CephFS - Bug #4451

client: Ceph client not releasing cap

03/15/2013 08:59 AM - Sam Lang

Status:	Resolved	**% Done:**	0%
Priority:	Urgent		
Assignee:	Sam Lang		
Category:			
Target version:	v0.61 - Cuttlefish		
Source:	Development	**ceph-qa-suite:**	
Tags:		**Component(FS):**	
Backport:		**Labels (FS):**	
Regression:	No	**Pull request ID:**	
Severity:	3 - minor	**Crash signature (v1):**	
Reviewed:		**Crash signature (v2):**	
Affected Versions:			

Description

I'm occasionally hitting a hang in my backtrace testing, where unmount never completes. The client log shows a disconnected inode with a non-zero ref count:

2013-03-15 09:09:18.065054 7fd477466700 2 client.4117 cache still has 0+1 items, waiting (for caps to release?)
2013-03-15 09:09:23.065131 7fd477466700 1 client.4117 dump_cache
2013-03-15 09:09:23.065180 7fd477466700 1 client.4117 dump_inode: DISCONNECTED inode 1000000025f #1000000025f ref 01000000025f.head(ref=0 cap_refs={} open={3=0} mode=100644 size=0 mtime=2013-03-15 09:07:33.641899 caps=pAsXsFs objectset[1000000025f ts 0/0 objects 0 dirty

The inode in question is a file that's getting created/opened, closed, and then unlinked (part of the flush step in test-backtraces.py). The mds log is attached.

Associated revisions

Revision 4977f3ea - 04/11/2013 03:25 PM - Sam Lang

mds: Delay export on missing inodes for reconnect

The reconnect caps sent by the client on reconnect may not have
inodes found in the inode cache until after clientreplay (when
the client creates a new file, for example). Currently, we send an
export for that cap to the client if we don't see an inode in the cache
and path_is_mine() returns false (for example, if the client didn't
send a path because the file was already unlinked).
Instead, we want to delay handling of the reconnect cap until
clientreplay completes.

This patch modifies handle_client_reconnect() so that we don't assume
the cap isn't ours if we don't have an inode for it, but instead delay
recovery for later. An export cap message is only sent if the inode exists
and the cap isn't ours (non-auth) during reconnect. If any remaining
recovered caps exist in the recovered list once the mds goes active, we
send export messages at that point.

Also, after removing the path_is_mine check,
MDCache::parallel_fetch_traverse_dir() needs to skip non-auth dirfrags.

Fixes #4451.
Signed-off-by: Sam Lang <sam.lang@inktank.com>
Signed-off-by: Yan, Zheng <zheng.z.yan@intel.com>
Reviewed-by: Yan, Zheng <zheng.z.yan@intel.com>
Reviewed-by: Greg Farnum <greg@inktank.com>

History

#1 - 03/15/2013 09:31 AM - Greg Farnum

For some reason the MDS is sending back an "export" on the caps for that inode (timestamp 2013-03-15 09:07:38.098273), and no others. "Export" really shouldn't appear in a single-MDS system, and the only way it could happen is if during reconnect the MDS somehow thinks it's not auth for the inode in question. Maybe this is an edge case I'm not aware of and the client is supposed to repair it?
I do see that the unlink happens afterwards but it appears to be in replay...

#2 - 03/15/2013 09:37 AM - Ian Colle

- Priority changed from Normal to High

#3 - 03/15/2013 01:58 PM - Sage Weil

- Priority changed from High to Urgent

#4 - 03/15/2013 02:23 PM - Greg Farnum

Looked at this again briefly. I notice:
1) the inode was previously in the stray directory (before MDS restart)
2) the inode isn't appearing in the MDS log during replay
3) when checking for auth, it's checking path_is_mine, but the path consists solely of the inode number (which the MDS doesn't know about as it wasn't read into cache)
4) the client should have included a path if it had one (looks to me like it does).

This looks to me like we've got a small design problem, not just an implementation bug. But perhaps there's a design around this that I'm unaware of. Sage?

