OpenTSDB 底层 HBase 的 Rowkey 是如何设计的

OpenTSDB 是基于 HBase 的可扩展、开源时间序列数据库(Time Series Database)，可以用于存储监控数据、物联网传感器、金融K线等带有时间的数据。它的特点是能够提供最高毫秒级精度的时间序列数据存储，能够长久保存原始数据并且不失精度。它拥有很强的数据写入能力，支持大并发的数据写入，并且拥有可无限水平扩展的存储容量。目前，阿里云 HBase 产品是直接支持 OpenTSDB 组件的。

OpenTSDB 拥有如此的强大的读写和近乎无限的存储能力源自于基于 HBase 的架构设计，我们甚至可以说 OpenTSDB 就是 HBase 的一个应用。熟悉 HBase 的同学肯定知道，要看 HBase 的表设计的好不好，关键是看其 Rowkey 设计的好不好，HBase 的 Rowkey 设计会考虑到实际的查询场景。所以 OpenTSDB 拥有如此强大的时序功能，其中的一大原因就是 Rowkey 的合理设计。本文将介绍 OpenTSDB 的Rowkey 是如何设计的。

OpenTSDB 基本概念

在介绍 OpenTSDB 系统如何设计 Rowkey 之前，我们先来了解 OpenTSDB 的一些基本概念。（因为本文侧重于介绍 HBase 的 Rowkey 设计，所以关于 OpenTSDB 的其他一些知识本文并不会涉及，如果你对这部分知识感兴趣，请自行去网上搜索相关文章。）

我们往 OpenTSDB 里面写入一条时序数据，至少包含以下几个数据：

- 指标名字：这个就是我们监控的指标，比如 sys.cpu.user；
- 时间戳：监控数据产生的时间；
- 值：Long 或者 Double 类型的数据，这个是监控指标在某个时间的具体值；
- 标签：包括标签名字（tagk）和标签值（tagv），比如 tagk1=tagv1，主要用于描述数据属性，每条时序数据必须包含一组和多组的标签数据。目前 OpenTSDB 最多支持8组标签。

所以如果我们使用终端往 OpenTSDB 写入时序数据，格式如下：

put <metric> <timestamp> <value> <tagk1=tagv1[tagk2=tagv2 ...tagkN=tagvN]>

比如
put sys.cpu.user 1541946115 42.5 host=iteblog cpu=0

OpenTSDB 的 Rowkey 设计

上面我们已经简单了解了 OpenTSDB

每条时序数据所包含的要素。基于这些时序数据，OpenTSDB
为我们提供的查询功能其实很简单：
指定指标名称和时间范围，并且给定一个或多个标签名称和标签的值作为过滤条件，以此查询符合条件的数据。

Rowkey 设计版本一

OpenTSDB 为我们提供的查询业务场景已经有了，我们可以很快设计出 HBase 的 Rowkey：
metric + timestamp + tagk1 + tagv1 + tagk2 + tagv2 + ... + tagkn + tagvn
注意，实际存储的时候 + 并不会写入到磁盘，这里只是为了说明方便，人为加了这个符号。
比如如果我们往 OpenTSDB 插入下面的数据：

```
put sys.cpu.user 1541946115 42.5 host=iteblog cpu=0
```

那么按照上面的思路 Rowkey 应该为：

```
sys.cpu.user+1541946115+host+iteblog+cpu+0
```

那如果这个指标有很多监控数据，其存储在 HBase 的 key-value 如下：

Key	Value
sys.cpu.user+1541946115+host+iteblog+cpu+0	42.5
sys.cpu.user+1541946125+host+iteblog+cpu+0	39.1
sys.cpu.user+1541946135+host+iteblog+cpu+0	41.4
sys.cpu.user+1541946145+host+iteblog+cpu+0	40.0
sys.cpu.user+1541946135+host+iteblog+cpu+1	53.2
sys.cpu.user+1541946145+host+iteblog+cpu+1	50.9
sys.cpu.user+1541946155+host+iteblog+cpu+1	48.2

Rowkey 设计版本二

上面表格记录着指标名 sys.cpu.user 标签为 host=iteblog cpu=0 和 标签为 host=iteblog cpu=1 每隔十秒的监控数据。有些同学可能已经看出来了，如果我们按照这样的方式去设计 HBase 表的 Rowkey，虽然可以满足我们的查询需求，但是这种存储数据的方式导致 Key
大量的重复存储，这样会导致数据的急剧增加，所以 OpenTSDB 并没有这样存储的。在 OpenTSDB 里面，会对每个指标名、标签以及标签值进行编码，每个指标的编码都不一样；同理，每个标签的编码也不一样，但是标签和指标名称可以编码一样，不同类型之间的编码互不影响。所以编码后的数据如下：

sys.cpu.user => ₩x00₩x00₩x01
host => ₩x00₩x00₩x01
iteblog => ₩x00₩x00₩x01
cpu => ₩x00₩x00₩x02
0 => ₩x00₩x00₩x02
1 => ₩x00₩x00₩x03

在上面，OpenTSDB 默认使用三个字节来编码指标名称，三个字节编码标签名称以及标签值。经过这样的编码之后，OpenTSDB 的 Rowkey 就变成了下面的形式：

sys.cpu.user+1541946115+host+iteblog+cpu+0
变成
₩x00₩x00₩x01+1541946115+₩x00₩x00₩x01+₩x00₩x00₩x01+₩x00₩x00₩x02+₩x00₩x00₩x02

所以上表的数据就变成下面的了：

Key	Value
000001+1541946115+000001+000001+000002+000002	42.5
000001+1541946125+000001+000001+000002+000002	39.1
000001+1541946135+000001+000001+000002+000002	41.4
000001+1541946145+000001+000001+000002+000002	40.0
000001+1541946135+000001+000001+000002+000003	53.2
000001+1541946145+000001+000001+000002+000003	50.9
000001+1541946155+000001+000001+000002+000003	48.2

