bluestore - Bug #48036

bluefs corrupted in a OSD

10/29/2020 06:20 AM - Satoru Takeuchi

Status:	Closed	**% Done:**	0%
Priority:	Low		
Assignee:			
Category:			
Target version:			
Source:		**Affected Versions:**	
Tags:	BlueStore	**ceph-qa-suite:**	
Backport:		**Pull request ID:**	
Regression:	No	**Crash signature (v1):**	
Severity:	3 - minor	**Crash signature (v2):**	
Reviewed:			

Description

I hit a problem that is very similar to the following issue/PR in v15.2.4.

upgrade/nautilus-x-master: bluefs mount failed to replay log: (14) Bad address during upgrade
https://tracker.ceph.com/issues/46886

os/bluestore: get rid of obsolete stuff in bluefs. #36745 (it's not backported to Octopus)
https://github.com/ceph/ceph/pull/36745

In addition, This PR is not backported to Octopus. If my problem is the same as issue#46886, please backport PR#36754 to Octopus.

Here is the detail.

"ceph-bluestore-tool bluefs-bdev-expand --path <osdDataPath>" failed with the following log (the full log is attached as "osd-9-bluefs.txt").

```
2020-10-28T02:07:55.499+0000 7f0aa7f1e0c0 -1 bluefs _check_new_allocations invalid extent 1: 0x7ae18a0000~10000: duplicate reference, ino 26
2020-10-28T02:07:55.500+0000 7f0aa7f1e0c0 -1 bluefs mount failed to replay log: (14) Bad address
2020-10-28T02:07:55.500+0000 7f0aa7f1e0c0 -1 bluestore(/var/lib/ceph/osd/ceph-9) _open_bluefs failed bluefs mount: (14) Bad address
```

The tail of the bluefs's journal log is as follows (the full log is attached as "bluefs-dump.zip").

```
...
2020-10-28T08:39:58.788+0000 7f79675c70c0 10 bluefs _replay 0x263000: txn(seq 51924 len 0xc8 crc 0x4314d6b)
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x263000:  op_dir_unlink  db/CURRENT
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x263000:  op_file_remove 25
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x263000:  op_dir_link  db/CURRENT to 30
2020-10-28T08:39:58.788+0000 7f79675c70c0 30 bluefs _get_file ino 30 = 0x56375176dad0
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x263000:  op_dir_unlink  db/000020.dbtmp
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x263000:  op_file_update  file(ino 31 size 0x0 mtime 2020-10-27T23:10:42.718117+0000 allocated 0 extents [])
2020-10-28T08:39:58.788+0000 7f79675c70c0 30 bluefs _get_file ino 31 = 0x56375176c0d0 (new)
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x263000:  op_dir_link  db.wal/000021.log to 31
2020-10-28T08:39:58.788+0000 7f79675c70c0 30 bluefs _get_file ino 31 = 0x56375176c0d0
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x263000:  op_dir_unlink  db.wal/000015.log
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x263000:  op_file_remove 26                          ... (1)
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x263000:  op_dir_unlink  db/MANIFEST-000014
```

```
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x263000:  op_file_remove 24
2020-10-28T08:39:58.788+0000 7f79675c70c0 10 bluefs _read h 0x563751752d80 0x264000~1000 from file(ino 1 size 0x264000
mtime 0.000000 allocated 410000 extents [1:0x7ae18b0000~10000,1:0x7ae1470000~400000])
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _read left 0xac000 len 0x1000
2020-10-28T08:39:58.788+0000 7f79675c70c0 30 bluefs _read result chunk (0x1000 bytes):
00000000  01 01 80 00 00 00 1c 67  9e 3f 31 cb 40 5b bc f4  |.......g.?1.@[..|
00000010  49 4f 77 68 e7 24 d5 ca  00 00 00 00 00 00 60 00  |IOwh.$........`.|
00000020  00 00 08 01 01 0f 00 00  00 20 00 f2 a8 98 5f 7c  |......... ...._||
00000030  5b ec 36 00 00 00 00 00  04 02 00 00 00 64 62 14  |[.6..........db.|
00000040  00 00 00 4f 50 54 49 4f  4e 53 2d 30 30 30 30 32  |...OPTIONS-00002|
00000050  32 2e 64 62 74 6d 70 20  00 00 00 00 00 00 00 08  |2.dbtmp ........|
00000060  01 01 1c 00 00 00 20 d6  26 f2 a8 98 5f 30 83 f2  |...... .&..._0..|
00000070  3a 00 01 00 00 00 01 01  06 00 00 00 29 86 eb 01  |:...........)...|
00000080  43 01 29 e6 5d fa 00 00  00 00 00 00 00 00 00 00  |C.).]...........|
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| *
00000ff0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00001000

2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _read got 4096
2020-10-28T08:39:58.788+0000 7f79675c70c0 10 bluefs _replay 0x264000: txn(seq 51925 len 0x60 crc 0xfa5de629)
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x264000:  op_file_update  file(ino 32 size 0x0 mtime
2020-10-27T23:10:42.921459+0000 allocated 0 extents [])
2020-10-28T08:39:58.788+0000 7f79675c70c0 30 bluefs _get_file ino 32 = 0x56375176cd00 (new)
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x264000:  op_dir_link  db/OPTIONS-000022.dbtmp to 32
2020-10-28T08:39:58.788+0000 7f79675c70c0 30 bluefs _get_file ino 32 = 0x56375176cd00
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x264000:  op_file_update  file(ino 32 size 0x1356 mtime
2020-10-27T23:10:42.988971+0000 allocated 10000 extents [1:0x7ae18a0000~10000])   ... (2)
2020-10-28T08:39:58.788+0000 7f79675c70c0 30 bluefs _get_file ino 32 = 0x56375176cd00
2020-10-28T08:39:58.788+0000 7f79675c70c0 10 bluefs _read h 0x563751752d80 0x265000~1000 from file(ino 1 size 0x265000
mtime 0.000000 allocated 410000 extents [1:0x7ae18b0000~10000,1:0x7ae1470000~400000])
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _read left 0xab000 len 0x1000
2020-10-28T08:39:58.788+0000 7f79675c70c0 30 bluefs _read result chunk (0x1000 bytes):
00000000  01 01 82 00 00 00 1c 67  9e 3f 31 cb 40 5b bc f4  |.......g.?1.@[..|
00000010  49 4f 77 68 e7 24 d6 ca  00 00 00 00 00 00 62 00  |IOwh.$........b.|
00000020  00 00 04 02 00 00 00 64  62 0e 00 00 00 4f 50 54  |.......db....OPT|
00000030  49 4f 4e 53 2d 30 30 30  30 32 33 20 00 00 00 00  |IONS-000023 ....|
00000040  00 00 00 05 02 00 00 00  64 62 14 00 00 00 4f 50  |........db....OP|
00000050  54 49 4f 4e 53 2d 30 30  30 30 32 32 2e 64 62 74  |TIONS-000022.dbt|
00000060  6d 70 05 02 00 00 00 64  62 0e 00 00 00 4f 50 54  |mp.....db....OPT|
00000070  49 4f 4e 53 2d 30 30 30  30 31 34 09 16 00 00 00  |IONS-000014.....|
00000080  00 00 00 00 69 19 3e 68  00 00 00 00 00 00 00 00  |....i.>h........|
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| *
00000ff0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00001000

2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _read got 4096
2020-10-28T08:39:58.788+0000 7f79675c70c0 10 bluefs _replay 0x266000: txn(seq 51927 len 0x4b crc 0x9ae4c5e9)
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x266000:  op_file_update  file(ino 26 size 0x24e934 mtime
2020-10-23T08:29:44.981029+0000 allocated 250000 extents [1:0x7ae18a0000~10000,1:0x7ae18c0000~
10000,1:0x7ae1870000~20000,1:0x7ae1e10000~210000])
2020-10-28T08:39:58.788+0000 7f79675c70c0 30 bluefs _get_file ino 26 = 0x56375176cea0 (new)
2020-10-28T08:39:58.788+0000 7f79675c70c0 -1 bluefs _check_new_allocations invalid extent 1: 0x7ae18a0000~10000: duplicate
reference, ino 26           ... (3)
2020-10-28T08:39:58.788+0000 7f79675c70c0 -1 bluefs mount failed to replay log: (14) Bad address
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _stop_alloc
```