#5 - 03/20/2013 09:45 AM - Sam Lang

- File log.7db added

Uploaded an annotated log with only the lines related to the inode exhibiting the problem. The problem occurs from the following pattern:

1) create file, return ino 0xf00 in unsafe reply, journal write offset: 100
2) unlink file, return unsafe reply, journal write offset: 200
3) mds crash (neither of those two ops get written to the journal)
4) mds replay at journal 50-100 (doesn't replay past 100)
5) client reconnect with caps for 0xf00
6) mds doens't know about 0xf00 since it never made it to the journal
7) mds sends client cap export
8) client unmount causes hang due to 0xf00 never getting removed from cache (because mds doesn't know about it)

So the bug here appears to be at the client. We need to remove inodes and caps from the cache if we are replaying unsafe requests to the mds, and we need to do so before we send a reconnect. I'm still looking at the client code to figure out the right approach there.

Also, it turns out the right way to reproduce this is not to mimic the test-backtrace.py workload (which sets the segment size to 16K and max_segments to 1), but to create and remove a file and then crash the mds immediately after the unlink. I'm working on a test that triggers it reliably now.

#6 - 04/01/2013 01:06 PM - Ian Colle

- Target version set to v0.61 - Cuttlefish

#7 - 04/05/2013 10:25 AM - Sam Lang

- Status changed from In Progress to Fix Under Review

Pushed a proposed fix to wip-4451. The fix is to not adjust the conditional for checking if an inode is auth or not. If we don't have the inode in cache, we just delay the export until the mds goes active, so that the client has a chance to replay requests (such as those that create the inode we're looking for). If we still have recovered caps in the mdcache once we go active, those caps are designated for export.

This needs a review.

#8 - 04/05/2013 03:37 PM - Zheng Yan

After removing the path_is_mine check in Server::handle_client_reconnect(), I think we should also call mdcache->rejoin_export_caps() if we don't have the inode.

Another suggestion: We can process the leftover cap imports and send cap exports in batch when we go active. This avoids introducing class C_MDS_CapExports and Server::_do_cap_exports().

#9 - 04/08/2013 09:13 AM - Sam Lang

"After removing the path_is_mine check in Server::handle_client_reconnect(), I think we should also call mdcache->rejoin_export_caps() if we don't have the inode."

I remove the path_is_mine() call, because the caps sent on reconnect only have the ino, so the path_is_mine() check reduces to (in && in->is_auth()).

The case I'm concerned with is one where the client creates a file (and gets back an early reply), but the inode doesn't make it into the journal before the mds gets restarted, so the inode is unknown to the mds when the client sends the reconnect caps. Before reconnect, the client sends a replay request of the create, but that request gets delayed until the mds transitions to clientreplay.

Regarding the cap export, is it possible that the client has a cap that it thinks belongs to the mds, but the mds doesn't have the inode in cache after restart because its been exported? It doesn't look like it, but I'm not familiar enough with the export/migrate code to know definitively.

"Another suggestion: We can process the leftover cap imports and send cap exports in batch when we go active. This avoids introducing class C_MDS_CapExports and Server::_do_cap_exports()."

That would be cleaner, but its the reconnect code that knows about which Session to send the cap exports on, whereas in MDS::active_start(), we don't have that info. I suppose we could handle all sessions in MDS::active_start(), assuming all clients will have reconnected...

#10 - 04/08/2013 09:43 AM - Zheng Yan

"Regarding the cap export, is it possible that the client has a cap that it thinks belongs to the mds, but the mds doesn't have the inode in cache after restart because its been exported? It doesn't look like it, but I'm not familiar enough with the export/migrate code to know definitively"

Client may request caps from non-auth mds, so it's possible.

"That would be cleaner, but its the reconnect code that knows about which Session to send the cap exports on, whereas in MDS::active_start(), we don't have that info. "

I think we have that info in mdcache->cap_imports, because when a cap get reconnected we erase corresponding entry from it. When MDS goes active, we should export all leftover entries in mdcache->cap_imports.

