HBase Rowkey 设计指南

本文来自本人于2018年12月25日在
HBase生态+Spark社区钉钉大群直播，本群每周二下午18点-19点之间进行
HBase+Spark技术分享。加群地址：https://dwz.cn/Fvqv066s。本文 PPT 下载：关注
iteblog_hadoop 微信公众号，并回复 HBase_Rowkey 关键字获取。

为什么Rowkey这么重要

RowKey 到底是什么

如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

我们常说看一张 HBase 表设计的好不好，就看它的 RowKey 设计的好不好。可见 RowKey 在 HBase 中的地位。那么 RowKey 到底是什么？RowKey 的特点如下：

- 类似于 MySQL、Oracle中的主键，用于标示唯一的行；
- 完全是由用户指定的一串不重复的字符串；
- HBase 中的数据永远是根据 Rowkey 的字典排序来排序的。

RowKey的作用

- 读写数据时通过 RowKey 找到对应的 Region；
- MemStore 中的数据按 RowKey 字典顺序排序；
- HFile 中的数据按 RowKey 字典顺序排序。

Rowkey对查询的影响

如果我们的 RowKey 设计为 uid+phone+name，那么这种设计可以很好的支持以下的场景：

- uid = 111 AND phone = 123 AND name = iteblog
- uid = 111 AND phone = 123
- uid = 111 AND phone = 12?
- uid = 111

难以支持的场景：

- phone = 123 AND name = iteblog
- phone = 123
- name = iteblog

Rowkey对Region划分影响

HBase 表的数据是按照 Rowkey 来分散到不同 Region，不合理的 Rowkey
设计会导致热点问题。热点问题是大量的 Client
直接访问集群的一个或极少数个节点，而集群中的其他节点却处于相对空闲状态。

		personal			office	
	RowKey	name	city	phone	tel	address
Region1	00001	张三	北京	13111111111	010-1111111	帝都大厦-18F-01
	000011	李四	上海		010-4444444	帝都大厦-19F-02
	00002	王五	武汉	18655555555	010-3333333	帝都大厦-18F-02
	00003	赵六		15166666666		帝都大厦-18F-03
	00004	孙七	北京		010-7777777	帝都大厦-18F-04
Region2	00005	周八	深圳	15388888888		帝都大厦-18F-05
Region3	00006	吴九	杭州		010-9999999	帝都大厦-18F-06
	00007	郑十	武汉	13599999999	010-5555555	帝都大厦-18F-07

如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

如上图，Region1 上的数据是 Region 2 的5倍，这样会导致 Region1
的访问频率比较高，进而影响这个 Region 所在机器的其他 Region。

RowKey设计技巧

我们如何避免上面说到的热点问题呢？这就是这章节谈到的三种方法。

避免热点的方法 - Salting

这里的加盐不是密码学中的加盐，而是在rowkey 的前面增加随机数。具体就是给 rowkey 分配一个随机前缀
以使得它和之前排序不同。分配的前缀种类数量应该和你想使数据分散到不同的 region 的数量一致。 如果你有一些 热点 rowkey 反复出现在其他分布均匀的 rwokey 中，加盐是很有用的。考虑下面的例子：它将写请求分散到多个 RegionServers，但是对读造成了一些负面影响。

假如你有下列 rowkey，你表中每一个 region 对应字母表中每一个字母。 以 'a' 开头是同一个region, 'b'开头的是同一个region。在表中，所有以 'f'开头的都在同一个 region，它们的 rowkey 像下面这样：

foo0001
foo0002
foo0003
foo0004

现在，假如你需要将上面这个 region 分散到 4个 region。你可以用4个不同的盐：'a', 'b', 'c', 'd'.在这个方案下，每一个字母前缀都会在不同的 region 中。加盐之后，你有了下面的 rowkey:

a-foo0003
b-foo0001
c-foo0004
d-foo0002

所以，你可以向4个不同的 region 写。理论上说，如果这四个 Region 存放在不同的机器上，经过加盐之后你将拥有之前4倍的吞吐量。

现在，如果再增加一行，它将随机分配a,b,c,d中的一个作为前缀，并以一个现有行作为尾部结束：

a-foo0003
b-foo0001
c-foo0003
c-foo0004
d-foo0002

因为分配是随机的，所以如果你想要以字典序取回数据，你需要做更多工作。加盐这种方式增加了写时的吞吐量，但是当读时有了额外代价。

避免热点的方法 - Hashing

Hashing 的原理是计算 RowKey 的 hash 值，然后取 hash 的部分字符串和原来的 RowKey 进行拼接。这里说的 hash 包含 MD5、sha1、sha256或sha512等算法。比如我们有如下的 RowKey：

foo0001
foo0002
foo0003
foo0004

我们使用 md5 计算这些 RowKey 的 hash 值，然后取前 6 位和原来的 RowKey 拼接得到新的 RowKey：

95f18cfoo0001
6ccc20foo0002
b61d00foo0003
1a7475foo0004

优缺点：可以一定程度打散整个数据集，但是不利于 Scan；比如我们使用 md5 算法，来计算Rowkey的md5值，然后截取前几位的字符串。subString(MD5(设备ID), 0, x) + 设备ID，其中x一般取5或6。

避免热点的方法 - Reversing

Reversing 的原理是反转一段固定长度或者全部的键。比如我们有以下 URL ，并作为 RowKey：

flink.iteblog.com
www.iteblog.com
carbondata.iteblog.com
def.iteblog.com

这些 URL 其实属于同一个域名，但是由于前面不一样，导致数据不在一起存放。我们可以对其进

行反转，如下：

moc.golbeti.knilf
moc.golbeti.www
moc.golbeti.atadnobrac
moc.golbeti.fed

经过这个之后，这些 URL 的数据就可以放一起了。

RowKey的长度

RowKey 可以是任意的字符串，最大长度64KB（因为 Rowlength
占2字节）。建议越短越好，原因如下：

- 数据的持久化文件HFile中是按照KeyValue存储的，如果rowkey过长，比如超过100字节，1000w行数据，光rowkey就要占用100*1000w=10亿个字节，将近1G数据，这样会极大影响HFile的存储效率；
- MemStore将缓存部分数据到内存，如果rowkey字段过长，内存的有效利用率就会降低，系统不能缓存更多的数据，这样会降低检索效率；
- 目前操作系统都是64位系统，内存8字节对齐，控制在16个字节，8字节的整数倍利用了操作系统的最佳特性。

RowKey 设计案例剖析

交易类表 Rowkey 设计

- 查询某个卖家某段时间内的交易记录
  sellerId + timestamp + orderId
- 查询某个买家某段时间内的交易记录
  buyerId + timestamp + orderId
- 根据订单号查询
  orderNo
- 如果某个商家卖了很多商品，可以如下设计 Rowkey 实现快速搜索
  salt + sellerId + timestamp 其中，salt 是随机数。
  可以支持的场景：
    - 全表 Scan
    - 按照 sellerId 查询
    - 按照 sellerId + timestamp 查询

金融风控 Rowkey 设计

查询某个用户的用户画像数据

- prefix + uid
- prefix + idcard
- prefix + tele

其中 prefix = substr(md5(uid),0 ,x)， x 取 5-6。uid、idcard以及 tele 分别表示用户唯一标识符、身份证、手机号码。

车联网 Rowkey 设计

- 查询某辆车在某个时间范围的交易记录
  carId + timestamp
- 某批次的车太多，造成热点
  prefix + carId + timestamp 其中 prefix = substr(md5(uid),0 ,x)

查询最近的数据

查询用户最新的操作记录或者查询用户某段时间的操作记录，RowKey 设计如下：
uid + Long.Max_Value - timestamp
支持的场景

- 查询用户最新的操作记录
  Scan [uid] startRow [uid][000000000000] stopRow [uid][Long.Max_Value - timestamp]
- 查询用户某段时间的操作记录
  Scan [uid] startRow [uid][Long.Max_Value – startTime] stopRow [uid][Long.Max_Value - endTime]

OpenTSDB 的 Rowkey 设计

参见 《OpenTSDB 底层 HBase 的 Rowkey 是如何设计的》

如果 RowKey 无法满足我们的需求，可以尝试二级索引。Phoenix、Solr 以及 ElasticSearch 都可以用于构建二级索引。