MapX论文分享ppt

论文分享

MapX: Controlled Data Migration in the Expansion of Decentralized Object-Based Storage Systems

目录

- 简介
- **问题**
- **方案**
- **测试**
- 想法

简介

MapX: Controlled Data Migration in the Expansion of Decentralized Object-Based Storage Systems

1. 论文信息

- 会议：FAST'20
- 作者：Li Wang（滴滴）；Yiming Zhang （NiceX Lab）；jiawei xu，Guangtao Xue（上交大）

2. 论文简介

- 问题：CRUSH 算法在集群扩容时容易产生**不受控制**的数据迁移，特别是集群扩容较大规模的时候，系统性能下降严重；
- 方案：**MapX**，一种基于CRUSH的实现**可控迁移的扩容**的扩展；
- 测试：在集群扩容期间，基于 MAPX 的系统比基于 CRUSH 的系统的尾延迟降低了 **4.25 倍**

问题

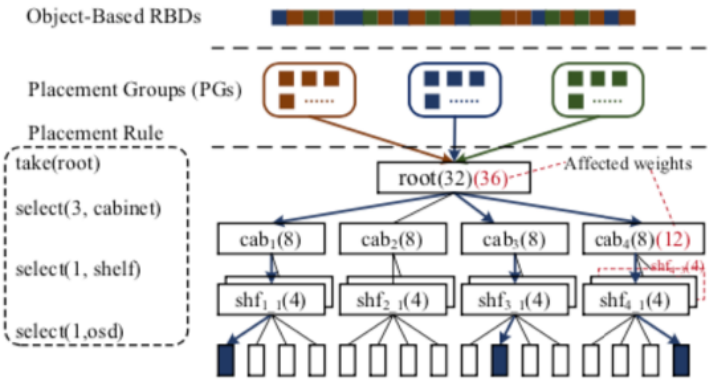
CRUSH算法

- 对象到层次集群映射的结构化
- buckets：root、cabinets、shelves
- 算法：

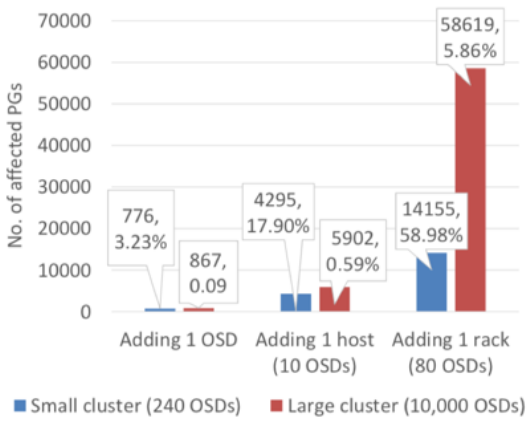
$$C(pgid,\vec{i},r)=\underset{i\in\vec{i}}{\operatorname{argmax}}\,HASH(pgid,r,ID(i))\times W(i)$$

缺点

- 很多不可控的数据迁移
- 在shf4扩容时，从下到上权重改变，会有Shf4之间和其它cab到cab4的数据迁移
- 迁移量可达h * ▲w/W



问题



具体测试结果

- 两个 Ceph 集群，均为三层结构，三副本
- 集群 1 对应 3 个cab，24 个shf，240 个 OSDs，24000 PGs，
- 集群 2 有 125 个cab，1000 个shf，10000 个 OSDs，100w PGs。
- 两个集群都执行扩容操作，粒度分别为 一个 OSD，一个shf，一个cab。

最坏近60%PGs受到影响

思考

问题产生的原因是什么呢?

