fs - Bug #16914

multimds: pathologically slow deletions in some tests

08/03/2016 09:08 PM - Patrick Donnelly

Status:	Resolved	**Start date:**	08/03/2016
Priority:	High	**Due date:**	
Assignee:	John Spray	**% Done:**	0%
Category:		**Estimated time:**	0.00 hour
Target version:	v12.0.0		
Source:	Development	**Affected Versions:**	
Tags:		**ceph-qa-suite:**	
Backport:		**Component(FS):**	
Regression:	No	**Labels (FS):**	multimds
Severity:	3 - minor	**Pull request ID:**	
Reviewed:		**Crash signature:**	

Description

http://qa-proxy.ceph.com/teuthology/pdonnell-2016-07-18_20:02:54-multimds-master---basic-mira/321823/teuthology.log

```
Timeout 3h running suites/dbench.sh
                                                                                                   1 job
s: ['321823']
                                                                                              suites: ['
clusters/9-mds.yaml', 'debug/mds_client.yaml', 'fs/btrfs.yaml', 'inline/yes.yaml', 'mount/kclient.
yaml', 'multimds/basic/{ceph/base.yaml', 'overrides/whitelist_wrongly_marked_down.yaml', 'tasks/su
ites_dbench.yaml}']
```

http://qa-proxy.ceph.com/teuthology/pdonnell-2016-07-21_13:20:27-multimds-master---basic-mira/327465/teuthology.log

```
Timeout 3h running suites/dbench.sh
1 jobs: ['327465']
suites: ['clusters/3-mds.yaml', 'debug/mds_client.yaml', 'fs/btrfs.yaml', 'inline/no.yaml', 'mount
/kclient.yaml', 'multimds/basic/{ceph/base.yaml', 'overrides/whitelist_wrongly_marked_down.yaml',
'tasks/suites_dbench.yaml}']
```

History

#1 - 08/08/2016 01:49 PM - John Spray

retest with fuse default permissions set differently because it's doing too many getattr at the moment

#2 - 08/08/2016 01:49 PM - John Spray

- Assignee set to Patrick Donnelly

#3 - 09/13/2016 11:42 PM - Patrick Donnelly

Looking at this new run which toggles fuse_default_permissions:

http://pulpito.ceph.com/pdonnell-2016-08-11_20:40:52-multimds-master-testing-basic-mira/

Apparently the getattr calls were indeed the cause. Most apparently would take up to 5 seconds to complete. Here is one such run:

http://pulpito.ceph.com/pdonnell-2016-08-11_20:40:52-multimds-master-testing-basic-mira/358878

This has 3555 getattr requests:

```
$ zcat remote/mira075/log/ceph-client.0.27249.log.gz | grep 'send_request.*getattr' | wc -l
3555
```

The run is killed during the `make clean` of building the kernel.