It looks this bluefs corrupted after the following steps.

(1) delete ino 26
(2) allocate "0x7ae18a0000~10000" to ino 32
(3) allocate the same extent to already-removed ino 26 and detect duplicate reference.

In addition, the mtime of the last op_file_update of ino 26 is "2020-10-23T08:29:44.981029+0000".

```
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x266000:  op_file_update  file(ino 26 size 0x24e934 mtime
2020-10-23T08:29:44.981029+0000 allocated 250000 extents [1:0x7ae18a0000~10000,1:0x7ae18c0000~
```

However, it's far older than the previous ones.

```
2020-10-28T08:39:58.788+0000 7f79675c70c0 20 bluefs _replay 0x264000:  op_file_update  file(ino 32 size 0x1356 mtime
2020-10-27T23:10:42.988971+0000 allocated 10000 extents [1:0x7ae18a0000~10000])
```

History

#1 - 10/30/2020 10:41 AM - Igor Fedotov

Both
https://tracker.ceph.com/issues/46886
and https://github.com/ceph/ceph/pull/36745
were following up the https://github.com/ceph/ceph/pull/30838

And the latter is present in master(pacific) only. Hence there is no need to for backports to Octopus.

#2 - 10/30/2020 10:41 AM - Igor Fedotov

As far as I can see you're attempting to expand DB volume, weren't you? Any rationale for that?
Wasn't that a volume squeeze by some chance?

Please set debug-bluestore & debug-bluefs to 20 and collect OSD startup log.

#3 - 10/30/2020 10:56 AM - Igor Fedotov

Igor Fedotov wrote:

> Please set debug-bluestore & debug-bluefs to 20 and collect OSD startup log.

Never mind, attached one is good enough for now...

#4 - 11/02/2020 12:01 AM - Satoru Takeuchi

> As far as I can see you're attempting to expand DB volume, weren't you? Any rationale for that?
> Wasn't that a volume squeeze by some chance?

I use Rook to deploy Ceph cluster. When starting OSD, Rook always tries to expand volume by `ceph-bluestore-tool bluefs-bdev-expand`.

Rook can configure OSD's size in a YAML file and if users update the size in the configuration file, Rook restarts OSD and then the OSD is expanded. Even if the configuration file is not changed, the above-mentioned expansion command is called anyway when restarting OSD. Rook developers expect this command is noop when the size is not changed.

#5 - 11/02/2020 11:27 AM - Satoru Takeuchi

I succeeded to reproduce this problem in my Rook/Ceph cluster.

https://github.com/rook/rook/issues/6530

I guess that calling `ceph-bluestore-tool bluefs-bdev-expand` corrupts OSD if it's in an inconsistent state. Although this problem would happen in Rook relatively easier than in other Ceph orchestrations, it can happen in all Ceph cluster.

#6 - 11/02/2020 12:10 PM - Igor Fedotov

Hi Satoru,
thanks for the update.
Nevertheless I'm not completely sure whether bluefs-bdev-expand is a trigger for the issue or it's just the first "sensor" to detect it.

Your initial analysis about ino 26 being removed and later reused is very helpful and indicative. Wondering if the same pattern applies for your repro with rook?

The next step would be to investigate which stuff triggers this "broken" log sequence - there should be no op_file_update entry after file delete... This is definitely a bug. But again I'm not sure it's bluefs-bdev-expand which causes it...

#7 - 11/02/2020 12:25 PM - Satoru Takeuchi

> Nevertheless I'm not completely sure whether bluefs-bdev-expand is a trigger for the issue or it's just the first "sensor" to detect it.

I agree with you.

> Your initial analysis about ino 26 being removed and later reused is very helpful and indicative. Wondering if the same pattern applies for your repro with rook?

I'll analyze the journal log after running my reproducer in Rook.

> The next step would be to investigate which stuff triggers this "broken" log sequence - there should be no op_file_update entry after file delete... This is definitely a bug. But again I'm not sure it's bluefs-bdev-expand which causes it...

OK, please let me know if you want more information other than the above-mentioned investigation.

