CephFS - Bug #37378

truncate_seq ordering issues with object creation

11/23/2018 10:17 AM - Luis Henriques

Status:	Resolved	**% Done:**	0%
Priority:	High		
Assignee:			
Category:			
Target version:			
Source:	Development	**ceph-qa-suite:**	
Tags:		**Component(FS):**	
Backport:		**Labels (FS):**	
Regression:	No	**Pull request ID:**	31728
Severity:	3 - minor	**Crash signature (v1):**	
Reviewed:		**Crash signature (v2):**	
Affected Versions:			

Description

I'm seeing a bug with copy_file_range in recent clients. Here's a simple way to reproduce it:

```
# set files layouts
touch a b
setfattr -n ceph.file.layout -v "stripe_unit=65536 stripe_count=1 object_size=65536" a
setfattr -n ceph.file.layout -v "stripe_unit=65536 stripe_count=1 object_size=65536" b

# create 'a' and 'b' with 3 objects
xfs_io -f -c "pwrite -S 0x61 0       65536" a
xfs_io -f -c "pwrite -S 0x62 65536   65536" a
xfs_io -f -c "pwrite -S 0x63 131072  65536" a
xfs_io -f -c "pwrite -S 0x64 0      196608" b

# truncate 'b'
xfs_io -c "truncate 0" b
# copy 'a' into 'b'
xfs_io -c "copy_range -s 0 -d 0 -l 196608 a" b
```

at this point the contents of 'b' is:

```
hexdump b
0000000 0000 0000 0000 0000 0000 0000 0000 0000
*
0030000
```

If I write 'b' again with 'xfs_io -f -c "pwrite -S 0x64 0 196608" b' it's content is restored, but running the copy_range results again in a file with zeros.

Initially I thought this was a bug in the copy_file_range, but at this point it looks more an issue with the TRUNC operation, as the remote copy operation will always result in zero'ed objects. Could this be a MDS bug? Or OSD...?

History

#1 - 11/23/2018 11:33 AM - Luis Henriques

I forgot to mention that using the 'rados' command I'm able to see that the objects in the data pool actually seem to be correct. I.e., after the truncate I

see that the 3 objects for 'b' are truncated; after doing the copy_range, I can see the new copied data in those objects.

#2 - 11/23/2018 05:36 PM - Luis Henriques

I don't fully understand the following code, but I suspect the issue could be related to truncate_seq in this OSD function:

```
int PrimaryLogPG::do_read(OpContext *ctx, OSDOp& osd_op) {
  dout(20) << __func__ << dendl;
  auto& op = osd_op.op;
  auto& oi = ctx->new_obs.oi;
  auto& soid = oi.soid;
  __u32 seq = oi.truncate_seq;
  uint64_t size = oi.size;
  bool trimmed_read = false;

  dout(30) << __func__ << " oi.size: " << oi.size << dendl;
  dout(30) << __func__ << " oi.truncate_seq: " << oi.truncate_seq << dendl;
  dout(30) << __func__ << " op.extent.truncate_seq: " << op.extent.truncate_seq << dendl;
  dout(30) << __func__ << " op.extent.truncate_size: " << op.extent.truncate_size << dendl;

  // are we beyond truncate_size?
  if ( (seq < op.extent.truncate_seq) &&
       (op.extent.offset + op.extent.length > op.extent.truncate_size) &&
       (size > op.extent.truncate_size) )
    size = op.extent.truncate_size;

  if (op.extent.length == 0) //length is zero mean read the whole object
    op.extent.length = size;

  if (op.extent.offset >= size) {
    op.extent.length = 0;
    trimmed_read = true;
  } else if (op.extent.offset + op.extent.length > size) {
    op.extent.length = size - op.extent.offset;
    trimmed_read = true;
  }

  dout(30) << __func__ << "op.extent.length is now " << op.extent.length << dendl;
...
```

I see that op.extent.length is zero in the above debug message when I try to read file 'b'.

But it's Friday and I can't go any further digging on the OSDs code :-)

#3 - 11/26/2018 04:52 PM - Luis Henriques

Anyone more knowledgable with OSD code could please confirm if the following PrimaryLogPG::do_read() patch makes sense?

```
@@ -5438,7 +5438,7 @@ int PrimaryLogPG::do_read(OpContext *ctx, OSDOp& osd_op) {
   dout(30) << __func__ << " op.extent.truncate_size: " << op.extent.truncate_size << dendl;

   // are we beyond truncate_size?
-  if ( (seq < op.extent.truncate_seq) &&
+  if ( seq && (seq < op.extent.truncate_seq) &&
         (op.extent.offset + op.extent.length > op.extent.truncate_size) &&
         (size > op.extent.truncate_size) )
     size = op.extent.truncate_size;
```

oi.truncate_seq ('seq' variable) seems to make sense only if different from zero. This extra check is done in CEPH_OSD_OP_WRITE as well (see PrimaryLogPG::do_osd_ops).

I've done a quick test and it seems to fix the issue. But the patch may be completely wrong, as this is my first incursion in the OSD code!

