crush: straw is dead, long live straw2

[Date Prev][Date Next][Thread Prev][Thread Next][Date Index][Thread Index]

- *Subject*: crush: straw is dead, long live straw2
- *From*: Sage Weil <sweil@xxxxxxxxxx>
- *Date*: Mon, 8 Dec 2014 15:48:01 -0800 (PST)
- *User-agent*: Alpine 2.00 (DEB 1167 2008-08-23)

0

Hey everyone,

When I was writing the original CRUSH code ages ago, I made several
different bucket types, each using a different 'choose' algorithm for
pseudorandomly selecting an item. Most of these were modeled after the
original RUSH algorithms from RJ Honicky, but there was one new bucket
type I called 'straw' with some new properties that I was rather proud of:

1- items could have any weight.
2- O(n) to choose an item.
3- if an item's weight was adjusted up or down, mappings would either
move to or from the adjusted item, but never between other unmodified
items in the bucket.

The basic motivation for the process to look like drawing straws, where
the longest straw would win, except that each item would have their random
straw lengths scaled based on their weight.

This is all fine and good, except that it turns out the third property
doesn't actually hold true! The amount that the random straw lengths got
scaled turned out to be a complicated function of the other weights in the
bucket. Although the placement calculation is pretty clearly independent
for each item, a change in the weights affects the scaling factors for
other items, which means that a weight change could affect the placement
of items between other items.

In retrospect this should have been obvious, but it turns out nobody has
looked too carefully at this bit of code or the underlying algorithm for
some 8 years. It took a customer making a deliberately miniscule change
and seeing a huge data movement to dig into this bit of behavior. Also in
retrospect, a well written test that verified the third property held true
for various changes would have caught it, but my testing at the time was
limited to strange combinations of weights.

We suspect that this has actually been noticed by lots of people who have
complained about more data moving after a weight change than they
expected. These sorts of reports are hard to quantify and investigate and
are easy to ignore. Sigh.

Anyway, that's the bad news.

The good news is that we figured out how to fix the placement algorithm to
get all three properties. The new bucket type will be called 'straw2'
(unless someone has a better suggestion). Here's the difference.

For straw buckets, the choose function looks roughly like this:

```
 max_x = -1
 max_item = -1
```

```
 for each item:
    x = random value from 0..65535
    x *= scaling factor
    if x > max_x:
       max_x = x
       max_item = item
 return item
```

The issue, as I mentioned, is that the scaling factor is a strange function of *all* of the other item weights, which means a change in a single item A's weight could affect the relative distribution of items previously on B and C.

The new straw2 bucket works like this:

```
 max_x = -1
 max_item = -1
 for each item:
    x = random value from 0..65535
    x = ln(x / 65536) / weight
    if x > max_x:
       max_x = x
       max_item = item
 return item
```

That ln() is a natural log (well, a 16-bit fixed-point approximation of it) and as you can see it's a simple function of the weight of that item. That means that changing one item's weight won't have any effect on the straw lengths for other items, so a change will either make the item win or not win but won't change who the other winner is in the not-win case.

Somewhat embarassingly, I stumbled onto this solution half by accident. Sam found the underlying principle that makes it work:

http://en.wikipedia.org/wiki/Exponential_distribution#Distribution_of_the_minimum_of_exponential_random_variables

Calculating the ln() function is a bit annoying because it is a floating point function and CRUSH is all fixed-point arithmetic (integer-based). The current draft implementation uses a 128 KB lookup table (2 bytes per entry for 16 bits of input precision). It seems to be about 25% slower than the original straw code in my simple microbenchmark, but I'm not sure that's a number we should trust since it loops through a zillion inputs and will pull most of the lookup table into the processor caches.

We could probably try a few others things:

- Lose half the precision to reduce the table size. I don't think this will work well, and in most cases we'll still miss the CPU cache so the benefit is marginal at best.

- Use floating point log function. This is problematic for the kernel implementation (no floating point), is slower than the lookup table, and makes me worry about whether the floating point calculations are consistent across architectures (the mapping has to be completely deterministic).

- Use some approximation of the logarithm with fixed-point arithmetic. I spent a bit of time search and found

http://www.researchgate.net/publication/230668515_A_fixed-point_implementation_of_the_natural_logarithm_based_on_a_expanded_hyperbolic_CORDIC_algorithm

but it also involves a couple of lookup tables and (judging by figure 1) is probably slower than the 256 KB table.

We could probably expand out taylor series and get something half decent, but any precision we lose will translate to OSD utilizations that are off from the input weights--something that costs real disk space and we probably want to avoid.

```
 - Stick with the 128KB lookup table.  Performance sensitive clients can
precalculate all PG mappings when they get OSDMap updates if they are
concerned about leave the CRUSH calculation in the IO path.

Any other suggestions?

Here is my implementation of the lookup-table based approach:

        https://github.com/ceph/ceph/commit/1d462c9f6a262de3a51533193ed2dff34c730727
        https://github.com/ceph/ceph/commits/wip-crush-straw2

You'll notice that included in there is a unit test that verifies that
changing a single item's weight does not effect the distribution of inputs
among other items in the bucket.  :)

Thanks-
sage
--
To unsubscribe from this list: send the line "unsubscribe ceph-devel" in
the body of a message to majordomo@xxxxxxxxxxxxxxx
More majordomo info at  http://vger.kernel.org/majordomo-info.html
```

- **Follow-Ups**:
  - **Re: crush: straw is dead, long live straw2**
    - *From:* Loic Dachary
  - **Re: crush: straw is dead, long live straw2**
    - *From:* Yehuda Sadeh
  - **Re: crush: straw is dead, long live straw2**
    - *From:* Thorsten Behrens
  - **Re: crush: straw is dead, long live straw2**
    - *From:* Josh Durgin

- Prev by Date: **Re: rados read ordering**
- Next by Date: **Re: crush: straw is dead, long live straw2**
- Previous by thread: **wip-claim-2**
- Next by thread: **Re: crush: straw is dead, long live straw2**
- Index(es):
  - **Date**
  - **Thread**