Commit 958a2765 authored 4 years ago by Ilya Dryomov

crush: straw2 bucket type with an efficient 64-bit crush_ln()

```
This is an improved straw bucket that correctly avoids any data movement
between items A and B when neither A nor B's weights are changed.  Said
differently, if we adjust the weight of item C (including adding it anew
or removing it completely), we will only see inputs move to or from C,
never between other items in the bucket.

Notably, there is not intermediate scaling factor that needs to be
calculated.  The mapping function is a simple function of the item weights.

The below commits were squashed together into this one (mostly to avoid
adding and then yanking a ~6000 lines worth of crush_ln_table):

- crush: add a straw2 bucket type
- crush: add crush_ln to calculate nature log efficently
- crush: improve straw2 adjustment slightly
- crush: change crush_ln to provide 32 more digits
- crush: fix crush_get_bucket_item_weight and bucket destroy for straw2
- crush/mapper: fix divide-by-0 in straw2
  (with div64_s64() for draw = ln / w and INT64_MIN -> S64_MIN - need
   to create a proper compat.h in ceph.git)

Reflects ceph.git commits 242293c908e923d474910f2b8203fa3b41eb5a53,
                          32a1ead92efcd351822d22a5fc37d159c65c1338,
                          6289912418c4a3597a11778bcf29ed5415117ad9,
                          35fcb04e2945717cf5cfe150b9fa89cb3d2303a1,
                          6445d9ee7290938de1e4ee9563912a6ab6d8ee5f,
                          b5921d55d16796e12d66ad2c4add7305f9ce2353.
Signed-off-by: Ilya Dryomov <idryomov@gmail.com>
```

parent 45002267 master ···

No related merge requests found

Showing 5 changed files with 316 additions and 2 deletions

include/linux/crush/crush.h

```diff
@@ -96,13 +96,15 @@ struct crush_rule {
 96  96  *  uniform        O(1)       poor       poor
 97  97  *  list           O(n)       optimal    poor
 98  98  *  tree           O(log n)   good       good
 99     -  *  straw          O(n)       optimal    optimal
     99 +  *  straw          O(n)       better     better
    100 +  *  straw2         O(n)       optimal    optimal
100 101  */
101 102  enum {
102 103          CRUSH_BUCKET_UNIFORM = 1,
103 104          CRUSH_BUCKET_LIST = 2,
104 105          CRUSH_BUCKET_TREE = 3,
105     -        CRUSH_BUCKET_STRAW = 4
    106 +        CRUSH_BUCKET_STRAW = 4,
    107 +        CRUSH_BUCKET_STRAW2 = 5,
106 108  };
107 109  extern const char *crush_bucket_alg_name(int alg);
108 110
...  ...  @@ -149,6 +151,11 @@ struct crush_bucket_straw {
149 151          __u32 *straws;         /* 16-bit fixed point */
150 152  };
151 153
    154 + struct crush_bucket_straw2 {
    155 +        struct crush_bucket h;
    156 +        __u32 *item_weights;   /* 16-bit fixed point */
    157 + };
    158 +
152 159
```

```
153 160
154 161   /*
... ...   @@ -189,6 +196,7 @@ extern void crush_destroy_bucket_uniform(struct crush_bucket_uniform *b);
189 196   extern void crush_destroy_bucket_list(struct crush_bucket_list *b);
190 197   extern void crush_destroy_bucket_tree(struct crush_bucket_tree *b);
191 198   extern void crush_destroy_bucket_straw(struct crush_bucket_straw *b);
    199 + extern void crush_destroy_bucket_straw2(struct crush_bucket_straw2 *b);
192 200   extern void crush_destroy_bucket(struct crush_bucket *b);
193 201   extern void crush_destroy_rule(struct crush_rule *r);
194 202   extern void crush_destroy(struct crush_map *map);
... ...
```

