rgw - Bug #20663

Segmentation fault when exporting rgw bucket in nfs-ganesha

07/18/2017 11:42 AM - supriti singh

Status:	Resolved	**% Done:**	0%
Priority:	Normal	**Spent time:**	0.00 hour
Assignee:	Matt Benjamin		
Category:			
Target version:	v12.1.0		
Source:	Q/A	**Affected Versions:**	v12.1.0
Tags:		**ceph-qa-suite:**	
Backport:	kraken jewel	**Pull request ID:**	
Regression:	No	**Crash signature (v1):**	
Severity:	3 - minor	**Crash signature (v2):**	
Reviewed:			

Description

We see a segmentation fault when trying to export rgw bucket using nfs-ganesha (with latest master and ceph v 12.1.0) It happens only on Intel arch.
Looking at the core dump, the segmentation fault occurs at __lll_unlock_elision (). It occurs because unlock is called on object on which no lock was taken. From man page of pthread_mutex_unlock: "f a thread attempts to unlock a mutex that it has not locked or a mutex which is unlocked, undefined behavior results." With lock elision unlocking a free lock is not tolerated anymore. Hence it occurs only on intel arch.

This seems like the culprit code: https://github.com/ceph/ceph/blob/master/src/rgw/rgw_file.h#L1032-L1037

The lock on "fh->mtx.lock" is taken and in insert_latched(), "lat.lock->unlock();" is called. Lock (lat.lock->lock()) should be taken on latch before calling unlock.

Related issues:

Copied to rgw - Backport #20711: kraken: Segmentation fault when exporting rg...	**Rejected**
Copied to rgw - Backport #20820: jewel: Segmentation fault when exporting rgw...	**Resolved**

History

#1 - 07/18/2017 12:38 PM - Matt Benjamin

- Status changed from New to In Progress

- Assignee set to Matt Benjamin

hi supriti,

Releasing lat.lock having taken fh->mtx is not per-se a smoking gun, as we have taken lat.lock at find_latch unconditionally wrt the value of fh, at l. 997. Whether we take (and maybe hold) fh->mtx deals with serialization on the new handle that callers may require. By the time we return from this routine, latch will be unlocked, but fh->mtx may not be. fh_cache.insert_latched should be releasing the lock taken at 997.

This looks like it could be a bug in using Mark's export by bucket syntax? Could you share more of your config (e.g., ganesha.conf, indicate whether demo-demo exists and should be visible to the export credential)--feel free to share by email.

regards,

Matt

#2 - 07/18/2017 06:24 PM - Matt Benjamin

@supriti, so far I'm not hitting this assert, mounting with pseudo /demo, at demo-demo alongside unmounted demo1-demo.

One thing that would cause this is simple corruption of the filehandle table, which I guess would have to have happened almost immediately. Just to rule it out, could you maybe run with valgrind (default engine)?

Matt

#3 - 07/19/2017 02:01 PM - supriti singh

Matt Benjamin wrote:

> hi supriti,
>
> Releasing lat.lock having taken fh->mtx is not per-se a smoking gun, as we have taken lat.lock at find_latch unconditionally wrt the value of fh, at l. 997. Whether we take (and maybe hold) fh->mtx deals with serialization on the new handle that callers may require. By the time we return from this routine, latch will be unlocked, but fh->mtx may not be. fh_cache.insert_latched should be releasing the lock taken at 997.

As you said we are unlocking the lock on "lat" unconditionally in function find_latch (as we pass flag FLAG_LOCK when calling fh_cache.find_latch()). Then why do we need to again call unlock on latch in fh_cache.insert_latch?

> This looks like it could be a bug in using Mark's export by bucket syntax? Could you share more of your config (e.g., ganesha.conf, indicate whether demo-demo exists and should be visible to the export credential)--feel free to share by email.
>
> regards,
>
> Matt

#4 - 07/19/2017 02:05 PM - Matt Benjamin

Hi Supriti,

I think we ~~lock~~ lat.lock unconditionally in find_latch, but should NOT be unlocking it. we ~~would~~ do the unlock if the FLAG_UNLOCK is given, but the intent is rather to hold the partition lock until the caller either fails or has inserted positionally into the already-locked partition. then we want to unlock lat.lock.