#8 - 11/02/2020 12:50 PM - Satoru Takeuchi

As you suspect, `bluefs-bdev-expand` seems to be the first sensor. After running the reproducer with my custom Rook, that doesn't call `bluefs-bdev-expand`, starting OSD daemon failed with the following log.

```
debug 2020-11-02T12:40:38.929+0000 7f9521355f40  0 set uid:gid to 167:167 (ceph:ceph)
debug 2020-11-02T12:40:38.929+0000 7f9521355f40  0 ceph version 15.2.4 (7447c15c6ff58d7fce91843b705a268a1917325c) octopus (stable), process ceph-osd, pid 1
debug 2020-11-02T12:40:38.929+0000 7f9521355f40  0 pidfile_write: ignore empty --pid-file
debug 2020-11-02T12:40:38.933+0000 7f9521355f40  1 bdev create path /var/lib/ceph/osd/ceph-0/block type kernel
debug 2020-11-02T12:40:38.933+0000 7f9521355f40  1 bdev(0x561c4f6f8380 /var/lib/ceph/osd/ceph-0/block) open path /var/lib/ceph/osd/ceph-0/block
debug 2020-11-02T12:40:38.933+0000 7f9521355f40  1 bdev(0x561c4f6f8380 /var/lib/ceph/osd/ceph-0/block) open size 6442450944 (0x180000000, 6 GiB) block_size 4096 (4 KiB) rotational discard supported
debug 2020-11-02T12:40:38.933+0000 7f9521355f40  1 bluestore(/var/lib/ceph/osd/ceph-0) _set_cache_sizes cache_size 1073741824 meta 0.4 kv 0.4 data 0.2
debug 2020-11-02T12:40:38.933+0000 7f9521355f40  1 bdev create path /var/lib/ceph/osd/ceph-0/block type kernel
debug 2020-11-02T12:40:38.933+0000 7f9521355f40  1 bdev(0x561c4f6f8a80 /var/lib/ceph/osd/ceph-0/block) open path /var/lib/ceph/osd/ceph-0/block
debug 2020-11-02T12:40:38.933+0000 7f9521355f40  1 bdev(0x561c4f6f8a80 /var/lib/ceph/osd/ceph-0/block) open size 6442450944 (0x180000000, 6 GiB) block_size 4096 (4 KiB) rotational discard supported
debug 2020-11-02T12:40:38.933+0000 7f9521355f40  1 bluefs add_block_device bdev 1 path /var/lib/ceph/osd/ceph-0/block size 6 GiB
debug 2020-11-02T12:40:38.933+0000 7f9521355f40  1 bdev(0x561c4f6f8a80 /var/lib/ceph/osd/ceph-0/block) close
debug 2020-11-02T12:40:39.249+0000 7f9521355f40  1 bdev(0x561c4f6f8380 /var/lib/ceph/osd/ceph-0/block) close
debug 2020-11-02T12:40:39.485+0000 7f9521355f40  0 starting osd.0 osd_data /var/lib/ceph/osd/ceph-0 /var/lib/ceph/osd/ceph-0/journal
debug 2020-11-02T12:40:39.485+0000 7f9521355f40 -1 Falling back to public interface
debug 2020-11-02T12:40:39.493+0000 7f9521355f40  0 load: jerasure load: lrc load: isa
debug 2020-11-02T12:40:39.493+0000 7f9521355f40  1 bdev create path /var/lib/ceph/osd/ceph-0/block type kernel
debug 2020-11-02T12:40:39.493+0000 7f9521355f40  1 bdev(0x561c4f6f9180 /var/lib/ceph/osd/ceph-0/block) open path /var/lib/ceph/osd/ceph-0/block
debug 2020-11-02T12:40:39.493+0000 7f9521355f40  1 bdev(0x561c4f6f9180 /var/lib/ceph/osd/ceph-0/block) open size 6442450944 (0x180000000, 6 GiB) block_size 4096 (4 KiB) rotational discard supported
debug 2020-11-02T12:40:39.493+0000 7f9521355f40  1 bluestore(/var/lib/ceph/osd/ceph-0) _set_cache_sizes cache_size 1073741824 meta 0.4 kv 0.4 data 0.2
debug 2020-11-02T12:40:39.493+0000 7f9521355f40  1 bdev(0x561c4f6f9180 /var/lib/ceph/osd/ceph-0/block) close
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  0 osd.0:0.OSDShard using op scheduler ClassedOpQueueScheduler(queue=WeightedPriorityQueue, cutoff=196)
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  0 osd.0:1.OSDShard using op scheduler ClassedOpQueueScheduler(queue=WeightedPriorityQueue, cutoff=196)
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  0 osd.0:2.OSDShard using op scheduler ClassedOpQueueScheduler(queue=WeightedPriorityQueue, cutoff=196)
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  0 osd.0:3.OSDShard using op scheduler ClassedOpQueueScheduler(queue=WeightedPriorityQueue, cutoff=196)
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  0 osd.0:4.OSDShard using op scheduler ClassedOpQueueScheduler(queue=WeightedPriorityQueue, cutoff=196)
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  1 bluestore(/var/lib/ceph/osd/ceph-0) _mount path /var/lib/ceph/osd/ceph-0
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  1 bdev create path /var/lib/ceph/osd/ceph-0/block type kernel
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  1 bdev(0x561c4f6f9180 /var/lib/ceph/osd/ceph-0/block) open path /var/lib/ceph/osd/ceph-0/block
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  1 bdev(0x561c4f6f9180 /var/lib/ceph/osd/ceph-0/block) open size 6442450944 (0x180000000, 6 GiB) block_size 4096 (4 KiB) rotational discard supported
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  1 bluestore(/var/lib/ceph/osd/ceph-0) _set_cache_sizes cache_size 1073741824 meta 0.4 kv 0.4 data 0.2
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  1 bdev create path /var/lib/ceph/osd/ceph-0/block type kernel
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  1 bdev(0x561c4f6f8a80 /var/lib/ceph/osd/ceph-0/block) open path /var/lib/ceph/osd/ceph-0/block
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  1 bdev(0x561c4f6f8a80 /var/lib/ceph/osd/ceph-0/block) open size 6442450944 (0x180000000, 6 GiB) block_size 4096 (4 KiB) rotational discard supported
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  1 bluefs add_block_device bdev 1 path /var/lib/ceph/osd/ceph-0/block size 6 GiB
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  1 bluefs mount
debug 2020-11-02T12:40:39.797+0000 7f9521355f40  1 bluefs _init_alloc id 1 alloc_size 0x10000 size 0x180000000
debug 2020-11-02T12:40:39.801+0000 7f9521355f40 -1 bluefs _check_new_allocations invalid extent 1: 0xa0450000~30000: duplicate reference, ino 32
debug 2020-11-02T12:40:39.801+0000 7f9521355f40 -1 bluefs mount failed to replay log: (14) Bad address
debug 2020-11-02T12:40:39.801+0000 7f9521355f40 -1 bluestore(/var/lib/ceph/osd/ceph-0) _open_bluefs failed bluefs mount: (14) Bad address
debug 2020-11-02T12:40:39.801+0000 7f9521355f40  1 bdev(0x561c4f6f9180 /var/lib/ceph/osd/ceph-0/block) close
debug 2020-11-02T12:40:40.085+0000 7f9521355f40 -1 osd.0 0 OSD:init: unable to mount object store
debug 2020-11-02T12:40:40.085+0000 7f9521355f40 -1  ** ERROR: osd init failed: (14) Bad address
```

I'll investigate this OSD's journal log too.