Here is an excerpt of one of the final calls to unlink by the client:

```
2016-08-13 20:05:40.166346 7f73eb92a700  3 client.4162 ll_unlink 90000003ca7.head earlycpio.c
2016-08-13 20:05:40.166353 7f73eb92a700  3 client.4162 _unlink(90000003ca7 earlycpio.c uid 1000 gid 1171)
2016-08-13 20:05:40.166364 7f73eb92a700 10 client.4162 _lookup 90000003ca7.head(faked_ino=0 ref=4 ll_ref=3030
cap_refs={} open={} mode=40775 size=0/0 mtime=2016-08-13 20:05:35.164187 caps=pAsLsXsFs(6=pAsLsXsFs,8=pAsLsXs)
 parents=0x148998e0 0x1a520d00) earlycpio.c = 90000003ce3.head(faked_ino=0 ref=3 ll_ref=5 cap_refs={1024=0,204
8=0,4096=0,8192=0} open={1=0,2=0} mode=100664 size=4028/0 mtime=2015-06-06 15:21:22.000000 caps=pAsLsXsFscr(6=
pAsLsXsFscr) objectset[90000003ce3 ts 0/0 objects 1 dirty_or_tx 0] parents=0xefcfb20 0x1a5aad00)
2016-08-13 20:05:40.166393 7f73eb92a700 10 client.4162 choose_target_mds from caps on inode 90000003ca7.head(f
aked_ino=0 ref=5 ll_ref=3030 cap_refs={} open={} mode=40775 size=0/0 mtime=2016-08-13 20:05:35.164187 caps=pAs
LsXsFs(6=pAsLsXsFs,8=pAsLsXs) parents=0x148998e0 0x1a520d00)
2016-08-13 20:05:40.166408 7f73eb92a700 10 client.4162 send_request rebuilding request 1361294 for mds.8
2016-08-13 20:05:40.166413 7f73eb92a700 10 client.4162 send_request client_request(unknown.0:1361294 unlink #9
0000003ca7/earlycpio.c 2016-08-13 20:05:40.166392) v3 to mds.8
2016-08-13 20:05:40.166419 7f73eb92a700  1 -- 172.21.7.136:0/283400045 --> 172.21.7.132:6816/29464 -- client_r
equest(client.4162:1361294 unlink #90000003ca7/earlycpio.c 2016-08-13 20:05:40.166392) v3 -- ?+0 0x23824580 co
n 0x10262600
2016-08-13 20:05:40.167626 7f73f143f700  1 -- 172.21.7.136:0/283400045 <== mds.8 172.21.7.132:6816/29464 17082
0 ==== client_request_forward(1361294 to mds.6 num_fwd=1 client_must_resend) v1 ==== 9+0+0 (4179948797 0 0) 0x
efb7fe0 con 0x10262600
2016-08-13 20:05:40.167664 7f73f143f700 10 client.4162 handle_client_request tid 1361294 fwd 1 to mds.6, resen
ding to 6
2016-08-13 20:05:40.167770 7f73eb92a700 10 client.4162 choose_target_mds resend_mds specified as mds.6
2016-08-13 20:05:40.167781 7f73eb92a700 10 client.4162 send_request rebuilding request 1361294 for mds.6