net/ceph/crush/crush.c

```
... ...   @@ -17,6 +17,7 @@ const char *crush_bucket_alg_name(int alg)
17  17            case CRUSH_BUCKET_LIST: return "list";
18  18            case CRUSH_BUCKET_TREE: return "tree";
19  19            case CRUSH_BUCKET_STRAW: return "straw";
    20  +         case CRUSH_BUCKET_STRAW2: return "straw2";
20  21            default: return "unknown";
21  22            }
22  23    }
... ...   @@ -40,6 +41,8 @@ int crush_get_bucket_item_weight(const struct crush_bucket *b, int p)
40  41                    return ((struct crush_bucket_tree *)b)->node_weights[crush_calc_tree_node(p)];
41  42            case CRUSH_BUCKET_STRAW:
42  43                    return ((struct crush_bucket_straw *)b)->item_weights[p];
    44  +         case CRUSH_BUCKET_STRAW2:
    45  +                 return ((struct crush_bucket_straw2 *)b)->item_weights[p];
43  46            }
44  47            return 0;
45  48    }
... ...   @@ -77,6 +80,14 @@ void crush_destroy_bucket_straw(struct crush_bucket_straw *b)
77  80            kfree(b);
78  81    }
79  82
    83  + void crush_destroy_bucket_straw2(struct crush_bucket_straw2 *b)
    84  + {
    85  +         kfree(b->item_weights);
    86  +         kfree(b->h.perm);
    87  +         kfree(b->h.items);
    88  +         kfree(b);
    89  + }
    90  +
80  91    void crush_destroy_bucket(struct crush_bucket *b)
81  92    {
82  93            switch (b->alg) {
... ...   @@ -92,6 +103,9 @@ void crush_destroy_bucket(struct crush_bucket *b)
92  103           case CRUSH_BUCKET_STRAW:
93  104                   crush_destroy_bucket_straw((struct crush_bucket_straw *)b);
94  105                   break;
    106 +         case CRUSH_BUCKET_STRAW2:
    107 +                 crush_destroy_bucket_straw2((struct crush_bucket_straw2 *)b);
    108 +                 break;
95  109           }
96  110   }
97  111
... ...
```

net/ceph/crush/crush_ln_table.h 0 → 100644

This diff is collapsed.

