While looking closely at straw vs. straw2 buckets I realized that one property of CRUSH/straw that I thought was true is in fact not true. What I expected is, given the following:

- two OSDs with ids x and y
- OSD x fails and is replaced
- the replacement OSD gets a new id y
- OSD x is removed from CRUSH
- OSD y is added to CRUSH at the same location and with the same weight that x had

then:

- OSD y should get the same PGs that x had
- there should be no data movement on other OSDs in the cluster

But this turns out to be not true. And since we rely on this falsehood in our operations procedures, our disk replacements are moving a lot more data than they should.

Here is my example.

We start with crush.txt.orig:

```plaintext
# begin crush map
tunable choose_local_tries 0
tunable choose_local_fallback_tries 0
tunable choose_total_tries 50
tunable chooseleaf_descend_once 1
tunable straw_calc_version 1

devices
device 0 osd.0
device 1 osd.1
device 2 osd.2
device 3 osd.3

types
type 0 device
type 1 host
type 2 default

buckets
```

```plaintext
# devices
device 0 osd.0
device 1 osd.1
device 2 osd.2
device 3 osd.3

types
type 0 device
type 1 host
type 2 default

buckets
```
host host0 {
    id -1           # do not change unnecessarily
    # weight 2.000
    alg straw
    hash 0 # rjenkins1
    item osd.0 weight 1.000
    item osd.1 weight 1.000
}

host host1 {
    id -2           # do not change unnecessarily
    # weight 2.000
    alg straw
    hash 0 # rjenkins1
    item osd.2 weight 1.000
    item osd.3 weight 1.000
}

default default {
    id -3           # do not change unnecessarily
    # weight 4.000
    alg straw
    hash 0 # rjenkins1
    item host0 weight 2.000
    item host1 weight 2.000
}

# rules
rule replicated_ruleset {
    ruleset 0
    type replicated
    min_size 1
    max_size 10
    step take default
    step chooseleaf firstn 0 type host
    step emit
}

# end crush map

Then after replacing osd.0 with osd.4 (to make crush.txt.new):

# begin crush map
tunable choose_local_tries 0
tunable choose_local_fallback_tries 0
tunable choose_total_tries 50
tunable chooseleaf_descend_once 1
tunable straw_calc_version 1

device 0 device0
device 1 osd.1
device 2 osd.2
device 3 osd.3
device 4 osd.4

type 0 device
type 1 host
type 2 default

# buckets
host host0 {
    id -1           # do not change unnecessarily
    # weight 2.000
    item osd.4 weight 1.000
}

alg straw
hash 0  # rjenkins1
item osd.4 weight 1.000
item osd.1 weight 1.000
}
host host1 {
  id -2           # do not change unnecessarily
  # weight 2.000
  alg straw
  hash 0  # rjenkins1
  item osd.2 weight 1.000
  item osd.3 weight 1.000
}
default default {
  id -3           # do not change unnecessarily
  # weight 4.000
  alg straw
  hash 0  # rjenkins1
  item host0 weight 2.000
  item host1 weight 2.000
}

# rules
rule replicated_ruleset {
  ruleset 0
  type replicated
  min_size 1
  max_size 10
  step take default
  step chooseleaf firstn 0 type host
  step emit
}

# end crush map

Then we test the new maps vs expected:

crushtool -c crush.txt.orig -o cm.orig
crushtool -c crush.txt.new -o cm.new
crushtool -i cm.orig --num-rep 2 --test --show-mappings > orig.mappings 2>&1
cat orig.mappings | sed -e 's/\[0/\[4/' | sed -e 's/0\]/4\]/' > expected.mappings
crushtool -i cm.new --num-rep 2 --test --show-mappings > actual.mappings 2>&1
cruchtool -i cm.orig --num-rep 2 --test --show-mappings > orig.mappings 2>&1
cruchtool -i cm.new --num-rep 2 --test --show-mappings > actual.mappings 2>&1
wc -l orig.mappings
diff -u expected.mappings actual.mappings | grep -c ^+

I get 344/1024 PGs which move. Comments?

History

#1 - 03/16/2015 06:35 PM - Greg Farnum
Those numbers sound too large, but yes, the placement depends on the osd ID (the other option is to make it depend on the order of placement
within a bucket, which is also not great).

Are the PGs moving between different hosts as well as between osd.0, osd.1, osd.4? Can you try this test on a larger simulated cluster and see what
the numbers look like?

#2 - 03/20/2015 11:04 AM - Dan van der Ster
You're right, it's an effect that gets smaller with increased numbers of OSDs.

