The Scrub and Repair in Jewel

Kefu Cai

Red Hat

October 18, 2015

The Scrub and Repair in Jewel

Kefu Cai

Red Hat

October 18, 2015

谢谢大家现在还醒着

The Scrub and Repair in Jewel

Kefu Chai

Red Hat

October 18, 2015

The Scrub and Repair in Jewel

Kefu Chai

Red Hat

October 18, 2015

Outline

Outline
scrub 101
pain in scrub
scrub in jewel
Q & A

1. 接下来想和诸位聊一下 ceph 里面的 scrub 和 repair

The Scrub and Repair in Jewel

```
$ sudo ceph health detail
HEALTH_ERR 1 pgs inconsistent; 2 scrub errors
pg 17.1c1 is active+clean+inconsistent, acting [21,25,30]
2 scrub errors
```

有的时候需要检查 osd 的 log，才能知道具体是哪个 object 有了问题，进而手动解决。

The Scrub and Repair in Jewel

```
$ sudo ceph health detail
HEALTH_ERR 1 pgs inconsistent; 2 scrub errors
pg 17.1c1 is active+clean+inconsistent, acting [21,25,30]
2 scrub errors

$ ceph pg repair 17.1c1
```

有的时候需要检查 osd 的 log，才能知道具体是哪个 object 有了问题，进而手动解决。

scrub

why

- fix the data lost/corruption when we can

scrub
why
- fix the data lost/corruption when we can

由于硬盘出错导致的 EIO，使得数据不可读。
或者更可怕的是，数据出错，同时客户端没有觉察，
把错误的数据当作正确的数据处理， 从而造成灾难性的损失。

scrub

when

- least impact to production

1. 限定在 begin hour 和 end hour 之间
2. 在系统负载超过阈值的时候也不会贸然启动
3. 以 osd 为单位，同时进行的 scrub 数量是可以配置的

scrub
when
- least impact to production

scrub

when

- least impact to production
- scheduled periodically

scrub
when
- least impact to production
- scheduled periodically

1. min/max interval 可配置

scrub
when

- least impact to production
- scheduled periodically
- distributed randomly in configured time range

scrub
when
- least impact to production
- scheduled periodically
- distributed randomly in configured time range

1. 试图缓解因为 max interval 导致的大量 scrub 任务堆积

scrub

how

- pg based and initialized by the primary
- scan in batch

scrub
how
- pg based and initialized by the primary
- scan in batch

1. 每次以 pg 为单位，由 primary 发起，要求所有的 peered 的 pg 参与。 分批次处理 pg 里面的所有 object，以 object 的 hash 为界。这就是所谓的 chunky scrub。把收集到的 object 信息，组织成一个 scrub map，同时也向 replica 请求相应的 scrub map。很明显，scrub map 会包含 object, xattr, omap 的信息， 以供交叉比较。

scrub

how

- pg based and initialized by the primary
- scan in batch
- select authoritive copy

scrub
how
- pg based and initialized by the primary
- scan in batch
- select authoritive copy

1. 和网购一样，一个 object 也有好评差评之分。比如， object 对应 hash 读不出来了； omap 读不出， object 读出来的大小或者 digest 不一致； 干脆读取的时候出现 EIO。都会导致一个 replica 失去成为权威拷贝的机会。
   只有通过这些考验的副本才能成为权威拷贝。而出问题的情况，不外乎读取失败，数据缺失，和数据不一致几种。

scrub

how

- pg based and initialized by the primary
- scan in batch
- select authoritive copy
- write down the results
- fix it (later)

scrub
how
- pg based and initialized by the primary
- scan in batch
- select authoritive copy
- write down the results
- fix it (later)

1. 把 scrub 的结果通过 cluster log 发给 monitor 记录下来。这些结果包括一些 metadata 本身不一致的情况，比如 head 和 snapdir 共存，有 clone， 但是 snapset 里面的 seq 是空的， 也有 snapset 本身出现不一致，比如说自己觉得有 3 个 snap，但是发现只有 2 个。
2. 对于 digest 缺失的情况，就直接用已知的 digest 重写所有的 replica。 如果数据本身不一致， 或者缺失，那么就把它放到 missing 列表里面去。 如果在以后处理 io 请求的时候，正好碰到了缺少的 object，就立即恢复它。即随机选择一个 replica，发送 pull 请求， 用收到的 push 数据重写本地有错误，或者缺少的 object。

pain in scrub

- ceph pg repair does not always work.
- scrub/repair is not programmable/scriptable.

pain in scrub
- ceph pg repair does not always work.
- scrub/repair is not programmable/scriptable.

现有的机制还非常简单，无法应对绝大多数情形。
有时候需要观察日志，删除出错的 replica。
没有实现更智能、更复杂的策略。
比如说少数服从多数的算法，虽然直觉，但是目前没有这个设计。
如果 librados 能为 scrub 提供更丰富的接口，
那么客户端就能灵活地制定策略，选择有效的副本，指定正确的数据，
甚至恢复 snapset/clone，当然这需要对 snapset 有一些理解。

new APIs

query

- get_inconsistent_pools() => pool[]
- get_inconsistent_pgs() => pg[]
- get_inconsistent(pg) => epoch, inconsistent[]

new APIs
query
- get_inconsistent_pools() => pool[]
- get_inconsistent_pgs() => pg[]
- get_inconsistent(pg) => epoch, inconsistent[]

在 scrub 过程中，osd
会把缺失或者不一致的信息收集起来。保存在临时 object 里面，供 scrub API 查询。在同一个 pg interval 的期间，可以根据这次 scrub 的结果 进行查询，读取，进行恢复的操作。 但是如果 interval 过了，scrub API 会返回 EAGAIN 。 inconsistent 会是一个带有版本号的 json，方便日后扩展。 是 osd 到 shard_info_t 的 map， shard_info_t 包含 replica 是否存在， omap_sha1, data_sha1, size 等 metadata。
基于这些数据，用户可以实现各种策略，比如说投票。

new APIs

read

- operate_on_shard(epoch, osd, shard, op)

new APIs
read
- operate_on_shard(epoch, osd, shard, op)

其中，op 可能是 read 操作，这些操作允许用户检查数据。
从而选择有效的数据副本，同时跳过 OSD 一致性的检查。

new APIs

choose

- specify the good/bad shard/xattr/omap replica
- specify the metadata (snapset in particular)

new APIs
choose
allows user to
- specify the good/bad shard/xattr/omap replica
- specify the metadata (snapset in particular)

前者本身就是数据，没有和其它对象有互相引用的关系。 但是 snapset 和 clone
有其自身的一致性问题。所以具体的接口还在讨论。
如果操作不当，可能会产生更多的不一致，有些可能是预料之外的。

Q & A

Q & A

段经理在早上，说到 call for action。在这里也希望得到大家的宝贵意见。