这样我们可以节省一些存储空间（不要看这张表好像比上面的表要长了，这里其实是用十六进制表示的，每个₩x00占用一个字节，整个指标名称默认只占用三个字节，如果用字符串表示是不止三个字节的）。

Rowkey 设计版本三

但是细心的同学肯定发现了，上表中同一个指标每隔十秒发送一条监控数据，但是每条监控数据

就只是当前指标的监控值，如上表的42.5、39.1、41.4、40.0。而每次发送的数据都在 HBase 里面存储一行，这样会导致重复存储大量相同的指标名、标签名、标签值等数据。我们仔细观察可以发现，Rowkey
组成中同一个指标的监控数据除了的时间不一样，其他都是一样的！基于这个特点，OpenTSDB 对 Rowkey 进行了进一步的优化，思想为：将 Rowkey
中时间戳由原来的秒级别或毫秒级别统一转换成小时级别的，多余的秒数据或者毫秒数据作为 HBase 的列名称。可能大家没有理解这句话的含义，下面我们来具体介绍这个实现。

1541946115 时间戳转换成时间为 2018-11-11 22:21:55，其对应的整点小时为 2018-11-11 22:00:00，这个转换成时间戳是 1541944800。1541946115 相对于 1541944800
多余出来的秒数为 1315，在 HBase 里面，1315 就作为当前指标对应值的列名。经过这样的优化之后，同一小时的监控数据都放在一行的不同列里面，所以上面的表格就变成下面的了：

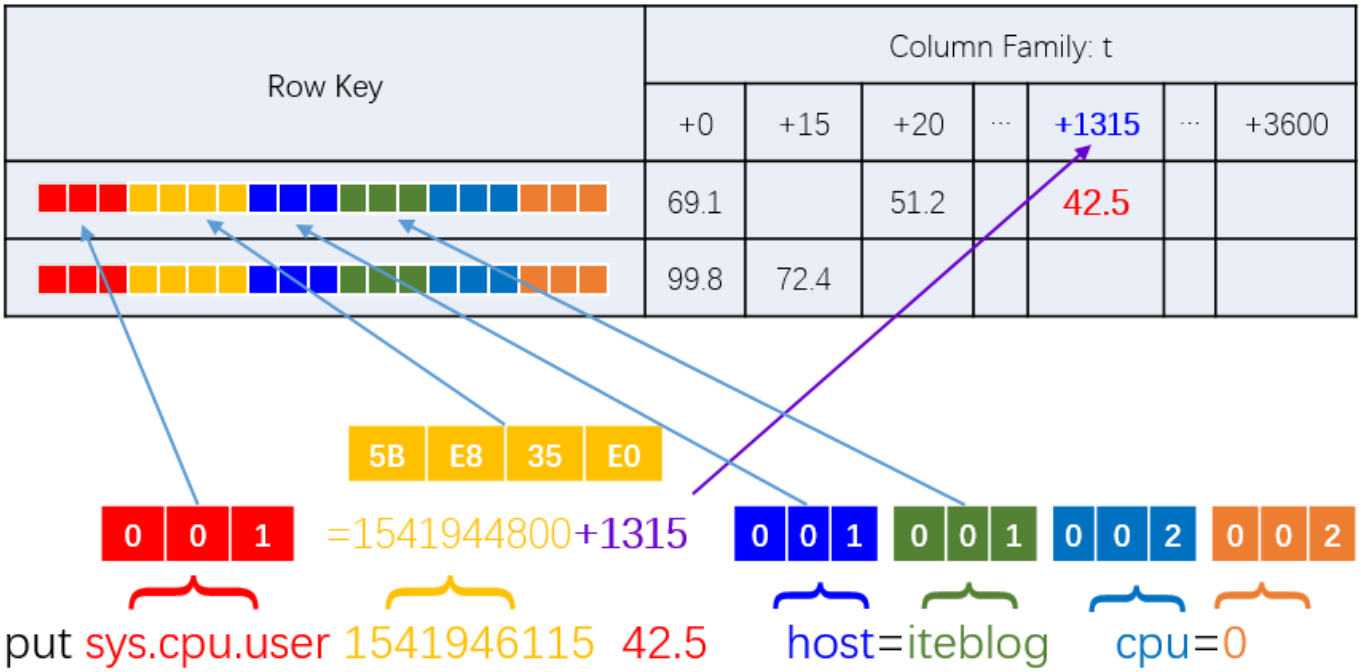
Key	1315	1325	1335	1345	1355
001+1541944800+001+001+002+002	42.5	39.1	41.4	40.0	
001+1541944800+001+001+002+003			53.2	50.9	48.2

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

注意：

- 第三张表中为了展示方便，我将 000001+1541944800+000001+000001+000002+000003 简写为 001+1541944800+001+001+002+003；
- 上面几张表中的 Rowkey 部分我这里都是使用时间戳的形式显示的，只是为了查看方便，在实际存储中时间戳其实是以二进制形式存储的，比如 1541944800 的十六进制表示为 5BE835E0；所以上面表格中 Rowkey 为 001+1541944800+001+001+002+003 在 HBase 实际存储为（十六进制表示）0000015BE835E000000100000100000200003；
- 第三张表中的列名称在实际存储中除了包含相对于 Rowkey
的秒数或者毫秒数，其实还包含了当前列值的数据类型，数据长度等标识。

如果说用一张图表示上面的过程，可以如下所示。



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

如果想通过例子进一步了解 Rowkey 到底是如何组织以及列名称是如何组成的，可以进一步阅读[通过例子剖析 OpenTSDB 的 Rowkey 及列名设计](#)。