2016-08-13 20:05:40.167796 7f73eb92a700 10 client.4162 send_request client_request(unknown.0:1361294 unlink #9
0000003ca7/earlycpio.c 2016-08-13 20:05:40.166392) v3 to mds.6
2016-08-13 20:05:40.167806 7f73eb92a700  1 -- 172.21.7.136:0/283400045 --> 172.21.7.132:6814/29449 -- client_r
equest(client.4162:1361294 unlink #90000003ca7/earlycpio.c 2016-08-13 20:05:40.166392) v3 -- ?+0 0x23827180 co
n 0x96a8600
2016-08-13 20:05:40.173440 7f73f143f700  1 -- 172.21.7.136:0/283400045 <== mds.6 172.21.7.132:6814/29449 19176
0 ==== client_reply(???:1361294 = 0 (0) Success unsafe) v1 ==== 387+0+0 (1284485451 0 0) 0x23827180 con 0x96a8
600
2016-08-13 20:05:40.173487 7f73f143f700 10 client.4162 insert_trace from 2016-08-13 20:05:40.167796 mds.6 is_t
arget=0 is_dentry=1
2016-08-13 20:05:40.173496 7f73f143f700 10 client.4162  features 0x7fffffefdfffff
2016-08-13 20:05:40.173499 7f73f143f700 10 client.4162 update_snap_trace len 48
2016-08-13 20:05:40.173502 7f73f143f700 10 client.4162 update_snap_trace snaprealm(1 nref=13351 c=0 seq=1 pare
nt=0 my_snaps=[] cached_snapc=1=[]) seq 1 <= 1 and same parent, SKIPPING
2016-08-13 20:05:40.173507 7f73f143f700 10 client.4162  hrm  is_target=0 is_dentry=1
2016-08-13 20:05:40.173516 7f73f143f700 10 client.4162 add_update_cap issued pAsLsXs -> pAsLsXs from mds.6 on
90000003ca7.head(faked_ino=0 ref=5 ll_ref=3030 cap_refs={} open={} mode=40775 size=0/0 mtime=2016-08-13 20:05:
35.164187 caps=pAsLsXs(6=pAsLsXs,8=pAsLsXs) parents=0x148998e0 0x1a520d00)
2016-08-13 20:05:40.173538 7f73f143f700 10 client.4162 put_inode on 90000003ce3.head(faked_ino=0 ref=4 ll_ref=
5 cap_refs={1024=0,2048=0,4096=0,8192=0} open={1=0,2=0} mode=100664 size=4028/0 mtime=2015-06-06 15:21:22.0000
00 caps=pAsXsFscr(6=pAsXsFscr) objectset[90000003ce3 ts 0/0 objects 1 dirty_or_tx 0] 0x1a5aad00)
2016-08-13 20:05:40.173670 7f73eb92a700  3 client.4162 unlink(#90000003ca7/earlycpio.c) = 0
2016-08-13 20:05:40.173679 7f73eb92a700 10 client.4162 put_inode on 90000003ce3.head(faked_ino=0 ref=3 ll_ref=
5 cap_refs={1024=0,2048=0,4096=0,8192=0} open={1=0,2=0} mode=100664 size=4028/0 mtime=2015-06-06 15:21:22.0000
00 caps=pAsXsFscr(6=pAsXsFscr) objectset[90000003ce3 ts 0/0 objects 1 dirty_or_tx 0] 0x1a5aad00)
```