CRUSH系统

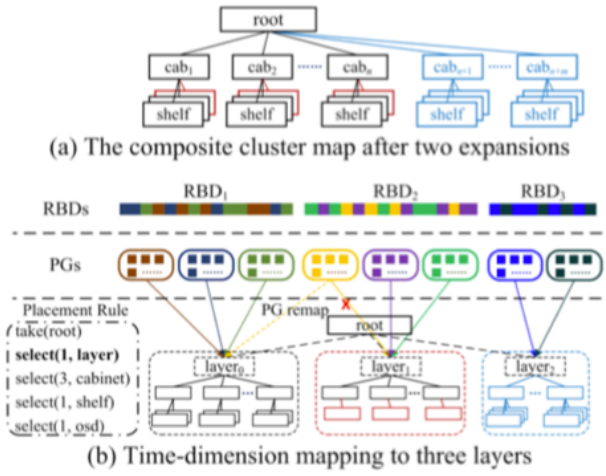
- 忽略了新旧对象/OSD 之间的差异
- *是否也可以在扩容时保持原有对象不变，只把新对象添加到新节点上?*

中心化对象存储系统（Haystack，Lustre，HDFS）

- 中心化的目录来记录对象的位置，来保证现有对象在扩容过程中不受影响，只把新的对象数据放置在新添加的 OSD 节点上
- 短时数据不平衡

MapX：保留 CRUSH 算法随机和均匀的优点的同时，引入**时间维度**的映射机制来区分新老对象/OSD

方案



(a) The composite cluster map after two expansions

(b) Time-dimension mapping to three layers

- 新增layer层，一个layer表示一次扩容
- 每个layer权值固定
- *每增加一个osd也要先增layer?*

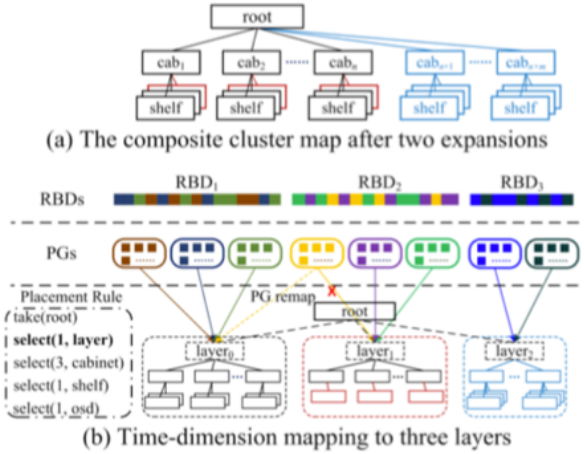
方案

Algorithm 1 Extended `select` Procedure of MAPX

```
1: procedure SELECT(number, type)
2:     if type ≠ "layer" then
3:         return CRUSH_SELECT(number, type)
4:     end if
5:     layers ← layers beneath currently-processing bucket
       ▷ each layer represents an expansion
6:     num_layers ← number of layers in layers
7:     pg ← current Placement Group
8:     o⃗ ← Φ                                   ▷ output list
9:     for (i = num_layers − 1; i ≥ 0; i − −) do
```

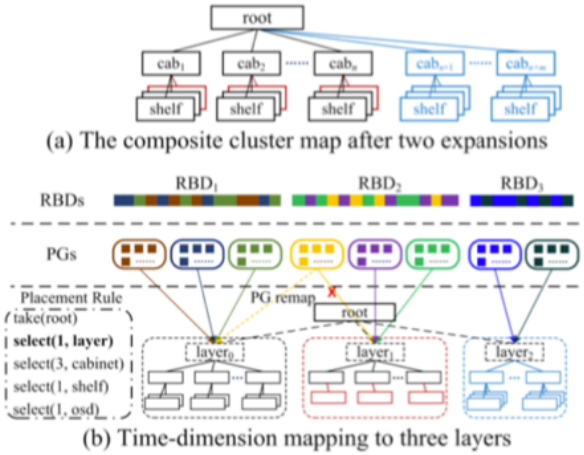
```
        layer ← layers[i]
        if layer.timestamp ≤ pg.timestamp then
            if layer was chosen by previous select then
                continue
            end if
            o⃗ ← o⃗ + {layer}
            number ← number − 1
            if number == 0 then
                break
            end if
        end if
    end for
    return o⃗
end procedure
```