Matt

#5 - 07/19/2017 03:10 PM - supriti singh

Matt Benjamin wrote:

> Hi Supriti,
>
> I think we ~~lock~~ lat.lock unconditionally in find_latch, but should NOT be unlocking it. we ~~would~~ do the unlock if the FLAG_UNLOCK is given, but the intent is rather to hold the partition lock until the caller either fails or has inserted positionally into the already-locked partition. then we want to unlock lat.lock.
>
> Matt

May be I am missing something, but in function find_latch() (https://github.com/ceph/ceph/blob/master/src/common/cohort_lru.h#L420) we release the lock as flag=FLAG_LOCK. If the intention is to hold the lock for the whole period, may be we should call unlock only in insert_latch().

#6 - 07/19/2017 03:32 PM - supriti singh

Somehow my original reply in comment#6 was deleted, restoring:
"""
Hi Supriti,

The condition is "if (flags & (FLAG_LOCK|FLAG_UNLOCK))" which means to
say, unlock it iff:
1. we have the flag which requests ~~us~~ to take the lock, and
2. we have the flag requesting us to unlock on exit

I hope my C stretches to getting this expression correct, but if it's
a howler, will be delighted to fix.
"""

Matt Benjamin wrote:

> @supriti, so far I'm not hitting this assert, mounting with pseudo /demo, at demo-demo alongside unmounted demo1-demo.
>
> One thing that would cause this is simple corruption of the filehandle table, which I guess would have to have happened almost immediately. Just to rule it out, could you maybe run with valgrind (default engine)?
>
> Matt

I will try to run again on hardware where we see the issue and report back.

#7 - 07/19/2017 09:21 PM - supriti singh

supriti singh wrote:

> Somehow my original reply in comment#6 was deleted, restoring:
> """
> Hi Supriti,
>
> The condition is "if (flags & (FLAG_LOCK|FLAG_UNLOCK))" which means to
> say, unlock it iff:
> 1. we have the flag which requests ~~us~~ to take the lock, and
> 2. we have the flag requesting us to unlock on exit
>
> I hope my C stretches to getting this expression correct, but if it's
> a howler, will be delighted to fix.
> """

This condition will be expanded as (flags & FLAG_LOCK) | (flags & FLAG_UNLOCK). Hence, it will be true even if only one of the flag is set.

If we want to check if both are set, then it should be: if (flags & FLAG_LOCK) && (flags & FLAG_UNLOCK))

> Matt Benjamin wrote:
>
> > @supriti, so far I'm not hitting this assert, mounting with pseudo /demo, at demo-demo alongside unmounted demo1-demo.
> >
> > One thing that would cause this is simple corruption of the filehandle table, which I guess would have to have happened almost immediately. Just to rule it out, could you maybe run with valgrind (default engine)?
> >
> > Matt
>
> I will try to run again on hardware where we see the issue and report back.

#8 - 07/20/2017 12:25 PM - Matt Benjamin

- Status changed from In Progress to Pending Backport

- Backport set to luminous kraken jewel

arg, it's...bitwise, sorry, will update

I'm surprised this isn't universally hit.

https://github.com/ceph/ceph/pull/16448

#9 - 07/20/2017 08:27 PM - Nathan Cutler

- Copied to Backport #20711: kraken: Segmentation fault when exporting rgw bucket in nfs-ganesha added

#10 - 07/20/2017 08:28 PM - Nathan Cutler

- Backport changed from luminous kraken jewel to kraken jewel

#11 - 07/20/2017 08:28 PM - Nathan Cutler

luminous backports haven't started yet

#12 - 07/24/2017 01:10 PM - supriti singh

Matt Benjamin wrote:

> arg, it's...bitwise, sorry, will update
>
> I'm surprised this isn't universally hit.
>
> https://github.com/ceph/ceph/pull/16448

Tested with this patch. Issue seemed to be resolved.

#13 - 07/28/2017 06:16 AM - Nathan Cutler

- Copied to Backport #20820: jewel: Segmentation fault when exporting rgw bucket in nfs-ganesha added

#14 - 09/12/2017 08:11 PM - Nathan Cutler

- Status changed from Pending Backport to Resolved

Files

backtrace.txt	3.48 KB	07/18/2017	supriti singh