Here is a call to getattr shortly after which has a 5 second turnaround time:

```
2016-08-13 20:05:40.173945 7f73e2d07700  3 client.4162 ll_getattr 90000003ca7.head
2016-08-13 20:05:40.173956 7f73e2d07700 10 client.4162 _getattr mask pAsLsXsFs issued=0
2016-08-13 20:05:40.173969 7f73e2d07700 10 client.4162 choose_target_mds from caps on inode 90000003ca7.head(f
aked_ino=0 ref=6 ll_ref=3030 cap_refs={} open={} mode=40775 size=0/0 mtime=2016-08-13 20:05:35.164187 caps=pAs
LsXs(6=pAsLsXs,8=pAsLsXs) parents=0x148998e0 0x1a520d00)
2016-08-13 20:05:40.173988 7f73e2d07700 10 client.4162 send_request rebuilding request 1361295 for mds.6
2016-08-13 20:05:40.173996 7f73e2d07700 10 client.4162 send_request client_request(unknown.0:1361295 getattr p
AsLsXsFs #90000003ca7 2016-08-13 20:05:40.173967) v3 to mds.6
2016-08-13 20:05:40.174004 7f73e2d07700  1 -- 172.21.7.136:0/283400045 --> 172.21.7.132:6814/29449 -- client_r
equest(client.4162:1361295 getattr pAsLsXsFs #90000003ca7 2016-08-13 20:05:40.173967) v3 -- ?+0 0x10dc7600 con
 0x96a8600
```

```
2016-08-13 20:05:40.923668 7f73f2441700  1 -- 172.21.7.136:0/283400045 --> 172.21.7.132:6814/29449 -- client_c
ap_release(1) v2 -- ?+0 0x10492760 con 0x96a8600
2016-08-13 20:05:45.153865 7f73f143f700  1 -- 172.21.7.136:0/283400045 <== mds.6 172.21.7.132:6814/29449 19176
1 ==== client_caps(grant ino 90000003ce3 3574 seq 10 caps=pAsLsXsFscr dirty=- wanted=Ls follows 0 mseq 1 size
4028/0 ts 1/18446744073709551615 mtime 2015-06-06 15:21:22.000000 tws 1) v8 ==== 216+0+0 (460624340 0 0) 0x19b
b4400 con 0x96a8600
2016-08-13 20:05:45.153921 7f73f143f700 10 client.4162  mds.6 seq now 69093
2016-08-13 20:05:45.153930 7f73f143f700  5 client.4162 handle_cap_grant on in 90000003ce3 mds.6 seq 10 caps no
w pAsLsXsFscr was pAsXsFscr
2016-08-13 20:05:45.153939 7f73f143f700 10 client.4162 update_inode_file_bits 90000003ce3.head(faked_ino=0 ref
=1 ll_ref=0 cap_refs={1024=0,2048=0,4096=0,8192=0} open={1=0,2=0} mode=100664 size=4028/0 mtime=2015-06-06 15:
21:22.000000 caps=pAsXsFscr(6=pAsXsFscr) objectset[90000003ce3 ts 0/0 objects 1 dirty_or_tx 0] 0x1a5aad00) pAs
XsFscr mtime 2015-06-06 15:21:22.000000
2016-08-13 20:05:45.153959 7f73f143f700 10 client.4162   grant, new caps are Ls
2016-08-13 20:05:45.153967 7f73f143f700  1 -- 172.21.7.136:0/283400045 <== mds.6 172.21.7.132:6814/29449 19176
2 ==== client_reply(???:1361294 = 0 (0) Success safe) v1 ==== 27+0+0 (2607689463 0 0) 0x10dc7600 con 0x96a8600
2016-08-13 20:05:45.153989 7f73f143f700 10 client.4162 insert_trace from 2016-08-13 20:05:40.167796 mds.6 is_t
arget=0 is_dentry=0
2016-08-13 20:05:45.153994 7f73f143f700 10 client.4162 insert_trace -- already got unsafe; ignoring
2016-08-13 20:05:45.153998 7f73f143f700 10 client.4162 put_inode on 90000003ca7.head(faked_ino=0 ref=6 ll_ref=
