CephFS - Bug #47563

qa: kernel client closes session improperly causing eviction due to timeout

09/21/2020 07:47 PM - Patrick Donnelly

Status:	Resolved	**% Done:**	0%
Priority:	High		
Assignee:	Jeff Layton		
Category:			
Target version:	v16.0.0		
Source:	Q/A	**Affected Versions:**	
Tags:		**ceph-qa-suite:**	
Backport:	octopus,nautilus	**Component(FS):**	MDS, kceph
Regression:	No	**Labels (FS):**	qa, qa-failure
Severity:	3 - minor	**Pull request ID:**	37619
Reviewed:		**Crash signature:**	

Description

```
2020-09-20T13:43:05.420 INFO:tasks.workunit:Stopping ['suites/pjd.sh'] on client.0...
2020-09-20T13:43:06.310 DEBUG:teuthology.parallel:result is None
2020-09-20T13:43:06.435 INFO:tasks.workunit:Deleted dir /home/ubuntu/cephtest/mnt.0/client.0
2020-09-20T13:43:06.435 DEBUG:teuthology.run_tasks:Unwinding manager kclient
2020-09-20T13:43:06.447 INFO:tasks.kclient:Unmounting kernel clients...
2020-09-20T13:43:06.447 DEBUG:tasks.cephfs.kernel_mount:Unmounting client client.0...
2020-09-20T13:43:06.448 INFO:teuthology.orchestra.run:Running command with timeout 900
2020-09-20T13:43:06.448 INFO:teuthology.orchestra.run.smithi093:> sudo umount /home/ubuntu/cephtes
t/mnt.0
2020-09-20T13:43:51.775 INFO:teuthology.orchestra.run.smithi084:> sudo adjust-ulimits ceph-coverag
e /home/ubuntu/cephtest/archive/coverage timeout 120 ceph --cluster ceph --log-early fs dump --for
mat=json
```

From:
/ceph/teuthology-archive/pdonnell-2020-09-20_07:13:47-multimds-wip-pdonnell-testing-20200920.040823-distro-basic-smithi/5451505/teuthology.log

(There is also thrashing going on but that's after the kernel unmount and not related.)

Then on the MDS:

```
2020-09-20T13:43:06.582+0000 7f69fb0ed700  7 mds.0.locker issue_caps loner client.4713 allowed=pAs
LsXsFsxcrwb, xlocker allowed=pAsLsXsFsxcrwb, others allowed=pAsLsXs on [inode 0x10000000511 [2,hea
d] ~mds0/stray1/10000000511 auth v2088 pv2302 ap=1 dirtyparent s=1 nl=0 n(v0 rc2020-09-20T13:42:39
.280144+0000 b1 1=1+0)->n(v0 rc2020-09-20T13:42:39.280144+0000 b1 1=1+0) (ifile excl w=1) (iversio
n lock) cr={4713=0-4194304@1} caps={4713=pFc/-@6},l=4713(-1) | ptrwaiter=1 request=0 lock=1 caps=1
 truncating=0 dirtyparent=1 dirty=1 waiter=0 authpin=1 0x55b1c67e5080]
2020-09-20T13:43:06.582+0000 7f69fb0ed700 20 mds.0.locker  client.4713 pending pFc allowed pAsLsXs
Fsxcrwb wanted -
2020-09-20T13:43:06.582+0000 7f69fb0ed700  7 mds.0.locker    sending MClientCaps to client.4713 se
q 7 new pending pAsLsXsFsxcrwb was pFc
2020-09-20T13:43:06.582+0000 7f69fb0ed700 20 mds.0.cache.ino(0x10000000511) encode_cap_message pfi
le 1 pauth 0 plink 0 pxattr 0 ctime 2020-09-20T13:42:39.280144+0000
2020-09-20T13:43:06.582+0000 7f69fb0ed700 10 mds.0.18 send_message_client_counted client.4713 seq 
1926 client_caps(grant ino 0x10000000511 1 seq 7 caps=pAsLsXsFsxcrwb dirty=- wanted=Ls follows 0 s
ize 1/0 ts 2/1 mtime 2020-09-20T13:42:39.237145+0000 tws 1) v11
2020-09-20T13:43:06.582+0000 7f69fb0ed700  1 -- [v2:172.21.15.84:6825/1405287455,v1:172.21.15.84:6
827/1405287455] --> v1:172.21.15.93:0/4080061763 -- client_caps(grant ino 0x10000000511 1 seq 7 ca
ps=pAsLsXsFsxcrwb dirty=- wanted=Ls follows 0 size 1/0 ts 2/1 mtime 2020-09-20T13:42:39.237145+000
0 tws 1) v11 -- 0x55b1c6718900 con 0x55b1c5084000
2020-09-20T13:43:06.582+0000 7f69fb0ed700 10 mds.0.locker eval done
2020-09-20T13:43:06.582+0000 7f69fb0ed700 10 -- [v2:172.21.15.84:6825/1405287455,v1:172.21.15.84:6
827/1405287455] dispatch_throttle_release 236 to dispatch throttler 264/104857600
2020-09-20T13:43:06.582+0000 7f69fb0ed700  1 -- [v2:172.21.15.84:6825/1405287455,v1:172.21.15.84:6
827/1405287455] <== client.4713 v1:172.21.15.93:0/4080061763 2777 ==== client_session(request_clos
e seq 1925) ==== 28+0+0 (unknown 1265461456 0 0) 0x55b1c6dd4e00 con 0x55b1c5084000
2020-09-20T13:43:06.582+0000 7f69fd8f2700 10 -- [v2:172.21.15.84:6825/1405287455,v1:172.21.15.84:6
827/1405287455] >> v1:172.21.15.93:0/4080061763 conn(0x55b1c5084000 legacy=0x55b1c5c1a000 unknown
:6827 s=STATE_CONNECTION_ESTABLISHED l=0).handle_write
2020-09-20T13:43:06.582+0000 7f69fd8f2700 10 --1- [v2:172.21.15.84:6825/1405287455,v1:172.21.15.84
:6827/1405287455] >> v1:172.21.15.93:0/4080061763 conn(0x55b1c5084000 0x55b1c5c1a000 :6827 s=OPENE
D pgs=13 cs=1 l=0).write_event
2020-09-20T13:43:06.582+0000 7f69fb0ed700 20 mds.0.18 get_session have 0x55b1c4305600 client.4713
v1:172.21.15.93:0/4080061763 state open
2020-09-20T13:43:06.582+0000 7f69fb0ed700  3 mds.0.server handle_client_session client_session(req
uest_close seq 1925) from client.4713
2020-09-20T13:43:06.582+0000 7f69fb0ed700 10 mds.0.server old push seq 1925 < 1926, dropping
```

