Flash存储优化

现状

每个累计因子会设定一个查询窗口，为了实现滚动窗口的计算效果，一个查询窗口会分为N个滑动窗口，每个滑动窗口包含其中一段时间范围。

计算时，根据事件的发生时间得出它属于哪个窗口，并将结果计入到这个窗口。

举例

累计因子：用户在1小时内点赞的次数

查询窗口：1小时

滑动窗口：5分钟一个窗口，共12个窗口

[0-5m),[5m-10m),[10m-15m),[15m-20m),[20m-25m),[25m-30m),

[30m-35m),[35m-40m),[40m-45m),[45m-50m),[50m-55m),[55m-60m)

写入：每个滑动窗口对应一个 key = 规则id:累计维度:维度值:滑动窗口时间戳，在滑动窗口的时间范围内的事件将被计入这个key。

查询：给定一个查询窗口，可以得出对应的N个滑动窗口（平均10-14个），批量从redkv查出后，再做一次汇总累计。

优化方案

和redkv团队讨论过，他们认为目前多key批量查询的方式对latency指标来说是最好的，因为是并发查询，而且flash缓存命中率很高。

redkv hash

https://wiki.xiaohongshu.com/pages/viewpage.action?pageId=84721064

改为使用hash存储，key为规则id:累计维度:维度值，将各个滑动窗口的时间戳和累计值作为hash field和value写入。

这样一次查询指定多field的hash查询（hmget）就能得到查到所有的窗口的值。

问题：

1. hmget在redkv 会转为batch get方式查询，相当于多field并发查底层rocksdb；（hscan是iterator）
2. 一个hash key它的field都落在同一个节点上，相当于单节点同时处理多个field请求，容易出现block请求的情况；
3. 批量查变为单次查，相当于一次请求返回的数据变大了，可能对latency产生影响；

redkv scan（推荐）

https://wiki.xiaohongshu.com/display/inf/RedKV+Scan

一次查询的多个滑动窗口的key存在连续的大小关系，通过将key改为带特定前缀，将同一规则同一累计值下的滑动窗口映射到固定的redkv节点，使用scan命令一次性从redkv查询出来。

问题：

1. 因为是连续存储，redkv底层会使用iterator方式查询，性能比hash hmget好
2. 批量查变为单次查，同hash问题3

聚合value

将原来多个滑动窗口的累计值合并存储到一个key下。

问题：

1. value体积最大，可能对latency产生影响，容易出现block请求的情况

举例

累计因子：用户在1小时内点赞的次数

查询窗口：1小时

滑动窗口：5分钟一个窗口，共12个窗口

[0-5m),[5m-10m),[10m-15m),[15m-20m),[20m-25m),[25m-30m),

[30m-35m),[35m-40m),[40m-45m),[45m-50m),[50m-55m),[55m-60m)

写入：每个累计对应一个 key = 规则id:累计维度:累计值，在滑动窗口的时间范围内的事件将被计入这个key。

查询：给定一个查询窗口，可以得出对应的N个滑动窗口（平均10-14个），批量从redkv查出后，再做一次汇总累计。

key

ruleId:groupByKey:groupByValue

value

[{"ts": 1678939009, "1"}, {"ts": 1678939009, "1"}, {"ts": 1678939009, "1"}, {"ts": 1678939009, "1"}]

资源优化

从目前redkv集群的表现来看，redkv团队评估下线一半redkv集群依然能打到我们latency要求，需要验证。