#4 - 11/26/2018 11:28 PM - Patrick Donnelly

- Project changed from CephFS to RADOS

- Subject changed from Failure to do remote object copy after a truncate to osd: failure to do remote object copy after a truncate

- Priority changed from Normal to High

- Target version set to v14.0.0

- Source set to Development

- Backport set to mimic,luminous

- Component(RADOS) OSD added

Moving to RADOS since the problem appears to be there.

#5 - 11/27/2018 10:18 AM - Luis Henriques

Patrick Donnelly wrote:

> Moving to RADOS since the problem appears to be there.

Thanks, Patrick. In the meantime I've created PR#25277 with the patch above, although I'm still not 100% about it.

#6 - 11/27/2018 03:18 PM - Greg Farnum

From what you're showing here, it looks like that patch is effectively just disabling the OSD's truncate_seq check in this test.
If you issue a sync command between the file create and issuing the truncate command, does Ceph behave correctly without the patch? That would make me think the kernel client is not handling truncates correctly when the backing RADOS objects haven't been created yet.

#7 - 11/27/2018 03:19 PM - Greg Farnum

- Project changed from RADOS to CephFS

- Subject changed from osd: failure to do remote object copy after a truncate to truncate_seq ordering issues with object creation

#8 - 11/27/2018 03:39 PM - Luis Henriques

Greg Farnum wrote:

> From what you're showing here, it looks like that patch is effectively just disabling the OSD's truncate_seq check in this test.
> If you issue a sync command between the file create and issuing the truncate command, does Ceph behave correctly without the patch? That would make me think the kernel client is not handling truncates correctly when the backing RADOS objects haven't been created yet.

No, a sync command doesn't help (I event tried a umount to make sure). Something I forgot to mention is that if I do a write(2) to the file, I am able to read the data I just wrote; but if I run again the copy_range command I'll get again only zeros. That's what made me suspicious about the OSDs. But again, I'm not claiming my patch is correct -- I just assumed that, since write(2) is working correctly, the read(2) could be missing the extra check that is present in commit e72cc23
("truncate: don't write beyong truncation with old trunc seq")

#9 - 11/27/2018 07:02 PM - Greg Farnum

Oh hrm. That does make it more interesting.

...oh my. I bet that when you do a copy-from op on the OSD side, the truncate_seq value is either left at the default 0, or else is copied from the base objects. But in this case it needs to be kept at the old value or incremented, since of course it's replacing old data!

We may not have the interfaces built in yet to support this correctly. :(

#10 - 11/28/2018 11:40 AM - Luis Henriques

Greg Farnum wrote:

> Oh hrm. That does make it more interesting.
>
> ...oh my. I bet that when you do a copy-from op on the OSD side, the truncate_seq value is either left at the default 0, or else is copied from the base objects. But in this case it needs to be kept at the old value or incremented, since of course it's replacing old data!
>
> We may not have the interfaces built in yet to support this correctly. :(

Looks like the truncate_seq is indeed being copied from the base object:

```
void PrimaryLogPG::finish_copyfrom(CopyFromCallback *cb)
{
...
  obs.oi.truncate_seq = cb->results->truncate_seq;
  obs.oi.truncate_size = cb->results->truncate_size;
...
}
```

I've done a quick hack to verify these values and to test it when keeping the old values (i.e. without copying the base object values) -- it seems to work as expected. For my test case, of course :-) I couldn't figure out any other side effects of this hack.

#11 - 11/28/2018 12:24 PM - Zheng Yan

It seems that CEPH_OSD_OP_COPY_FROM is designed for cache tier, not suite for general use. The problem happen if src inode's truncate seq is smaller than dest inode's truncate seq

#12 - 11/28/2018 02:04 PM - Luis Henriques

Would it be acceptable to do something like adding an extra CEPH_OSD_COPY_FROM_FLAG_IGNORE_TRUNCATE_SEQ flag? The kernel client could use this flag to ensure that we would skip this truncate_seq copy from the base object, and instead increment the truncate_seq in the copied object.

(Well, the problem with that would be that we wouldn't know if the OSD actually supports it... so this wouldn't really be enough.)

#13 - 11/28/2018 02:41 PM - Zheng Yan

For now, I think we can add a check to kernel client, make sure src inode and dest inode's truncate seq are the same.

Next step is introduce a new osd op or update OP_COPY_FROM.

#14 - 11/28/2018 11:01 PM - Luis Henriques

Zheng Yan wrote:

> For now, I think we can add a check to kernel client, make sure src inode and dest inode's truncate seq are the same.

I'm not sure I understand what you mean. Does it make sense to compare that field for 2 different inodes? Something like this:

```
@@ -1927,6 +1927,9 @@ static ssize_t ceph_copy_file_range(struct file *src_file, loff_t src_off,
        if (len < src_ci->i_layout.object_size)
                return -EOPNOTSUPP; /* no remote copy will be done */

+       if (src_ci->i_truncate_seq != dst_ci->i_truncate_seq)
+               return -EOPNOTSUPP;
+
        prealloc_cf = ceph_alloc_cap_flush();
        if (!prealloc_cf)
                return -ENOMEM;
```

This would fallback to a local copy instead of remote object copies. Is this what you mean? Because it doesn't seem to make sense to do this comparison. But I can't say I fully understand the usage of truncate_seq.