方案



- 删除?
- PG 重映射：可以控制 PGs 到 Layer 的映射，来保证 layers 负载均衡
- 集群缩容：缩容时需要进行 PG 重映射调整负载，但也要保留部分元数据来保证映射关系不变
- layers 合并：使用时间戳来保证物理层的变化在逻辑层 layer 上保持相同以实现负载均衡

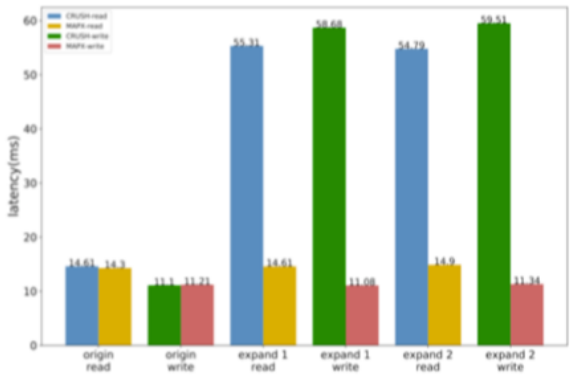
方案



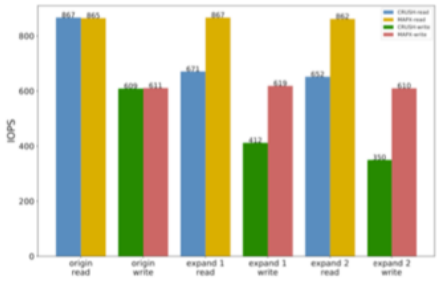
- 新增layer，一个layer表示一次扩容
- 每个layer权值固定

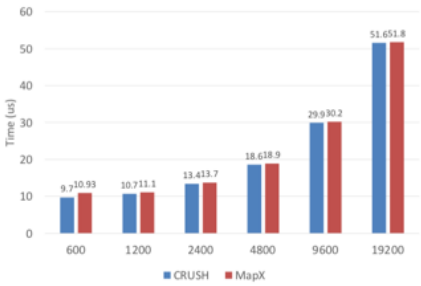
测试

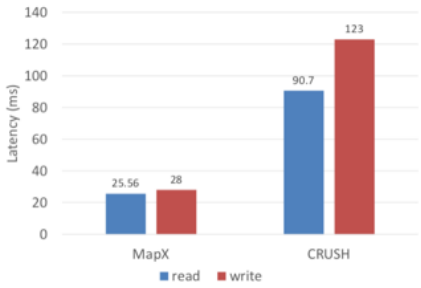
- 测试MAPX 和 传统 CRUSH
- 硬件环境：3台主机，每台主机配置：20 核 Xeon E5-2630 2.20GHz CPU，128G RAM，10GbE NIC，5.5 TB HDDs
- 软件环境：
  - OS：CentOS 7.0
  - Ceph：12.2 Luminous，BlueStore，Monitor co-located with one of the storage servers
  - Client：fio benchmark

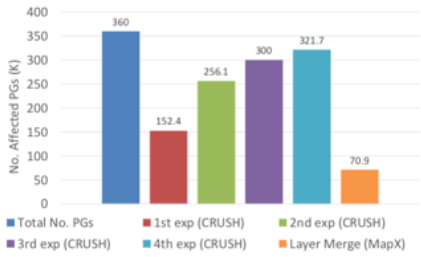


测试









想法

1. **关于论文**
   - 这篇论文本身没什么创新，就是用以前的思想来解决现有问题
   - 各种情况导致数据不平衡，论文中采用的是PG remapping，感觉描述太简单了；
   - 热点问题？缩容（最后提了一下，说效果可能表面上不会优于Ceph）？额外开销？
2. **关于CRUSH扩容问题**
   - 有不少论文提出， 在扩容时可能2 到 3 倍于理论迁移率的移动
   - CRUSH算法优化，Ceph实现上的优化（降低迁移的优先级）

- CRUSH算法优化，Ceph实现上的优化（降低迁移的优先级）
- 很多系统采用一致性hash算法来实现数据路由，可以有效减少扩容时的数据迁移；但其本身相对于Crush，需要额外的机制来进行建模，不能表示集群的层次结构，不够灵活