Here are the results on our very large test cluster. First, the disk replacement:

```bash
# diff -u straw1.before.txt straw1.after.txt
--- straw1.before.txt   2015-03-20 11:05:20.538499920 +0100
+++ straw1.after.txt    2015-03-20 11:18:32.069817137 +0100
@@ -165,7 +165,7 @@
 device 154 osd.154
 device 155 osd.155
 device 156 osd.156
-device 157 osd.157
+device 157 device157
 device 158 osd.158
 device 159 osd.159
 device 160 osd.160
@@ -7366,6 +7366,7 @@
 device 7355 osd.7355
 device 7356 osd.7356
 device 7357 osd.7357
+device 7358 osd.7358
```

```bash
# types
type 0 osd
@@ -7531,7 +7532,7 @@
     # weight 173.760
     alg straw
     hash 0 # rjenkins1
-    item osd.157 weight 3.620
+    item osd.7358 weight 3.620
     item osd.101 weight 3.620
     item osd.100 weight 3.620
     item osd.120 weight 3.620
```

And then the compilations:

```bash
crushtool -c straw1.before.txt -o straw1.before.map
crushtool -c straw1.after.txt -o straw1.after.map
crushtool -i straw1.before.map --num-rep 3 --test --show-mappings > before.mappings
crushtool -i straw1.after.map --num-rep 3 --test --show-mappings > after.mappings
cat before.mappings | sed -e 's/\[157,/\[7358,/' | sed -e 's/,157,/,7358,/g' | grep -c ^+ > ideal.mappings
diff -u ideal.mappings after.mappings
```

37 out of 14336, but more accurately 4 out of 1024 mappings per relevant rule are changing. straw2 results in exactly the same differences.

On our production cluster with ~1000 OSDs the effect changes 28 out of 5120 mappings, or 6-7 out of 1024 per rule.

IMHO this is not insignificant, since these PG moves are spread across the cluster, thereby triggering backfilling on 10's of OSDs when replacing a disk.

BTW, I did also confirm that changing the OSD order within a bucket does not change the placement. If the choice is between allowing bucket re-ordering and allowing replacement with a different ID, it's not obvious to me which is the more useful behaviour. Would it be difficult to make this behaviour configurable?
I mentioned this to Sage and he was surprised that it invoked backfill elsewhere in the cluster.

Doing it based on bucket position wouldn't be viable for straw buckets though, because reordering entries would move all the data. I suggested we could switch things to assign an internal per-bucket ID to each entry, but there's still some issue with existing bucket IDs I didn't quite understand. Not sure if there's more information to gather or if we just need to think about it for a while.

---

Ah, I understand the problem now.

So, in the scenario where you fail a disk and mark it out, recover, and then later add a new disk in the same position, this doesn't matter. It'll get different data but in both cases two disk's worth of data is copied (once for initial recovery after ~5 minutes, another when the replacement is added back in).

I take it in your environment you aren't marking the OSD out on failure, and are instead replacing disks quickly after a failure?

I think there are two possible fixes:
- re-use the same osd ID
- add an id to the crush bucket that can be set explicitly. this covers some other use-cases too but means an algorithm changes and tunable and all that. probably worth doing in the long term.

---

I take it in your environment you aren't marking the OSD out on failure, and are instead replacing disks quickly after a failure?

Actually, not really. We mark do like you said:

mark it out, recover, and then later add a new disk in the same position

but I don't understand why you said:

this doesn't matter

because my observation is that a few PGs that have nothing to do with the failed (or new) OSD are also moving.
Dan van der Ster wrote:

I take it in your environment you aren't marking the OSD out on failure, and are instead replacing disks quickly after a failure?

Actually, not really. We mark do like you said:

mark it out, recover, and then later add a new disk in the same position

but I don't understand why you said:

this doesn't matter

because my observation is that a few PGs that have nothing to do with the failed (or new) OSD are also moving.

Ah, yeah. Because there's a hidden step in there:

1. osd marked out (~1 disk of data moves)
2. osd id deleted and crush weight (effectively) zeroed
3. new osd id added with same weight

and the 2+3 combination is going to move a bunch of stuff. And if you stopped after 2 and waited before doing 3 you'd see ~3 disks worth of data move in total.

In any case, the fixes are still the same:

1. re-use the osd id
2. modify the crush format/algorithm to have another id that can be explicitly specified.

The UX for #2 may be somewhat annoying.. although perhaps we can make it such that in the normal case where there is only 1 failure the same internal id will get reused by default.

Alternatively.. we may be able to accomplish #1 in the monitor (outside of CRUSH). It could remember failed + removed id's by their crush position in the hierarchy. If a new OSD is created in that position it could try to re-use the same id. We'd need to pass the crush location into the 'ceph osd create ...' command for it to do that...