<http://www.infoq.com/cn/articles/condense-in-sequential-databases?utm_source=articles_about_imdbs&utm_medium=link&utm_campaign=imdbs>

物联网邻域近期如火如荼，互联网和传统公司争相布局物联网。作为物联网邻域数据存储的首选时序数据库也越来越多进入人们的视野，而早在 2016 年 7 月，百度云在其天工物联网平台上发布了国内首个多租户的分布式时序数据库产品 TSDB，成为支持其发展制造，交通，能源，智慧城市等产业领域的核心产品，同时也成为百度战略发展产业物联网的标志性事件。

前面我们有讲过时序数据库的存储篇（点击可查看）：[百度无人车和天工物联网都使用了时序数据库，但是你有多了解时序数据库？](http://mp.weixin.qq.com/s?__biz=MzA5NzkxMzg1Nw==&mid=2653162430&idx=1&sn=0afcdbbcf3aec3ed39880eee695e8e08&chksm=8b4936d0bc3ebfc68c8cd3f36695d6f3c6d87f74e5ad153cc98b0956aae2212092979f61a61c&scene=21#wechat_redirect)

和时序数据库的预处理篇（点击可查看）：[时序数据库如何支持秒级上亿数据的查询分组和聚合运算](http://mp.weixin.qq.com/s?__biz=MzA5NzkxMzg1Nw==&mid=2653162499&idx=1&sn=ca43c74ef541bd16a1d2450db6479c63&chksm=8b49316dbc3eb87b9b1b8049429c820ed557fd2b735c5a486350db3ef0d0c7735f8171ba28a9&scene=21#wechat_redirect)

压缩对于时序数据库是至关重要的。因为时序数据库面对的物联网场景每天都会产生上亿条数据。众所周知，在大数据时代的今天数据的重要性是不言而喻的，数据就是公司的未来。但如果无法对这些时序数据进行很好的管理和压缩，那将给客户带来非常高的成本压力。

如前文提到的，工业物联网环境监控方向的客户，一年产生 1P 的数据，如果每台服务器 10T 的硬盘，那么总共需要 100 多台。按照每台服务器 3 万来算，一年就需要 300 万的支出，这还不包括维护人员的成本。

压缩是个非常大的话题，本文希望能够先从大的宏观角度给出一个轮廓，讲述压缩的本质，压缩的可计算性问题。再从时序数据压缩这一个垂直领域，给出无损压缩和有损压缩各一个例子进行说明，希望能够抛砖引玉。

## 压缩的故事

先来讲个有关压缩的故事，外星人拜访地球，看中了大英百科全书，想要把这套书带回去。但这套书太大，飞船放不下。于是外星人根据飞船的长度，在飞船上画了一个点。这样外星人心满意足的返回了自己的星球，因为这个点就存储了整个大英百科全书。

这个并不是很严谨的故事，却道出了压缩的本质：用计算时间换取存储空间。外星人在飞船上画的点非常有技术含量，可以说是黑科技，代表一个位数非常长的不循环小数。而这串数字正代表了整个大英百科全书的内容。

## 压缩的两个问题

再来回答两个宏观的问题，帮助我们认识在压缩这件事上哪些是我们能做的，哪些是不能做的。

第一个问题：是否存在一个通用的压缩算法（Universal Compression），也就是说某个压缩算法能够压缩任意的数据。答案是否定的，并不存在这样的通用压缩算法。

用反证法可以做个快速的证明。假设存在通用的压缩算法，也就是说有个压缩算法，对于长度为 n 的字符串，总能压缩到长度小于 n 的字符串。总共有 2^n 个长度为 n 的不同字符串；但却只有 2^n-1 个长度小于 n 的字符串。那么必定存在两个长度为 n 的字符串 A,B，经过压缩得到同一个字符串。这样解压缩算法没有办法正确的解压。所以假设错误，并不存在通用的压缩算法。

第二个问题：是否能写出一个函数，输入字符串，可以得到这个字符串最短表示的长度。答案也是否定的，也就是说我们无法证明某个算法是最好的算法。柯尔莫哥洛夫复杂性的不可计算性解释的就是这个问题。用的也是反证法，有兴趣的朋友可以自行百度了解（注 1）。