#11 - 04/08/2013 10:59 AM - Greg Farnum

Zheng Yan wrote:

> "Regarding the cap export, is it possible that the client has a cap that it thinks belongs to the mds, but the mds doesn't have the inode in cache after restart because its been exported? It doesn't look like it, but I'm not familiar enough with the export/migrate code to know definitively"
>
> Client may request caps from non-auth mds, so it's possible.

Although I think the MDS would need to have the inode in cache for that to happen — it would have been replicated and that requires data being on-disk, right?

> "That would be cleaner, but its the reconnect code that knows about which Session to send the cap exports on, whereas in MDS::active_start(), we don't have that info. "
>
> I think we have that info in mdcache->cap_imports, because when a cap get reconnected we erase corresponding entry from it. When MDS goes active, we should export all leftover entries in mdcache->cap_imports.

Yeah, I like this idea better. Batching is good!

Also, while we're here we should figure out if we need to do anything with that FIXME about the migrate_seq...

#12 - 04/08/2013 08:04 PM - Zheng Yan

Greg Farnum wrote:

> Although I think the MDS would need to have the inode in cache for that to happen — it would have been replicated and that requires data being on-disk, right?

Yes, the inode should be logged or on-disk and its auth MDS should have it in the cache. Non-auth MDS can issue read caps to client, client sends cap reconnect to it, when it restarts. But it's likely the non-auth MDS doesn't have the inode in the cache after restart, so we need export the cap to its

auth MDS.

Greg Farnum wrote:

> Also, while we're here we should figure out if we need to do anything with that FIXME about the migrate_seq...

I think MDCache::rejoin_import_cap() should increase cap->mseq if frommds >= 0 and client should set cap->mseq to 0 when composing the reconnect message. This guarantees that cap import message's migrate_seq is larger than corresponding export message's migrate_seq.