net/ceph/crush/mapper.c

```
... ...   @@ -20,6 +20,7 @@
20  20
21  21    #include <linux/crush/crush.h>
22  22    #include <linux/crush/hash.h>
    23  + #include "crush_ln_table.h"
23  24
24  25    /*
25  26     * Implement the core CRUSH mapping algorithm.
... ...   @@ -237,6 +238,102 @@ static int bucket_straw_choose(struct crush_bucket_straw *bucket,
237 238           return bucket->h.items[high];
```

```
 }

+// compute 2^44*log2(input+1)
+uint64_t crush_ln(unsigned xin)
+{
+	unsigned x=xin, x1;
+	int iexpon, index1, index2;
+	uint64_t RH, LH, LL, xl64, result;
+
+	x++;
+
+	// normalize input
+	iexpon = 15;
+	while(!(x&0x18000)) { x<<=1; iexpon--; }
+
+	index1 = (x>>8)<<1;
+	// RH ~ 2^56/index1
+	RH = __RH_LH_tbl[index1 - 256];
+	// LH ~ 2^48 * log2(index1/256)
+	LH = __RH_LH_tbl[index1 + 1 - 256];
+
+	// RH*x ~ 2^48 * (2^15 + xf), xf<2^8
+	xl64 = (int64_t)x * RH;
+	xl64 >>= 48;
+	x1 = xl64;
+
+	result = iexpon;
+	result <<= (12 + 32);
+
+	index2 = x1 & 0xff;
+	// LL ~ 2^48*log2(1.0+index2/2^15)
+	LL = __LL_tbl[index2];
+
+	LH = LH + LL;
+
+	LH >>= (48-12 - 32);
+	result += LH;
+
+	return result;
+}
+
+
+/*
+ * straw2
+ *
+ * for reference, see:
+ *
+ * http://en.wikipedia.org/wiki/Exponential_distribution#Distribution_of_the_minimum_of_exponential_random_variables
+ *
+ */
+
+static int bucket_straw2_choose(struct crush_bucket_straw2 *bucket,
+                                int x, int r)
+{
+        unsigned i, high = 0;
+        unsigned u;
+        unsigned w;
+        __s64 ln, draw, high_draw = 0;
+
+        for (i = 0; i < bucket->h.size; i++) {
+                w = bucket->item_weights[i];
+                if (w) {
+                        u = crush_hash32_3(bucket->h.hash, x,
+                                           bucket->h.items[i], r);
+                        u &= 0xffff;
+
+                        /*
+                         * for some reason slightly less than 0x10000 produces
+                         * a slightly more accurate distribution... probably a
+                         * rounding effect.
+                         *
+                         * the natural log lookup table maps [0,0xffff]
+                         * (corresponding to real numbers [1/0x10000, 1] to
+                         * [0, 0xffffffffffff] (corresponding to real numbers
+                         * [-11.090355,0]).
```

```
+                         */
+                        ln = crush_ln(u) - 0x1000000000000ll;
+
+                        /*
+                         * divide by 16.16 fixed-point weight.  note
+                         * that the ln value is negative, so a larger
+                         * weight means a larger (less negative) value
+                         * for draw.
+                         */
+                        draw = div64_s64(ln, w);
+                } else {
+                        draw = S64_MIN;
+                }
+
+                if (i == 0 || draw > high_draw) {
+                        high = i;
+                        high_draw = draw;
+                }
+        }
+        return bucket->h.items[high];
+ }
+
+
 static int crush_bucket_choose(struct crush_bucket *in, int x, int r)
 {
         dprintk(" crush_bucket_choose %d x=%d r=%d\n", in->id, x, r);
@@ -254,12 +351,16 @@ static int crush_bucket_choose(struct crush_bucket *in, int x, int r)
         case CRUSH_BUCKET_STRAW:
                 return bucket_straw_choose((struct crush_bucket_straw *)in,
                                            x, r);
+        case CRUSH_BUCKET_STRAW2:
+                return bucket_straw2_choose((struct crush_bucket_straw2 *)in,
+                                            x, r);
         default:
                 dprintk("unknown bucket %d alg %d\n", in->id, in->alg);
                 return in->items[0];
         }
 }

+
 /*
  * true if device is marked "out" (failed, fully offloaded)
  * of the cluster
```

net/ceph/osdmap.c

```
@@ -122,6 +122,22 @@ static int crush_decode_straw_bucket(void **p, void *end,
         return -EINVAL;
 }

+ static int crush_decode_straw2_bucket(void **p, void *end,
+                                        struct crush_bucket_straw2 *b)
+ {
+        int j;
+        dout("crush_decode_straw2_bucket %p to %p\n", *p, end);
+        b->item_weights = kcalloc(b->h.size, sizeof(u32), GFP_NOFS);
+        if (b->item_weights == NULL)
+                return -ENOMEM;
+        ceph_decode_need(p, end, b->h.size * sizeof(u32), bad);
+        for (j = 0; j < b->h.size; j++)
+                b->item_weights[j] = ceph_decode_32(p);
+        return 0;
+ bad:
+        return -EINVAL;
+ }
+
 static int skip_name_map(void **p, void *end)
 {
         int len;
@@ -204,6 +220,9 @@ static struct crush_map *crush_decode(void *pbyval, void *end)
                 case CRUSH_BUCKET_STRAW:
                         size = sizeof(struct crush_bucket_straw);
                         break;
+                case CRUSH_BUCKET_STRAW2:
```

```
+                        size = sizeof(struct crush_bucket_straw2);
+                        break;
                 default:
                         err = -EINVAL;
                         goto bad;
@@ -261,6 +280,12 @@ static struct crush_map *crush_decode(void *pbyval, void *end)
                         if (err < 0)
                                 goto bad;
                         break;
+                case CRUSH_BUCKET_STRAW2:
+                        err = crush_decode_straw2_bucket(p, end,
+                                (struct crush_bucket_straw2 *)b);
+                        if (err < 0)
+                                goto bad;
+                        break;
                 }
         }

```

Please register or sign in to comment