#9 - 11/02/2020 03:34 PM - Igor Fedotov

@Satoru,
given you're able to reproduce the issue locally would you be able to collect OSD log (with debug-bluefs = 20) BEFORE the first corruption is detected?

If such a log runs too large please share last 100000 lines of it. or more specifically - part of it where ino is question is being removed and resurrected

Thanks in advance!

#10 - 11/06/2020 11:14 AM - Satoru Takeuchi

- File ceph-0.log added

> > Your initial analysis about ino 26 being removed and later reused is very helpful and indicative. Wondering if the same pattern applies for your repro with rook?
>
> I'll analyze the journal log after running my reproducer in Rook.

The same pattern is found in the broken OSD created by the reproducer (the full log is attached as ceph-0.log).

```
...
2020-11-06T05:23:39.789+0000 7f49a2ba50c0 20 bluefs read got 4096
2020-11-06T05:23:39.789+0000 7f49a2ba50c0 10 bluefs _replay 0x59000: txn(seq 90 len 0x32 crc 0x58a9cf2b)
2020-11-06T05:23:39.789+0000 7f49a2ba50c0 20 bluefs _replay 0x59000:  op_file_update  file(ino 26 size 0x5fbf5 mtime
2020-11-05T08:51:11.626129+0000 allocated 60000 extents [1:0xa04a0000~30000,1:0xa04e0000~30000])
2020-11-06T05:23:39.789+0000 7f49a2ba50c0 30 bluefs _get_file ino 26 = 0x55ccae0d45b0
2020-11-06T05:23:39.789+0000 7f49a2ba50c0 10 bluefs _read h 0x55ccae0bad80 0x5a000~1000 from file(ino 1 size 0x5a000 mtime 0.000000
allocated 400000 extents [1:0xa0000000~400000])
2020-11-06T05:23:39.789+0000 7f49a2ba50c0 20 bluefs _read left 0xa6000 len 0x1000
2020-11-06T05:23:39.789+0000 7f49a2ba50c0 30 bluefs _read result chunk (0x1000 bytes):
00000000  01 01 52 00 00 00 cb c2  67 e3 4f 0b 4b 98 9c 4b  |..R.....g.O.K..K|
00000010  3a 16 8f 3e be 30 5b 00  00 00 00 00 00 00 32 00  |:..>.0[.......2.|
00000020  00 00 08 01 01 2b 00 00  00 20 ce de 11 fc bc a3  |.....+... ......|
00000030  5f f4 f4 c6 1a 00 02 00  00 00 01 01 07 00 00 00  |...............|
00000040  15 81 02 00 c3 01 01 01  01 07 00 00 00 29 81 02  |.............)..|
00000050  00 83 01 01 d5 49 ae 9f  00 00 00 00 00 00 00 00  |.....I..........|
00000060  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| *
00000ff0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00001000

2020-11-06T05:23:39.789+0000 7f49a2ba50c0 20 bluefs _read got 4096
2020-11-06T05:23:39.789+0000 7f49a2ba50c0 10 bluefs _replay 0x5a000: txn(seq 91 len 0x32 crc 0x9fae49d5)
2020-11-06T05:23:39.789+0000 7f49a2ba50c0 20 bluefs _replay 0x5a000:  op_file_update  file(ino 32 size 0x46f4e mtime
2020-11-05T08:51:08.449246+0000 allocated 50000 extents [1:0xa0450000~30000,1:0xa04a0000~20000])
2020-11-06T05:23:39.789+0000 7f49a2ba50c0 30 bluefs _get_file ino 32 = 0x55ccae0d5040 (new)
2020-11-06T05:23:39.789+0000 7f49a2ba50c0 -1 bluefs _check_new_allocations invalid extent 1: 0xa0450000~30000: duplicate reference, ino 32
2020-11-06T05:23:39.789+0000 7f49a2ba50c0 -1 bluefs mount failed to replay log: (14) Bad address
2020-11-06T05:23:39.789+0000 7f49a2ba50c0 20 bluefs _stop_alloc
```

> given you're able to reproduce the issue locally would you be able to collect OSD log (with debug-bluefs = 20) BEFORE the first corruption is detected?

Unfortunately, after capturing logs, this problem hasn't been reproduced.

In addition, it's easier to cause this problem in OSD backed by fast devices. In my environment, this problem only happened in NVMe-SSD-backed OSD. It didn't happen in HDD-backed-OSD.

#11 - 11/06/2020 12:36 PM - Satoru Takeuchi

> Unfortunately, after capturing logs, this problem hasn't been reproduced.

More precisely, with setting `debug-bluefs = 20`, this problem could not be reproduced.

#12 - 11/09/2020 06:42 AM - Satoru Takeuchi

- File osd_tail.log.gz added

- File bluefs-log-dump.log.gz added

> given you're able to reproduce the issue locally would you be able to collect OSD log (with debug-bluefs = 20) BEFORE the first corruption is detected?

It's done. Attached "osd_tail.log.gz" is the last 20,000 lines of OSD log with setting "deubg-bluefs = 20". The full log can't be attached since it's too big to attach (over 1MB). So please let me know if it's necessary. I'll send it to you in other ways.

"bluefs-log-dump.log.gz" is the result of bluefs-log-dump after reproducing this problem. Here is the summary of my analysys.