The MDS drops the close request. (This could arguably be a bug in the MDS too!) It appears the connection was never shut down so the kclient is waiting for the MDS to respond. The thrasher caused a failover which appears to have allowed the kernel to finish unmounting:

```
2020-09-20T13:44:06.839 INFO:tasks.mds_thrash.fs.[cephfs]:waiting till mds map indicates mds.c is
laggy/crashed, in failed state, or mds.c is removed from mdsmap
2020-09-20T13:44:07.235 INFO:teuthology.orchestra.run:Running command with timeout 300
2020-09-20T13:44:07.235 INFO:teuthology.orchestra.run.smithi093:> (cd /home/ubuntu/cephtest && exe
c rmdir -- /home/ubuntu/cephtest/mnt.0)
```

Related issues:

Related to CephFS - Bug #48439: fsstress failure with mds thrashing: "mds.0.6... **New**

History

#1 - 09/28/2020 01:44 PM - Patrick Donnelly

- Assignee set to Jeff Layton

#2 - 09/28/2020 04:34 PM - Jeff Layton

Hmm, ok. This may be related to another bug I've been chasing where umount hangs waiting for the session to close. I wonder if this is the same issue...

The kclient never appears to retransmit a CEPH_SESSION_REQUEST_CLOSE request on a session. One and done:

```
static int __close_session(struct ceph_mds_client *mdsc,
                         struct ceph_mds_session *session)
{
        if (session->s_state >= CEPH_MDS_SESSION_CLOSING)
                return 0;
        session->s_state = CEPH_MDS_SESSION_CLOSING;
        return request_close_session(session);
}
```

I'm not that familiar yet with the seq number handling in the client. Is there some documentation on how this is intended to work? Under what circumstances should the client retransmit a request that might have been dropped? AFAICT, the client doesn't ever drop an MDS request due to s_seq being too old. Should it be doing that?

If the client sent the request with too old a s_seq value, then probably the cap request came in after that was sent, and we never resend the request. I guess I should look at libcephfs and see if it does this more sanely...

#3 - 09/28/2020 04:39 PM - Jeff Layton

- Status changed from New to In Progress

#4 - 09/28/2020 04:51 PM - Patrick Donnelly

Jeff Layton wrote:

> Hmm, ok. This may be related to another bug I've been chasing where umount hangs waiting for the session to close. I wonder if this is the same issue...
>
> The kclient never appears to retransmit a CEPH_SESSION_REQUEST_CLOSE request on a session. One and done:
>
> [...]
>
> I'm not that familiar yet with the seq number handling in the client. Is there some documentation on how this is intended to work?

Not to my knowledge. This code path has been mostly unchanged since Ceph started:
https://github.com/ceph/ceph/blob/c1865445c5b88cae2df499e4d2047af156225177/src/mds/Server.cc#L163-L168

> Under what circumstances should the client retransmit a request that might have been dropped? AFAICT, the client doesn't ever drop an MDS request due to s_seq being too old. Should it be doing that?

