Facebook的时序数据库技术读后感

Facebook的时序数据库技术(上)

推荐阅读原文，本文只是个人易于理解。

基础概念：

Time Series

时间序列，在本项目当中就是每隔一段时间骑手的坐标，一连串的(time,data)

Data Point

在时间序列当中的点，在本项目当中就是指的某个(time,data)

Gorilla数据库

与普通数据库区别：大多数数据库需要确保数据不丢失，比如一些存储用户信息的数据库，而该为了保证数据不丢失，则会牺牲数据库的吞吐量。而对于某些情景而言，尤其是数据量较大的情况下，数据丢失与否并不是非常关键，因此更加考虑数据库的吞吐量。

Gorilla特性：

1.**基于内存的设计**。针对时序数据，设计了高效的内存数据结构，可支持高效的时序数据扫描，针对单个Time Series查询时可提供稳定的低时延保障。

2.**高效的压缩算法**。对DataPoint中的Timestamp与Point Value的数据特点提出了针对性的**压缩编码**机制，压缩效果显著，存储空间节省**90%**以上，单个Data Point平均只占用1.37 Bytes。该压缩机制有力支撑了基于内存的整体设计方案。

3.**先缓存后批量写日志**。流式写入到Gorilla中的数据，先缓存到一个64KB大小后，写入到一个Append-Only的日志文件中。如果在数据未写满64KB之前，进程故障会导致数据丢失，而Gorilla**容忍这种数据丢失场景**。以少量的数据丢失换取写入的高吞吐量。

4.**无状态易扩展**。基于Shared-Nothing的设计，可以轻易实现数据节点的横向扩展。

5.**跨Region双活集群**。利用部署在两个Region区的两个集群来提升服务的高可用性。

Gorilla中的关键技术：

1.数据组织形式

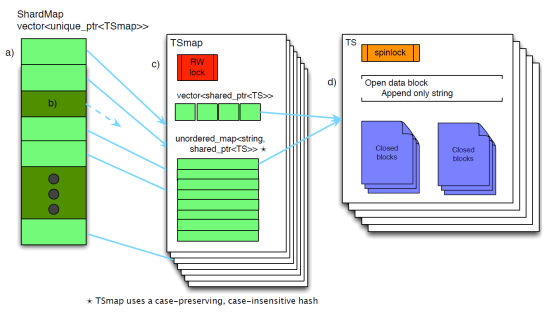
一般来说，有一个(time,data)需要存储，最先考虑到的可能是map结构，但在Gorilla中，使用了(string,time,data)形式保存数据，并且是通过一个非常小的元数据来索引string

Gorilla中的数据分片称为shard，数据节点称为Gorilla host，shard保存在host中。

在数据进行分配时，首先将string通过hash映射到shard中，然后shard通过路由找到host，而如果数据比较多，可以添加host，修改分片规则，将新的data送到新的host当中。

2.内存组织形式

Facebook调查发现26小时内的数据查询量能达到85%，所以，如果能将26小时的数据全都装入内存，则十分理想。



(没看懂)

3. 压缩算法

data point中的(time,data)数据有两个特点：

1. time数据基本上是固定时间间隔的，即等差数列

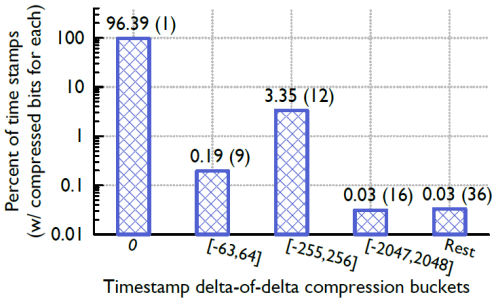
2. data数据变化比较缓慢，甚至有段时间是不会变化的

Timestamp压缩(Delta-Of-Delta)

- 在Block Header中，存放起始Timestamp T(-1)..一个Block对应2个小时的时间窗口。假设第一个Data Point的Timestamp为T(0)，那么，实际存放时，我们只需要存T(0)与T(-1)的差值。
- 对于接下来的Data Point的Timestamp T(N)，按如下算法计算Delta Of Delta值：
- *D = (T(N) - T(N-1)) - (T(N-1) - T(N-2))*

而后，按照D的值所处的区间范围，分别有5种不同情形的处理：

- 如果D为0，那么，存储一个bit ‘0’
- 如果D位于区间[-63, 64]，存储2个bits ’10’，后面跟着用7个bits表示的D值
- 如果D位于区间[-255, 256]，存储3个bits ‘110’，后面跟着9个bits表示的D值
- 如果D位于区间[-2047, 2048]，存储4个bits ‘1110’，后面跟着12个bits表示的D值
- 如果D位于其它区间则存储4个bits ‘1111’，后面跟着32个bits表示的D值



Point Value压缩（XOR）

第一个Value存储时不做任何压缩。

后面产生的每一个Value与前一个Value计算XOR值：

如果XOR值为0，即两个Value相同，那么存为’0’，只占用一个bit。

如果XOR为非0，首先计算XOR中位于**前端**的和**后端**的0的个数，即Leading Zeros与Trailing Zeros。

1) 第一个bit值存为’1’。

2.1) 如果Leading Zeros与Trailing Zeros与前一个XOR值相同，则第2个bit值存为’0’，而后，紧跟着去掉Leading Zeros与Trailing Zeros以后的**有效XOR值**部分。

2.2) 如果Leading Zeros与Trailing Zeros与前一个XOR值不同，则第2个bit值存为’1’，而后，紧跟着5个bits用来描述Leading Zeros的值，再用6个bits来描述**有效XOR值**的长度，最后再存储**有效XOR值**部分（这种情形下，至少产生了13个bits的冗余信息）