- duplication reference happened when creating ino 562. However, it different from the previous log dumps because this file was not deleted just before the last creation.
- mtime flip was detected between the last transaction and the second last transaction.

```
2020-11-09T01:52:56.750+0000 7eff064900c0 20 bluefs _read got 4096
2020-11-09T01:52:56.750+0000 7eff064900c0 10 bluefs _replay 0x4b4000: txn(seq 1205 len 0x26 crc 0x857db64)
2020-11-09T01:52:56.750+0000 7eff064900c0 20 bluefs _replay 0x4b4000:  op_file_update  file(ino 557 size 0x53e25 mtime
2020-11-06T14:51:21.422572+0000 allocated 60000 extents [1:0xa0450000~60000])
2020-11-09T01:52:56.750+0000 7eff064900c0 30 bluefs _get_file ino 557 = 0x5600b4d685b0
2020-11-09T01:52:56.750+0000 7eff064900c0 10 bluefs _read h 0x5600b4d4f100 0x4b5000~1000 from file(ino 1 size 0x4b5000 mtime 0.000000
allocated 800000 extents [1:0xa0000000~400000,1:0xa0510000~400000])
2020-11-09T01:52:56.750+0000 7eff064900c0 20 bluefs _read left 0x4b000 len 0x1000
2020-11-09T01:52:56.750+0000 7eff064900c0 30 bluefs _read result chunk (0x1000 bytes):
00000000  01 01 46 00 00 00 24 5a  02 10 d2 a2 4a 18 8e 7c  |..F...$Z....J..||
00000010  3e 71 91 d1 99 52 b6 04  00 00 00 00 00 00 26 00  |>q...R........&.|
00000020  00 00 08 01 01 1f 00 00  00 b2 04 d9 db 0e d3 62  |...............b|
00000030  a5 5f 47 be c0 0a 00 01  00 00 00 01 01 07 00 00  |._G.............|
00000040  00 29 81 02 00 83 02 01  48 1a a2 dd 00 00 00 00  |.)......H.......|
00000050  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................| *
00000ff0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
00001000

2020-11-09T01:52:56.750+0000 7eff064900c0 20 bluefs _read got 4096
2020-11-09T01:52:56.750+0000 7eff064900c0 10 bluefs _replay 0x4b5000: txn(seq 1206 len 0x26 crc 0xdda21a48)
2020-11-09T01:52:56.750+0000 7eff064900c0 20 bluefs _replay 0x4b5000:  op_file_update  file(ino 562 size 0x3add9 mtime
2020-11-06T14:50:59.180403+0000 allocated 40000 extents [1:0xa04a0000~40000])
2020-11-09T01:52:56.750+0000 7eff064900c0 30 bluefs _get_file ino 562 = 0x5600b4d69ad0 (new)
2020-11-09T01:52:56.750+0000 7eff064900c0 -1 bluefs _check_new_allocations invalid extent 1: 0xa04a0000~40000: duplicate reference, ino 562
2020-11-09T01:52:56.750+0000 7eff064900c0 -1 bluefs mount failed to replay log: (14) Bad address
2020-11-09T01:52:56.750+0000 7eff064900c0 20 bluefs _stop_alloc
```

Here I used raw mode OSD backed by RAM to ease the reproduction. This problem didn't happen with raw mode OSD backed by NVMe SSD.

In the previous reproductions, where "debug-bluefs = 20" was not set, I used the following OSDs.

- Raw mode OSD on Logical Volume on top of dmcrypt on top of NVMe SSD. In this case, this problem happened in the real system without my reproducer.
- raw mode OSD on NVMe SSD or RAM. Here I used my reproducer.

#13 - 11/09/2020 12:31 PM - Igor Fedotov

- Project changed from Ceph to bluestore

- Category deleted (OSD)

#14 - 11/20/2020 02:40 AM - Satoru Takeuchi

@Igor

Do you have any progress?

#15 - 11/23/2020 06:06 PM - Igor Fedotov

Satoru Takeuchi wrote:

> @Igor
>
> Do you have any progress?

Hi Satoru,
sorry for a long response.
At the second investigation attempt I finally realized that we're facing the same file delete race as before. Ino 557 was removed and flushed later on:

```
2020-11-06T14:49:54.497+0000 7f7c01991f40 20 bluefs open_for_write mapping db.wal/000425.log vsel_hint 0
2020-11-06T14:49:54.497+0000 7f7c01991f40 10 bluefs open_for_write h 0x5556ed3895e0 on file(ino 562 size 0x0 mtime
2020-11-06T14:49:54.502499+0000 allocated 0 extents [])
2020-11-06T14:49:54.497+0000 7f7c01991f40 10 bluefs readdir db
2020-11-06T14:49:54.497+0000 7f7c01991f40 10 bluefs readdir db.wal
2020-11-06T14:49:54.497+0000 7f7c01991f40 10 bluefs unlink db.wal/000422.log
2020-11-06T14:49:54.497+0000 7f7c01991f40 20 bluefs _drop_link had refs 1 on file(ino 557 size 0xfdcd mtime 2020-11-06T14:49:43.097027+0000
allocated 10000 extents [1:0xa0450000~10000])
2020-11-06T14:49:54.497+0000 7f7c01991f40 20 bluefs _drop_link destroying file(ino 557 size 0xfdcd mtime 2020-11-06T14:49:43.097027+0000
allocated 10000 extents [1:0xa0450000~10000])

....

2020-11-06T14:50:14.033+0000 7f97749e7700 10 bluefs _flush 0x55cb3c94b420 ignoring, length 563 < min_flush_size 524288
2020-11-06T14:50:14.033+0000 7f97749e7700 10 bluefs _flush 0x55cb3c94b420 ignoring, length 33331 < min_flush_size 524288
2020-11-06T14:50:14.033+0000 7f97749e7700 10 bluefs _flush 0x55cb3c94b420 ignoring, length 34348 < min_flush_size 524288
2020-11-06T14:50:14.033+0000 7f97749e7700 10 bluefs get_usage bdev 1 free 1064697856 (1015 MiB) / 1073741824 (1 GiB), used 0%
2020-11-06T14:50:14.033+0000 7f97749e7700 10 bluefs _flush 0x55cb3c94b420 ignoring, length 34371 < min_flush_size 524288
2020-11-06T14:50:14.033+0000 7f97749e7700 10 bluefs _flush 0x55cb3c94b420 ignoring, length 34371 < min_flush_size 524288
2020-11-06T14:50:14.033+0000 7f97749e7700 10 bluefs _fsync 0x55cb3c94b420 file(ino 557 size 0xfdcd mtime 2020-11-06T14:49:43.097027+0000
allocated 10000 extents [1:0xa0450000~10000])
2020-11-06T14:50:14.033+0000 7f97749e7700 10 bluefs _flush 0x55cb3c94b420 0xfdcd~8643 to file(ino 557 size 0xfdcd mtime
2020-11-06T14:49:43.097027+0000 allocated 10000 extents [1:0xa0450000~10000])
```

Looks like some long lasting task is running on ino 557. But ino 557 is removed by a different thread (presumably by RocksDB) meanwhile and hence corrupted ino is written to bluefs log by the first task.

You mentioned that osd_tail.log was truncated Now I believe I need the full one so could you please send it to me via email or share somewhere.

#16 - 11/23/2020 06:32 PM - Igor Fedotov

@Satoru,
could you please reproduce the issue once again, now with both debug_bluefs set to 20 and debug_bluestore set to 10.

It looks like your BlueStore has two _kv_sync_thread-s which is weird...

#17 - 11/25/2020 01:47 AM - Satoru Takeuchi

@Igor

> You mentioned that osd_tail.log was truncated Now I believe I need the full one so could you please send it to me via email or share somewhere.