I don't think so. Only the MDS increments the sequence number.It would be a bug in the MDS if there were an older seq number I would think. Maybe the kernel client should detect that and issue warnings?

> If the client sent the request with too old a s_seq value, then probably the cap request came in after that was sent, and we never resend the request. I guess I should look at libcephfs and see if it does this more sanely...

Yes, that'd be a good idea. It may be a bug in both clients. It's weird we're only seeing this for the first time in the last few months.

FWIW, the push sequence is one way the MDS apparently figures out where the client is in processing the MDS messages. We only use that in a few places in the mds (see `git grep -E get_push_seq\|get_seq`). Mostly for cap handling.

#5 - 09/28/2020 05:07 PM - Jeff Layton

Doesn't look like libcephfs does anything saner:

```
  while (!mds_sessions.empty()) {
    // send session closes!
    for (auto &p : mds_sessions) {
      if (p.second.state != MetaSession::STATE_CLOSING) {
        _close_mds_session(&p.second);
        mds_ranks_closing.insert(p.first);
      }
    }
```

```
    // wait for sessions to close
    double timo = cct->_conf.get_val<std::chrono::seconds>("client_shutdown_timeout").count();
    ldout(cct, 2) << "waiting for " << mds_ranks_closing.size() << " mds session(s) to close (timeout: "
                  << timo << "s)" << dendl;
    std::unique_lock l{client_lock, std::adopt_lock};
```

...and it looks like the timeout defaults to 2m there vs. 60s in kernel client. Basically, it's just sending the REQUEST_CLOSE once and then giving up after the wait times out.

A simple MDS-side fix might be to just have the MDS ignore the seq on a REQUEST_CLOSE. The client's not going to decide to keep the session after that point anyway.

On the client side, we could just retransmit the request several times until we get a reply, with a small delay between transmissions. It's not pretty but it would probably work.

#6 - 09/28/2020 05:39 PM - Patrick Donnelly

Jeff Layton wrote:

> Doesn't look like libcephfs does anything saner:
>
> [...]
>
> ...and it looks like the timeout defaults to 2m there vs. 60s in kernel client. Basically, it's just sending the REQUEST_CLOSE once and then giving up after the wait times out.
>
> A simple MDS-side fix might be to just have the MDS ignore the seq on a REQUEST_CLOSE. The client's not going to decide to keep the session after that point anyway.

I *think* the concern is that hte client could conceivably dirty the cap the MDS just issued. I don't really think that makes sense though.

> On the client side, we could just retransmit the request several times until we get a reply, with a small delay between transmissions. It's not pretty but it would probably work.

Perhaps the MDS is the right thing to fix but I'm not sure on the implications. We'll need Zheng's input on this.

#7 - 09/28/2020 05:50 PM - Jeff Layton

Patrick Donnelly wrote:

> I *think* the concern is that hte client could conceivably dirty the cap the MDS just issued. I don't really think that makes sense though.

Where we send a REQUEST_CLOSE is pretty close to final. In most cases, we're tearing the superblock down at that point. Nothing should be getting dirtied after that.

I'll see if I can draft something up to do some retries.

#8 - 09/28/2020 07:29 PM - Jeff Layton

I have a patch I'm testing now that seems to also anecdotally fix some of the umount hangs I've seen lately during xfstests runs. I wonder if we ought to combine that with some patches to make the client better vet incoming messages vs. the s_seq. I worry a bit though that the rules on this are vague.

Another idea might be to have the client ignore most messages when the state is >= CEPH_MDS_SESSION_CLOSING. We would still need to bump the s_seq for those messages too. FWIW, it doesn't seem like the session sequence is bumped for every message. On what sorts of messages is it incremented?

#9 - 10/09/2020 06:51 PM - Patrick Donnelly

- Status changed from In Progress to Fix Under Review

- Backport set to octopus,nautilus

- Pull request ID set to 37619

#10 - 10/09/2020 06:52 PM - Patrick Donnelly

- Labels (FS) qa, qa-failure added

#11 - 11/11/2020 02:46 AM - Patrick Donnelly

Jeff, this is just waiting on kcephfs patches now right?

#12 - 11/11/2020 12:41 PM - Jeff Layton

- Status changed from Fix Under Review to Resolved

Patrick Donnelly wrote:

> Jeff, this is just waiting on kcephfs patches now right?

Yes. The patch was just merged into mainline late last week in commit 62575e270f661aba64778cbc5f354511cf9abb21.

#13 - 12/02/2020 08:12 PM - Patrick Donnelly

- Related to Bug #48439: fsstress failure with mds thrashing: "mds.0.6 Evicting (and blocklisting) client session 4564 (v1:172.21.15.47:0/603539598)" added