Besides I think MDCache::export_remaining_imported_caps() doesn't need the inode-in-cache check, the patch should be something like:

```
diff --git a/src/mds/MDCache.cc b/src/mds/MDCache.cc
index 3f090bb..46a14e9 100644
--- a/src/mds/MDCache.cc
+++ b/src/mds/MDCache.cc
@@ -5045,6 +5045,28 @@ void MDCache::rejoin_import_cap(CInode *in, client_t client, ceph_mds_cap_reconn
     do_cap_import(session, in, cap);
 }

+void MDCache::export_remaining_imported_caps()
+{
+
+  dout(10) << "export_remaining_imported_caps" << dendl;
+  map<inodeno_t,map<client_t, map<int,ceph_mds_cap_reconnect> > >::iterator p = cap_imports.begin();
+  while (p != cap_imports.end()) {
+    for (map<client_t, map<int,ceph_mds_cap_reconnect> >::iterator q = p->second.begin();
+     q != p->second.end();
+     ++q) {
+      Session *session = mds->sessionmap.get_session(entity_name_t::CLIENT(p->first.v));
+      if (session) {
+    // mark client caps stale.
+    MClientCaps *stale = new MClientCaps(CEPH_CAP_OP_EXPORT, q->first, 0, 0, 0);
+    //stale->head.migrate_seq = 0; // FIXME ******
+    mds->send_message_client_counted(stale, q->first);
+      }
+    }
+  }
+
+  cap_imports.clear();
+}
+
 void MDCache::try_reconnect_cap(CInode *in, Session *session)
 {
   client_t client = session->info.get_client();
diff --git a/src/mds/MDCache.h b/src/mds/MDCache.h
index 73780e2..d837586 100644
--- a/src/mds/MDCache.h
+++ b/src/mds/MDCache.h
@@ -486,6 +486,7 @@ public:
   void rejoin_import_cap(CInode *in, client_t client, ceph_mds_cap_reconnect& icr, int frommds);
   void finish_snaprealm_reconnect(client_t client, SnapRealm *realm, snapid_t seq);
   void try_reconnect_cap(CInode *in, Session *session);
+  void export_remaining_imported_caps();

   // cap imports.  delayed snap parent opens.
   //  realm inode -> client -> cap inodes needing to split to this realm
diff --git a/src/mds/MDS.cc b/src/mds/MDS.cc
index 3b3b2d6..935fb0c 100644
--- a/src/mds/MDS.cc
+++ b/src/mds/MDS.cc
@@ -1504,6 +1504,7 @@ void MDS::active_start()

   mdcache->clean_open_file_lists();
   mdcache->scan_stray_dir();
+  mdcache->export_remaining_imported_caps();
   finish_contexts(g_ceph_context, waiting_for_replay);  // kick waiters
   finish_contexts(g_ceph_context, waiting_for_active);  // kick waiters
 }
diff --git a/src/mds/Server.cc b/src/mds/Server.cc
index 293640e..cf4f310 100644
--- a/src/mds/Server.cc
```

```
+++ b/src/mds/Server.cc
@@ -636,15 +636,12 @@ void Server::handle_client_reconnect(MClientReconnect *m)
     }

     filepath path(p->second.path, (uint64_t)p->second.capinfo.pathbase);
-    if ((in && !in->is_auth()) ||
-    !mds->mdcache->path_is_mine(path)) {
+    if (in && !in->is_auth()) {
       // not mine.
       dout(0) << "non-auth " << p->first << " " << path
           << ", will pass off to authority" << dendl;

       // mark client caps stale.
-      inode_t fake_inode;
-      fake_inode.ino = p->first;
       MClientCaps *stale = new MClientCaps(CEPH_CAP_OP_EXPORT, p->first, 0, 0, 0);
       //stale->head.migrate_seq = 0; // FIXME ******
       mds->send_message_client_counted(stale, session);
@@ -652,11 +649,11 @@ void Server::handle_client_reconnect(MClientReconnect *m)
       // add to cap export list.
       mdcache->rejoin_export_caps(p->first, from, p->second);
     } else {
-      // mine.  fetch later.
+      // don't know if the inode is mine
       dout(0) << "missing " << p->first << " " << path
-          << " (mine), will load later" << dendl;
-      mdcache->rejoin_recovered_caps(p->first, from, p->second,
-                     -1);  // "from" me.
+          << " will load or export later" << dendl;
+      mdcache->rejoin_recovered_caps(p->first, from, p->second, -1);
+      mdcache->rejoin_export_caps(p->first, from, p->second);
     }
   }
```

#13 - 04/09/2013 06:00 AM - Zheng Yan

- File patch added

After removing the path_is_mine check, MDCache::parallel_fetch_traverse_dir() needs skip non-auth dirfrags. The modified patch passed 100+ rounds of random MDS restart.

#14 - 04/09/2013 08:48 AM - Sam Lang

Thanks Yan for fixing up that patch and testing it out. The inode check was just cruft from the previous changes, and definitely should be removed. The migrate_seq changes look good to me as well, I've updated the commit and re-pushed wip-4451.

#15 - 04/09/2013 04:27 PM - Greg Farnum

Does this need more review or just testing? (I ask because I notice you've got two reviewed-by tags on it, although I haven't checked out the current form yet in any detail so let me know if I should.)

#16 - 04/09/2013 07:49 PM - Sam Lang

Please review again based on the latest changed pushed to wip-4451.

#17 - 04/11/2013 09:06 AM - Greg Farnum

- Status changed from Fix Under Review to Resolved

Merged into next via commit:e32849c4eef2f5d911288aabeac0a6967b1e6ae4

I'm electing not to backport this despite its applicability to Bobtail as a general policy around the MDS and its less-supported state. Please comment if that seems inappropriate for some reason!

Files

mds.a.log.gz	17 MB	03/15/2013	Sam Lang
log.7db	11.8 KB	03/20/2013	Sam Lang
patch	5.45 KB	04/09/2013	Zheng Yan