Please download the full log from the following URL.

https://drive.google.com/file/d/1sbqcyqDAL98dYyI5yOKr-ho5XYmwX-bl/view?usp=sharing

> could you please reproduce the issue once again, now with both debug_bluefs set to 20 and debug_bluestore set to 10.

I'm trying to reproduce with these options. Please wait for a while.

#18 - 11/25/2020 05:56 AM - Satoru Takeuchi

> > could you please reproduce the issue once again, now with both debug_bluefs set to 20 and debug_bluestore set to 10.
>
> I'm trying to reproduce with these options. Please wait for a while.

Here is the full log.

https://drive.google.com/file/d/1kyxdw9ixK_kpxneUJxnyZRbEIsmn45Yt/view?usp=sharing

And here is the log between the last kill and the second last kill.

https://drive.google.com/file/d/1kyxdw9ixK_kpxneUJxnyZRbEIsmn45Yt/view?usp=sharing

#19 - 11/25/2020 11:58 AM - Igor Fedotov

So my hypothesis about multiple kv_sync_thread-s is confirmed. Here is the log snippet from OSD log:

```
Thread 7faf0eb7a700 is started and running
2020-11-24T15:49:35.726+0000 7faf0eb7a700 10 bluestore(/var/lib/ceph/osd/ceph-0) _kv_sync_thread start
...
2020-11-24T15:49:39.018+0000 7faf0eb7a700 20 bluefs _flush_and_sync_log log_seq_stable 718
2020-11-24T15:49:39.018+0000 7faf0eb7a700 20 bluefs _flush_and_sync_log cleaned file file(ino 415 size 0x1835c mtime
2020-11-24T15:49:39.023514+0000 allocated 20000 extents [1:0xa04a0000~20000])
//////////////section to ignore, some other parallel threads
2020-11-24T15:49:39.018+0000 7faf13b84700 10 bluestore(/var/lib/ceph/osd/ceph-0) _txc_state_proc txc 0x5562cd89f500 kv_submitted
2020-11-24T15:49:39.018+0000 7faf13b84700 10 bluestore(/var/lib/ceph/osd/ceph-0) _txc_state_proc txc 0x5562cd89f500 finishing
2020-11-24T15:49:39.018+0000 7faf13b84700 10 bluestore(/var/lib/ceph/osd/ceph-0) _txc_release_alloc(sync) 0x5562cd89f500 []
2020-11-24T15:49:40.739+0000 7faf13383700  5 bluestore.MempoolThread(0x5562cc5a5a98) _resize_shards cache_size: 2845415832 kv_alloc:
1073741824 kv_used: 114336 meta_alloc: 1073741824 meta_used: 29271 data_alloc: 671088640 data_used: 0
2020-11-24T15:49:40.751+0000 7faf0bd7a700 10 bluestore(/var/lib/ceph/osd/ceph-0) pool_statfsstore_statfs(0x0/0x0/0x0, data 0x0/0x0, compress
0x0/0x0/0x0, omap 0x0, meta 0x0)
2020-11-24T15:49:41.743+0000 7faf13383700 10 bluestore(/var/lib/ceph/osd/ceph-0) _deferred_submit_unlock osr 0x5562cc57a140 4 ios pending
2020-11-24T15:49:41.743+0000 7faf17d92700 10 bluestore(/var/lib/ceph/osd/ceph-0) _deferred_aio_finish osr 0x5562cc57a140
2020-11-24T15:49:45.751+0000 7faf0bd7a700 10 bluestore(/var/lib/ceph/osd/ceph-0) pool_statfsstore_statfs(0x0/0x0/0x0, data 0x0/0x0, compress
0x0/0x0/0x0, omap 0x0, meta 0x0)
2020-11-24T15:49:45.751+0000 7faf13383700  5 bluestore.MempoolThread(0x5562cc5a5a98) _resize_shards cache_size: 2845415832 kv_alloc:
1073741824 kv_used: 114336 meta_alloc: 1073741824 meta_used: 29271 data_alloc: 671088640 data_used: 0
2020-11-24T15:49:50.751+0000 7faf0bd7a700 10 bluestore(/var/lib/ceph/osd/ceph-0) pool_statfsstore_statfs(0x0/0x0/0x0, data 0x0/0x0, compress
0x0/0x0/0x0, omap 0x0, meta 0x0)
2020-11-24T15:49:50.767+0000 7faf13383700  5 bluestore.MempoolThread(0x5562cc5a5a98) _resize_shards cache_size: 2845415832 kv_alloc:
1073741824 kv_used: 114336 meta_alloc: 1073741824 meta_used: 29271 data_alloc: 671088640 data_used: 0
2020-11-24T15:49:55.751+0000 7faf0bd7a700 10 bluestore(/var/lib/ceph/osd/ceph-0) pool_statfsstore_statfs(0x0/0x0/0x0, data 0x0/0x0, compress
0x0/0x0/0x0, omap 0x0, meta 0x0)
2020-11-24T15:49:55.779+0000 7faf13383700  5 bluestore.MempoolThread(0x5562cc5a5a98) _resize_shards cache_size: 2845415832 kv_alloc:
1073741824 kv_used: 114336 meta_alloc: 1073741824 meta_used: 29271 data_alloc: 671088640 data_used: 0
2020-11-24T15:50:00.756+0000 7faf0bd7a700 10 bluestore(/var/lib/ceph/osd/ceph-0) pool_statfsstore_statfs(0x0/0x0/0x0, data 0x0/0x0, compress
0x0/0x0/0x0, omap 0x0, meta 0x0)
2020-11-24T15:50:00.792+0000 7faf13383700  5 bluestore.MempoolThread(0x5562cc5a5a98) _resize_shards cache_size: 2845415832 kv_alloc:
1073741824 kv_used: 114336 meta_alloc: 1073741824 meta_used: 29271 data_alloc: 671088640 data_used: 0
^^^^^^^^^^^^^^^^
new ceph instance or something is started
2020-11-24T15:50:01.624+0000 7f60bf7e4f40  0 set uid:gid to 167:167 (ceph:ceph)
2020-11-24T15:50:01.624+0000 7f60bf7e4f40  0 ceph version 15.2.4 (7447c15c6ff58d7fce91843b705a268a1917325c) octopus (stable), process
ceph-osd, pid 1
...
2020-11-24T15:50:03.708+0000 7f60bf7e4f40 20 bluefs _replay 0x290000:  op_file_update  file(ino 377 size 0x10 mtime
2020-11-24T14:37:22.466440+0000 allocated 10000 extents [1:0xa0450000~10000])
...
Another kv_sync_thread = 7f60a9ad2700
2020-11-24T15:50:04.404+0000 7f60a9ad2700 10 bluestore(/var/lib/ceph/osd/ceph-0) _kv_sync_thread start
...
2020-11-24T15:50:04.404+0000 7f60a9ad2700 10 bluefs _flush 0x55c8422695e0 ignoring, length 78 < min_flush_size 524288
2020-11-24T15:50:04.404+0000 7f60a9ad2700 10 bluefs _flush 0x55c8422695e0 ignoring, length 78 < min_flush_size 524288
2020-11-24T15:50:04.404+0000 7f60a9ad2700 10 bluefs _fsync 0x55c8422695e0 file(ino 420 size 0x0 mtime 2020-11-24T15:50:04.403812+0000
allocated 0 extents [])
2020-11-24T15:50:04.404+0000 7f60a9ad2700 10 bluefs _flush 0x55c8422695e0 0x0~4e to file(ino 420 size 0x0 mtime
2020-11-24T15:50:04.403812+0000 allocated 0 extents [])
2020-11-24T15:50:04.404+0000 7f60a9ad2700 10 bluefs _flush_range 0x55c8422695e0 pos 0x0 0x0~4e to file(ino 420 size 0x0 mtime
2020-11-24T15:50:04.403812+0000 allocated 0 extents [])
...
2020-11-24T15:50:04.420+0000 7f60bf7e4f40 10 bluestore(/var/lib/ceph/osd/ceph-0) collect_metadata
2020-11-24T15:50:04.420+0000 7f60bf7e4f40 10 bluestore(/var/lib/ceph/osd/ceph-0) get_numa_node bdev ram0 can't detect numa_node
2020-11-24T15:50:04.420+0000 7f60bf7e4f40  0 osd.0 165 done with init, starting boot process
2020-11-24T15:50:04.420+0000 7f60bf7e4f40  1 osd.0 165 start_boot
^^^^^^^^^^^^^^^^
...
Thread 7faf0eb7a700 is proceeding
2020-11-24T15:50:05.828+0000 7faf0eb7a700 10 bluefs _flush 0x5562cd35f420 ignoring, length 25780 < min_flush_size 524288
2020-11-24T15:50:05.828+0000 7faf0eb7a700 10 bluefs get_usage bdev 1 free 1068826624 (1019 MiB) / 1073741824 (1 GiB), used 0%
2020-11-24T15:50:05.828+0000 7faf0eb7a700 10 bluestore(/var/lib/ceph/osd/ceph-0) _get_bluefs_size_delta bluefs 1019 MiB free (0.995422)
bluestore 5.0 GiB free (0.829905), bluefs_ratio 0.166602
2020-11-24T15:50:05.828+0000 7faf0eb7a700 10 bluestore(/var/lib/ceph/osd/ceph-0) _get_bluefs_size_delta bluefs_free 1068826624 < min
1073741824, should gift 4.7 MiB
2020-11-24T15:50:05.828+0000 7faf0eb7a700 10 bluefs _flush 0x5562cd35f420 ignoring, length 25803 < min_flush_size 524288
2020-11-24T15:50:05.828+0000 7faf0eb7a700 10 bluefs _flush 0x5562cd35f420 ignoring, length 25803 < min_flush_size 524288
2020-11-24T15:50:05.828+0000 7faf0eb7a700 10 bluefs _fsync 0x5562cd35f420 file(ino 415 size 0x1835c mtime
2020-11-24T15:49:39.023514+0000 allocated 20000 extents [1:0xa04a0000~20000])
2020-11-24T15:50:05.828+0000 7faf0eb7a700 10 bluefs _flush 0x5562cd35f420 0x1835c~64cb to file(ino 415 size 0x1835c mtime
2020-11-24T15:49:39.023514+0000 allocated 20000 extents [1:0xa04a0000~20000])
....
then back to 7f60a9ad2700
2020-11-24T15:50:05.828+0000 7f60a9ad2700 10 bluefs _flush 0x55c8422695e0 ignoring, length 14199 < min_flush_size 524288
2020-11-24T15:50:05.828+0000 7f60a9ad2700 10 bluefs _flush 0x55c8422695e0 ignoring, length 17245 < min_flush_size 524288
and again 7faf0eb7a700
2020-11-24T15:50:05.828+0000 7faf0eb7a700 10 bluefs _fsync 0x5562cd35f420 file(ino 415 size 0x1e827 mtime
2020-11-24T15:50:05.829650+0000 allocated 20000 extents [1:0xa04a0000~20000])
```

