From the mailing list

I am testing the tiering functionality with cephfs. I used a replicated cache with an EC data pool, and a replicated metadata pool like this:

```bash
ceph osd pool create cache 1024 1024
ceph osd pool set cache size 2
ceph osd pool set cache min_size 1
ceph osd erasure-code-profile set profile11 k=8 m=3 ruleset-failure-domain=osd
ceph osd pool create ecdata 128 128 erasure profile11
ceph osd tier add ecdata cache
ceph osd tier cache-mode cache writeback
ceph osd tier set-overlay ecdata cache
ceph osd pool set cache hit_set_type bloom
ceph osd pool set cache hit_set_count 1
ceph osd pool set cache hit_set_period 3600
ceph osd pool set cache target_max_bytes $((280*1024*1024*1024))
ceph osd pool create metadata 128 128
ceph osd pool set metadata crush_ruleset 1 # SSD root in crushmap
ceph fs new ceph_fs metadata cache      <-- wrong ?
```

I started testing with this, and this worked, I could write to it with cephfs and the cache was flushing to the ecdata pool as expected. But now I notice I made the fs right upon the cache, instead of the underlying data pool. I suppose I should have done this:

```bash
ceph fs new ceph_fs metadata ecdata
```

So my question is: Was this wrong and not doing the things I thought it did, or was this somehow handled by ceph and didn't it matter I specified the cache instead of the data pool?

This basically works, but gets awkward later on. I'm not sure what's involved in blocking the use of cache pools, and I'm not sure we want to fix it as an FS-specific thing rather than somehow doing it generically in the monitor command system. But we should do something.
I mean something like this (although I'm not positive I got all the requirements right):

```c
diff --git a/src/mon/MDSMonitor.cc b/src/mon/MDSMonitor.cc
index fea198c..14fb003 100644
--- a/src/mon/MDSMonitor.cc
+++ b/src/mon/MDSMonitor.cc
@@ -985,6 +985,15 @@ bool MDSMonitor::management_command(
     return true;
 }
 
+    const pg_pool_t *p = mon->osdmon()->osdmap.get_pg_pool(metadata);
+    assert(p != NULL);
+    if (p->is_tier() && p->cache_mode != pg_pool_t::CACHEMODE_NONE) {
+      r = -EINVAL;
+      ss << "pool '" << metadata_name << "' cannot be a cache pool";
+      return true;
+    }
+}
+
#3 - 09/11/2014 01:07 PM - Greg Farnum
Yeah, that's the simple solution. I was also wondering though if we wanted to do something more sophisticated trying to have the OSDMap or similar structures just transparently map from the cache pool to the base pools.

#4 - 09/11/2014 01:13 PM - Sage Weil
My vote is NAK on this. This is exactly what I want to do on my cluster and this is the only way EC can be used for CephFS currently. Why do we want to prevent it?

#5 - 09/11/2014 01:32 PM - Greg Farnum
This conversation is getting split across several mediums, but this shouldn't prevent specifying the use of a base pool which is EC with a cache pool that is replicated. Just the hooking of the filesystem's metadata to a pool which is sort of "ephemeral" like cache pools are.

#6 - 09/12/2014 01:43 PM - Greg Farnum
- Status changed from New to Verified
- Assignee set to Greg Farnum
- Target version set to v0.87

The user pointed out that right now we prevent assigning EC pools to CephFS. I believe this is the result of a user who set an EC pool as their data pool WITHOUT using a cache pool on top of it. I'm not sure if the better user experience here is the nasty hack we're using now where we set the cache pool, or if we should only accept base pools but an EC base pool has to have a cache (this is what I'm leaning towards), or if there's some other option.

The issue with choice 2 is that the user could remove the cache pool from in front...which we could prevent in the OSDMonitor by having it check for use in the FS?

11/25/2015
We do already have OSDMonitor::_check_remove_pool, so we could add similar logic when users run remove-overlay...maybe...

#7 - 09/12/2014 01:43 PM - Greg Farnum
- Assignee deleted (Greg Farnum)

#8 - 09/12/2014 01:48 PM - Sage Weil
I lean toward setting data pool to the base pool too. I worry about having to stand up so many guard rails, though. Hopefully we can consolidate the checks into "is_safe_to_remove" and "is_safe_to_remove_tier" type functions?

#9 - 09/12/2014 01:53 PM - Greg Farnum
Yes, that's what I'm hoping as well. That's what _check_remove_pool() is; we'd need to add an equivalent for tiering. And, maybe, have it check when changing the tier type as well... :/

#10 - 09/15/2014 12:47 PM - John Spray
- Status changed from Verified to In Progress
- Assignee set to John Spray

#11 - 09/15/2014 01:04 PM - John Spray
First half here: https://github.com/ceph/ceph/tree/wip-9435 (no handling of tiering updates yet)

#12 - 09/16/2014 02:57 AM - John Spray
- Status changed from In Progress to Need Review

https://github.com/ceph/ceph/pull/2507

#13 - 09/16/2014 05:37 PM - Greg Farnum
- Status changed from Need Review to Resolved

Merged into giant branch in commit:eb1b2e0072bf605095f1404c2b6c2abfba216dbe