 06:44 AM - Orit Wasserman

- File deleted (ceph-client.rgw.baydat04.log.zip)

#7 - 09/13/2018 06:45 AM - Orit Wasserman

Hello,

I’m experiencing this myself at the moment. Is there a workaround? I’m running Ceph 12.2.7 on a 4-node cluster, each node with one 2.8TB SSD OSD, and 20 CPU cores (3GHz).

Scenario: I have several large buckets, the largest one with over 9 million objects. I have 2 rgw clients that seem to interfere with each other, as they alternate “failed to acquire lock on obj_delete_at_hint.{number}” log entries and appear to be processing at around the same hint entry.

I don’t think either rgw client finishes the resharding process, as they reach:
object expiration: stop
object expiration: start
at which point they loop back to object_delete_at_hint.0000000000

I’ve tried increasing rgw_reshard_hints_num_shards to 1024, hoping that the rgw clients would finish resharding before reaching the rgw_reshard_hints_num_shards limit, with no success. The resharding process ran all night, over and over, without resharding any bucket (there's 6 buckets that need resharding).

Is there any other workaround I can try to let resharding finish? Also, any advice on how I can speed up the resharding process? I don't mind using lots more CPU for this to minimize the amount of time that my large buckets are locked (which prevents my app from writing to it).

I'm new to ceph, but have been devouring online documentation and bug reports to learn the ceph internals and search for a workaround, but haven't found a solution to this yet.

Best regards,
Jorge

#8 - 09/13/2018 06:48 AM - Orit Wasserman

Hi Joreg,
You can try disabling dynamic resharding in the ceph conf file as a temporary workaround.
You can use "reshard cancel command" to cancel ongoing resharding

#9 - 09/16/2018 06:59 PM - Jorge Campos

Thanks Orit.
I disabled resharding and have no problems writing to my buckets. I’ll await for the next major development regarding rgw resharding before re-enabling it.

Also, I upgraded to Mimic, and it’s working great so far! My deep scrubs were taking too long due to stupidalloc dumps (I store over 30M objects), but the new bitmap allocator seems to have solved those issues.

Cheers,
Jorge

#10 - 09/16/2018 07:22 PM - Jorge Campos

Update: I just noticed Mimic still uses stupidalloc by default. Either way, I'm not experiencing the same slowdown and stupdalloc dumps during scrubs.

I'm tempted to try bitmap allocator given that my cluster stores over 30M small objects. Do you recommend I make the switch at this point?

Cheers,
Jorge

#11 - 10/03/2018 08:59 PM - Eric Ivancich

- Assignee changed from Orit Wasserman to Eric Ivancich

PR: https://github.com/ceph/ceph/pull/24406

#12 - 11/03/2018 03:17 AM - Nathan Cutler

- Status changed from New to Pending Backport

- Backport set to mimic,luminous

#13 - 11/03/2018 03:17 AM - Nathan Cutler

- Copied to Backport #36687: mimic: lock in resharding may expires before the dynamic resharding completes added

#14 - 11/03/2018 03:17 AM - Nathan Cutler

- Copied to Backport #36688: luminous: lock in resharding may expires before the dynamic resharding completes added

#15 - 11/03/2018 03:25 AM - Nathan Cutler

- Related to Bug #24551: RGW Dynamic bucket index resharding keeps resharding all buckets added

#16 - 11/03/2018 03:27 AM - Nathan Cutler

- Related to Bug #24937: [rgw] Very high cache misses with automatic bucket resharding added

#17 - 12/12/2018 07:42 PM - Nathan Cutler

- Status changed from Pending Backport to Resolved