#20 - 11/25/2020 12:02 PM - Igor Fedotov

Hence presumable we have multiple ceph-osd instances using the same bluefs.
I can see at least two issues here. Both are likely to be caused by the containerized environment:
1) multiple instances are started. OSD startup mechanics in containers to be verified.
2) BlueStore's protection mechanism to control mutually exclusive OSD data usage works improperly in this setup.

#21 - 11/25/2020 12:04 PM - Igor Fedotov

You can double check the above by trying to run multiple OSD-0 instance in parallel manually. Highly likely they will start successfully which is invalid.

#22 - 11/25/2020 12:11 PM - Igor Fedotov

May be multiple containers attached to the same volume by some chance?

#23 - 11/26/2020 06:24 AM - Satoru Takeuchi

> You can double check the above by trying to run multiple OSD-0 instance in parallel manually. Highly likely they will start successfully which is invalid.

Succeeded to reproduce this problem by running two OSD containers. I'll continue to investigate this issue with other Rook developers.

#24 - 11/26/2020 06:32 PM - Igor Fedotov

To troubleshoot 2) one might try the following:
- Create two containers that access a single shared folder from a host which is mapped in the containers to /mnt
E.g. in docker one can use "-v /mnt:/mnt" option for "docker run" command to do that.
- Put 'test' file to the host's folder
- Build the following code in both containers with gcc. This simulates Ceph's volume locking mechanics:

```
#include <stdio.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/file.h>
int main()
{
    const char *path = "/mnt/test";
    int fd = ::open(path, O_RDWR | O_DIRECT);
    int r = ::flock(fd, LOCK_EX | LOCK_NB);
    if (r < 0)
      printf(" flock failed on %s\n", path);
    else
      printf (" flock succeeded \n");
    int i;
    scanf("%d", &i);
    ::close(fd);
    return 0;
}
```

- Run resulting binary from both the containers.
The first one will take a lock and indicate a success with "flock succeeded". The second one will report "flock failed" while the first container keeps the lock.

If that's not the case then locking mechanism is broken in your environment which isn't true in my case (OpenSUSE Linux) though.

#25 - 11/29/2020 11:48 PM - Satoru Takeuchi

Thank you for your input!

> I'll continue to investigate this issue with other Rook developers.