> Next step is introduce a new osd op or update OP_COPY_FROM.

Yeah, fixing copy-from would be my preference, of course, but I obviously need a few more years looking at the OSD code before I can be of any use for that :-)

#15 - 11/29/2018 01:43 AM - Zheng Yan

yes, it's what I mean. besides, we should do the check after getting RW caps of src/dest inode

#16 - 11/29/2018 02:22 PM - Luis Henriques

- File copy-from.patch added

Zheng Yan wrote:

> yes, it's what I mean. besides, we should do the check after getting RW caps of src/dest inode

Ok, thank you for your help. I can definitely submit that patch to the kernel client.

However, I'm attaching here a quick hack that, from the tests I've done, fixes this issue on the OSD side. Basically, it introduces a new flag (as I suggested in comment 12) for the copy-from operation that should be used only by the filesystem clients. It basically skips the truncate_seq copy if this flag is set in the Op. Would this fix make sense? (Please forgive me my C++ illiteracy.)

#17 - 11/29/2018 06:54 PM - Luis Henriques

One thing I forgot to mention in my last comment is that we would also need a new CEPH_FEATURE_<something> to make sure a client would know if the copy-from operation can be used or not.

Comments are welcome. And please let me know if you would rather have this discussed on a PR.

#18 - 11/30/2018 08:52 PM - Greg Farnum

Luis Henriques wrote:

> However, I'm attaching here a quick hack that, from the tests I've done, fixes this issue on the OSD side. Basically, it introduces a new flag (as I suggested in comment 12) for the copy-from operation that should be used only by the filesystem clients. It basically skips the truncate_seq copy if this flag is set in the Op. Would this fix make sense? (Please forgive me my C++ illiteracy.)

I think what we want is an option that lets you set the truncate_seq to use during the copy, rather than blindly preserving it or copying it. Isn't that necessary for the CephFS protocols around it?

Since this only needs to be advertised from servers out to the clients, it should be simple enough to generate one that uses the same actual bit as SERVER_NAUTILUS, so that won't be a big deal unless it needs to be backported (seems unlikely?). :)

#19 - 11/30/2018 08:52 PM - Greg Farnum

And a PR is probably the best way to develop new code rather than diagnose the issue, yeah!

#20 - 12/03/2018 11:37 AM - Luis Henriques

Greg Farnum wrote:

> Luis Henriques wrote:
>
> > However, I'm attaching here a quick hack that, from the tests I've done, fixes this issue on the OSD side. Basically, it introduces a new flag (as I suggested in comment 12) for the copy-from operation that should be used only by the filesystem clients. It basically skips the truncate_seq copy if this flag is set in the Op. Would this fix make sense? (Please forgive me my C++ illiteracy.)
>
> I think what we want is an option that lets you set the truncate_seq to use during the copy, rather than blindly preserving it or copying it. Isn't that necessary for the CephFS protocols around it?

My understanding of the truncate_seq usage is that it does not make sense to copy it from the base object for this cephfs use-case; also, since we're not executing a truncate operation, I don't see why we would want to change it's value. That's why I'm proposing an option to not change it's value for this usage scenario. Is there really any scenario where we would want to set truncate_seq when performing an object copy? If I understand you correctly, what you're asking is a change to struct ceph_osd_op so that it would include 2 new fields (truncate_size and truncate_seq) in the copy_from (in rados.h).

> Since this only needs to be advertised from servers out to the clients, it should be simple enough to generate one that uses the same actual bit as SERVER_NAUTILUS, so that won't be a big deal unless it needs to be backported (seems unlikely?). :)

Ah, that's a very good point.

#21 - 12/03/2018 12:20 PM - Luis Henriques

Greg Farnum wrote:

> And a PR is probably the best way to develop new code rather than diagnose the issue, yeah!

Here's the PR: https://github.com/ceph/ceph/pull/25374

#22 - 03/07/2019 11:22 PM - Patrick Donnelly

- Target version changed from v14.0.0 to v15.0.0

#23 - 03/07/2019 11:32 PM - Patrick Donnelly

- Target version deleted (v15.0.0)

#24 - 04/02/2019 03:24 PM - Luis Henriques

- File 0001-add-support-for-osd-copy-file-feature.patch added

Just a quick update: I've pushed another version of my copy_from/truncate OSD fix into https://github.com/ceph/ceph/pull/25374. I'm also attaching the patch I'm using to actually test it in a kernel CephFS client.

#25 - 11/01/2019 04:58 PM - Sage Weil

- Status changed from New to Resolved

- Backport deleted (mimic,luminous)

#26 - 12/26/2019 10:01 AM - Kefu Chai

- Pull request ID set to 31728

Files

copy-from.patch	1.28 KB	11/29/2018	Luis Henriques
0001-add-support-for-osd-copy-file-feature.patch	5.88 KB	04/02/2019	Luis Henriques