3030 cap_refs={} open={} mode=40775 size=0/0 mtime=2016-08-13 20:05:35.164187 caps=pAsLsXs(6=pAsLsXs,8=pAsLsXs
) parents=0x148998e0 0x1a520d00)
2016-08-13 20:05:45.154012 7f73f143f700 10 client.4162 put_inode on 90000003ce3.head(faked_ino=0 ref=1 ll_ref=
0 cap_refs={1024=0,2048=0,4096=0,8192=0} open={1=0,2=0} mode=100664 size=4028/0 mtime=2015-06-06 15:21:22.0000
00 caps=pAsLsXsFscr(6=pAsLsXsFscr) objectset[90000003ce3 ts 0/0 objects 1 dirty_or_tx 0] 0x1a5aad00)
2016-08-13 20:05:45.154027 7f73f143f700 10 client.4162 remove_cap mds.6 on 90000003ce3.head(faked_ino=0 ref=0
ll_ref=0 cap_refs={1024=0,2048=0,4096=0,8192=0} open={1=0,2=0} mode=100664 size=4028/0 mtime=2015-06-06 15:21:
22.000000 caps=pAsLsXsFscr(6=pAsLsXsFscr) objectset[90000003ce3 ts 0/0 objects 1 dirty_or_tx 0] 0x1a5aad00)
2016-08-13 20:05:45.154043 7f73f143f700 10 client.4162 put_inode deleting 90000003ce3.head(faked_ino=0 ref=0 l
l_ref=0 cap_refs={1024=0,2048=0,4096=0,8192=0} open={1=0,2=0} mode=100664 size=4028/0 mtime=2015-06-06 15:21:2
2.000000 caps=- objectset[90000003ce3 ts 0/0 objects 1 dirty_or_tx 0] 0x1a5aad00)
2016-08-13 20:05:45.163049 7f73f143f700  1 -- 172.21.7.136:0/283400045 <== mds.6 172.21.7.132:6814/29449 19176
3 ==== client_reply(???:1361295 = 0 (0) Success) v1 ==== 354+0+0 (2291237687 0 0) 0x244eb600 con 0x96a8600
2016-08-13 20:05:45.163096 7f73f143f700 10 client.4162 insert_trace from 2016-08-13 20:05:40.173995 mds.6 is_t
arget=1 is_dentry=0
2016-08-13 20:05:45.163105 7f73f143f700 10 client.4162  features 0x7fffffefdfffff
2016-08-13 20:05:45.163107 7f73f143f700 10 client.4162 update_snap_trace len 48
2016-08-13 20:05:45.163110 7f73f143f700 10 client.4162 update_snap_trace snaprealm(1 nref=13350 c=0 seq=1 pare
nt=0 my_snaps=[] cached_snapc=1=[]) seq 1 <= 1 and same parent, SKIPPING
2016-08-13 20:05:45.163116 7f73f143f700 10 client.4162  hrm  is_target=1 is_dentry=0
2016-08-13 20:05:45.163124 7f73f143f700 10 client.4162 update_inode_file_bits 90000003ca7.head(faked_ino=0 ref
=5 ll_ref=3030 cap_refs={} open={} mode=40775 size=0/0 mtime=2016-08-13 20:05:35.164187 caps=pAsLsXs(6=pAsLsXs
,8=pAsLsXs) parents=0x148998e0 0x1a520d00) pAsLsXs mtime 2016-08-13 20:05:40.166392
2016-08-13 20:05:45.163145 7f73f143f700 10 client.4162 add_update_cap issued pAsLsXs -> pAsLsXsFs from mds.6 o
n 90000003ca7.head(faked_ino=0 ref=5 ll_ref=3030 cap_refs={} open={} mode=40775 size=0/0 mtime=2016-08-13 20:0
5:40.166392 caps=pAsLsXsFs(6=pAsLsXsFs,8=pAsLsXs) parents=0x148998e0 0x1a520d00)
2016-08-13 20:05:45.163273 7f73e2d07700 10 client.4162 _getattr result=0
2016-08-13 20:05:45.163283 7f73e2d07700 10 client.4162 fill_stat on 90000003ca7 snap/devhead mode 040775 mtime
 2016-08-13 20:05:40.166392 ctime 2016-08-13 17:20:38.183148
2016-08-13 20:05:45.163293 7f73e2d07700  3 client.4162 ll_getattr 90000003ca7.head = 0
```