I found there is a race that causes this problem in Rook. Now we're thinking how to fix it.

#26 - 12/04/2020 02:03 PM - Igor Fedotov

- Status changed from New to Triaged

- Priority changed from Normal to Low

#27 - 12/04/2020 02:04 PM - Igor Fedotov

- Status changed from Triaged to Need More Info

#28 - 12/05/2020 12:11 AM - Satoru Takeuchi

@Igor

This problem can be fixed by providing an option to move fsid file to other place.
Then Rook, and possibly other containerized Ceph like cephadm, can lock OSD properly
by sharing a directory containing fsid file. This way can keep shared file between
containers as small as possible comparing with sharing OSD's directory.

What do you think about this fix? In addition, I'd like to ask for a hotfix
because this bluefs corruption has happened in my real system and can also happen
in other containerized Ceph.

#29 - 12/14/2020 12:40 PM - Igor Fedotov

Satoru Takeuchi wrote:

> @Igor
>
> This problem can be fixed by providing an option to move fsid file to other place.
> Then Rook, and possibly other containerized Ceph like cephadm, can lock OSD properly
> by sharing a directory containing fsid file. This way can keep shared file between
> containers as small as possible comparing with sharing OSD's directory.
>
> What do you think about this fix?

Hi Satoru,
it seems to me that this is OSD main device (or corresponding symlink in OSD fulder) not fsid file which is locked to control exclusive access to OSD data.

And I think this ticket isn't the proper place to discuss this sort of fixes. And I'm not the proper person to make such decisions as well.
Hence suggest to bring the topic up at ceph-dev mailing list and/or CDM monthly meeting..

> In addition, I'd like to ask for a hotfix
> because this bluefs corruption has happened in my real system and can also happen
> in other containerized Ceph.

It's still unclear to me why multiple OSD instances are able to bypass exclusive lock protection on the main device symlink. Do you have any insight?
Have you tried the code snipped from comment #24? How does it work when running simultaneously from multiple containers? May be your container environment doesn't provide this sort of locking or something?

#30 - 12/16/2020 06:41 AM - Satoru Takeuchi

> it seems to me that this is OSD main device (or corresponding symlink in OSD fulder) not fsid file which is locked to control exclusive access to OSD data.

I confirmed you're correct. Actually, ceph-osd locks both fsid file and block device file.

> And I think this ticket isn't the proper place to discuss this sort of fixes. And I'm not the proper person to make such decisions as well.
> Hence suggest to bring the topic up at ceph-dev mailing list and/or CDM monthly meeting.

Thank you for your information, I'll ask in ceph-dev ML.

> It's still unclear to me why multiple OSD instances are able to bypass exclusive lock protection on the main device symlink.
> Do you have any insight?

It's because each OSD container of Rook doesn't share the OSD directory (/var/lib/ceph/osd/ceph-X/).
Instead, each container creates its own OSD directory in RAM and creates the device file under this
directory. Although these device files have the same major number and minor number, these are regarded
as different files. So the file locking mechanism doesn't work.

> Have you tried the code snipped from comment #24? How does it work when running simultaneously from multiple containers?

Yes, I've tried. It worked well. Sorry, I didn't tell you the result.

> May be your container environment doesn't provide this sort of locking or something?

Since OSD container in Rook doesn't share the OSD directory, both fsid file locking
and block device file locking doesn't work.

The simplest way is to modify Rook to share one persistent OSD directory among all OSD
containers. However, I'd like to keep shared data between each container as small as possible.
So it's better for me to get the place of lock files movable and share only these files.

Is it an acceptable way? Or is it better to make a draft PR first?

#31 - 12/16/2020 02:55 PM - Igor Fedotov

Satoru Takeuchi wrote:

> > It's still unclear to me why multiple OSD instances are able to bypass exclusive lock protection on the main device symlink.
> > Do you have any insight?
>
> It's because each OSD container of Rook doesn't share the OSD directory (/var/lib/ceph/osd/ceph-X/).
> Instead, each container creates its own OSD directory in RAM and creates the device file under this
> directory. Although these device files have the same major number and minor number, these are regarded
> as different files. So the file locking mechanism doesn't work.

I see, thanks for the information.

On the other hand log directory is shared among containers as we could see output from multiple containers in a single file, isn't it?

> > May be your container environment doesn't provide this sort of locking or something?
>
> Since OSD container in Rook doesn't share the OSD directory, both fsid file locking
> and block device file locking doesn't work.
>
> The simplest way is to modify Rook to share one persistent OSD directory among all OSD
> containers. However, I'd like to keep shared data between each container as small as possible.
> So it's better for me to get the place of lock files movable and share only these files.
>
> Is it an acceptable way? Or is it better to make a draft PR first?

Certainly this is technically a doable way but IMO there are many other aspects which IMO makes it not that simple and attractive.
E .g. this will rather complicate upgrade process or some other tools (deployment or testing or something) might depend on this etc.
So I'd like to hear from other developers...

#32 - 12/17/2020 01:37 AM - Satoru Takeuchi

Igor Fedotov wrote:

> On the other hand log directory is shared among containers as we could see output from multiple containers in a single file, isn't it?

It's not shared. More precisely, log is written to stdout instead of logfile in Rook. Then Kuberentes can get each containers's log from a command.

> > Is it an acceptable way? Or is it better to make a draft PR first?
>
> Certainly this is technically a doable way but IMO there are many other aspects which IMO makes it not that simple and attractive.
> E .g. this will rather complicate upgrade process or some other tools (deployment or testing or something) might depend on this etc.
> So I'd like to hear from other developers...

Indeed, my idea seems to be far complicated than I expected. I'll discuss with Rook developers again.

#33 - 01/20/2021 01:32 AM - Satoru Takeuchi

This issue is fixed in Rook.

https://github.com/rook/rook/pull/6793

#34 - 01/20/2021 08:12 AM - Igor Fedotov

- Status changed from Need More Info to Closed

#35 - 01/21/2021 10:23 PM - Sage Weil

I've tried to reproduce this problem of multiple ceph-osds sharing the same device (with cephadm) and the second ceph-osd is correctly failing to start due to the exclusive lock on the block device. I'm not sure how this could happen with rook.

Files

osd-9-bluefs.txt	6.86 KB	10/29/2020	Satoru Takeuchi
bluefs-dump.zip	81.1 KB	10/29/2020	Satoru Takeuchi
ceph-0.log	156 KB	11/06/2020	Satoru Takeuchi
osd_tail.log.gz	179 KB	11/09/2020	Satoru Takeuchi
bluefs-log-dump.log.gz	193 KB	11/09/2020	Satoru Takeuchi