We can compare to this successful run which only toggles the fuse_default_permissions to off:

http://pulpito.ceph.com/pdonnell-2016-08-11_20:40:52-multimds-master-testing-basic-mira/358914

That run has 163 getattr requests:

```
pdonnell@teuthology /ceph/teuthology-archive/pdonnell-2016-08-11_20:40:52-multimds-master-testing-basic-mira/3
58914$ zcat remote/mira*/log/ceph-client.0* | grep 'send_request.*getattr' | wc -l
163
```

So "fuse_default_permissions: false" cuts about 95% of getattr requests for the kernel build/clean test.

The other issue is what's causing the 5 second turnaround for getattr. Greg thinks it is related to a log flush (in another mds) delaying a lock acquisition. I'm looking into this as well.

#4 - 09/14/2016 01:30 AM - Patrick Donnelly

Okay so it appears Greg is right after looking through the mds logs for the client getattr op posted above. It appears from:

/ceph/teuthology-archive/pdonnell-2016-08-11_20:40:52-multimds-master-testing-basic-mira/358878/remote/mira077/log/ceph-mds.i.log.gz

that the MDS was stalled waiting for the log flush for the unlink of earlycpio.c:

```
2016-08-13 20:05:45.154497 7eff24caf700 10 MDSIOContextBase::complete: 13C_MDL_Flushed
2016-08-13 20:05:45.154499 7eff24caf700 10 MDSInternalContextBase::complete: 11C_MarkEvent
2016-08-13 20:05:45.154510 7eff24caf700 10 MDSInternalContextBase::complete: 25C_MDS_unlink_local_finish
2016-08-13 20:05:45.154512 7eff24caf700 10 mds.6.server _unlink_local_finish [dentry #1/client.0/tmp/t/linux-4
.0.5/lib/earlycpio.c [2,head] auth (dn xlockdone l=1 x=1) (dversion lock w=1 last_client=4162) pv=4899 v=680 a
p=2+2 inode=0xc082e20 | request=1 lock=2 inodepin=1 replicated=0 dirty=0 authpin=1 clientlease=1 0xc1c1080]
2016-08-13 20:05:45.154523 7eff24caf700 12 mds.6.cache.dir(90000003ca7) unlink_inode [dentry #1/client.0/tmp/t
/linux-4.0.5/lib/earlycpio.c [2,head] auth (dn xlockdone l=1 x=1) (dversion lock w=1 last_client=4162) pv=4899
 v=680 ap=2+2 inode=0xc082e20 | request=1 lock=2 inodepin=1 replicated=0 dirty=0 authpin=1 clientlease=1 0xc1c
1080] [inode 90000003ce3 [2,head] {/client.0/tmp/t/linux-4.0.5/lib/earlycpio.c ~mds6/stray6/90000003ce3} auth
v680 pv17749 ap=2+0 s=4028 n(v0 b4028 1=1+0)->n(v0 b4028 1=1+0) (ilink xlockdone x=1) (iversion lock w=1 last_
client=4162) caps={4162=pAsXsFscr/-@9} | request=1 lock=2 importingcaps=0 caps=1 dirtyparent=0 replicated=0 di
rty=0 authpin=1 0xc082e20]
```

Shortly afterwards it looks like the necessary locks were obtained and the getattr completes.

There were some interactions I don't understand with another mds.8 by mds.6 (the MDS fulfilling the requests). The client would send the unlink request first to mds.8 and then be forwarded to mds.6. While mds.6 was handling the unlink/getattr, it would (try to) get some locks from mds.8. I think mds.8 has a directory fragment of the same inode so that's probably why.

So in summary, it seems that setting "fuse_default_permissions: false" is a good workaround for this bug. However, there is still an issue with flushes delaying getattrs. Does this sound right?

#5 - 11/09/2016 01:49 PM - John Spray

- Priority changed from Normal to High

- Target version set to v12.0.0

#6 - 12/24/2016 02:45 PM - John Spray

I have a nice simple reproducer for this now (even with fuse default permissions = false it has the slowdown).

It's TestStrays.test_replicated_delete_speed jcsp/wip-11950

#7 - 12/28/2016 10:19 AM - Zheng Yan

The reason is that you use "rm -rf delete_me/*" to delete files. ceph-fuse needs to do a lookup "delete_me" for each files. Using "rm -rf delete/" has no problem

#8 - 01/04/2017 01:57 PM - John Spray

Right, but users are going to do this -- it needs to work.

#9 - 03/03/2017 10:18 AM - John Spray

Pulled the test off wip-11950 and onto https://github.com/ceph/ceph/pull/13770

#10 - 03/08/2017 05:15 PM - John Spray

Notes on discussion today:

- The issue is that we usually (single mds) do fine in these cases because of the projection mechanism in the MDS -- we don't have to wait for something to persist before reading it, if it's the same client.
- In the multi-mds case, MDSs don't know about each other's projected state, so they end up waiting for something to persist. In these cases we end up waiting for a tick().
- There are ways we could introduce something smart to track all this (some kind of inter-MDS projection !!!) but it would not be straightforward.

The hope is that with some more analysis of this we can identify just the right place to put an extra (:-/) explicit log flush, probably somewhere in the Locker.

#11 - 06/20/2017 06:09 PM - John Spray

- Status changed from New to Need Review

- Assignee changed from Patrick Donnelly to John Spray

It looks like this case is now working properly with the latest code, so flipping this ticket to need review and removing the DNM from the pull request.

#12 - 07/07/2017 04:35 AM - Patrick Donnelly

- Status changed from Need Review to Resolved

#13 - 03/09/2019 12:31 AM - Patrick Donnelly

- Category deleted (90)

- Labels (FS) multimds added