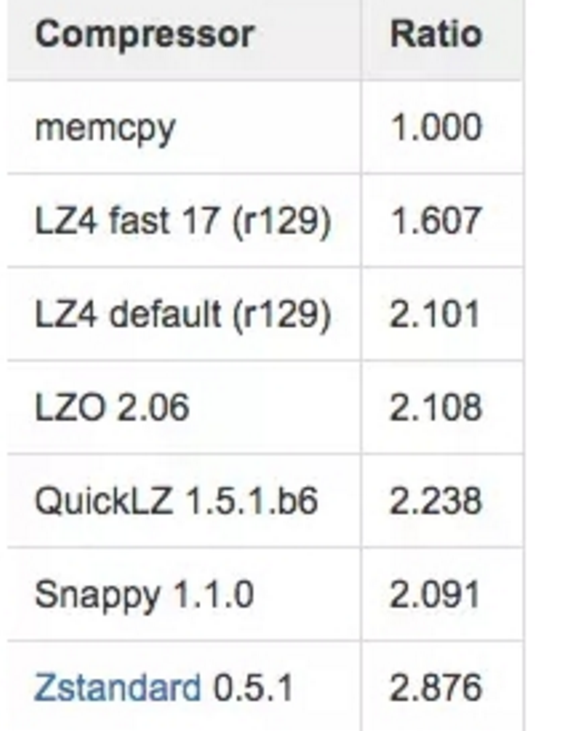
这两个问题的答案，告诉我们三件事情，

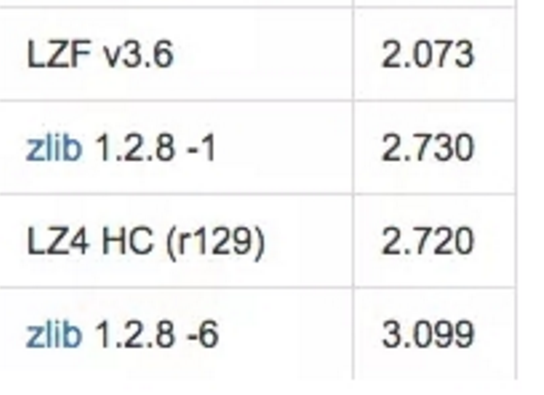
1. 压缩算法的选择需要具体情况具体分析，不可压缩的字符串总是存在。
2. 不要妄图获得最好的压缩算法，它是不可计算的。因为总有你想不到的压缩算法存在。举个例子，[一百万个 0 的字符串，以“foo”作为 key，经过 AES 加密算法的 CBC 模式得到的字符串]。这串字符串看起来完全是随机的，不可压缩的。但我却用 43 个中文（中括号之间的内容）就表示了出来。
3. 压缩是件很难很有技术含量的事情，需要不断的挖掘，才能将他做到更好。

## 时序数据压缩

针对不同的数据，会有不同的压缩，大致压缩的对象可以分为文档、音频、视频等。如果直接采用文档的压缩算法用于时序数据，效果并不理想。下图是一些常用的压缩算法的 benchmark，可以看到压缩率那一栏最高也只能够达到 3 左右的压缩率（压缩率 = 原始数据大小 / 压缩后的数据大小）。更多压缩算法可以查看注 2。

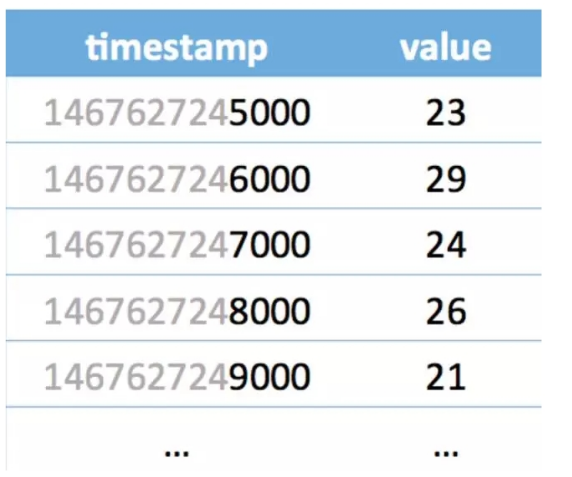
如果要得到更好的压缩率，我们需要采取更加适合时序数据的压缩算法。时序数据的压缩可以分为无损压缩和有损压缩。





## 无损压缩

无损压缩是说被压缩的数据和解压后的数据完全一样，不存在精度的损失。对数据的压缩说到底是对数据规律性的总结。时序数据的规律可以总结为两点：1、timestamp 稳定递增、2、数值有规律性，变化稳定。下面来举个例子。



上图是一组时序数据，如果我们一行一行的看感觉压缩有点困难，但如果我们一列一列的看，压缩方案就呼之欲出了。

先看 timestamp 那一列是等差递增数列，可以用 [1467627245000,1000,4] 来表示。1467627245000 代表了第一个时间，1000 代表后一个时间比前一个时间的大 1000，4 代表了这样的规律出现了 4 次。如果一共有 100 个这样规律的 timestamp，那就意味着，我们用 3 个 Long 型就可以表示出来。timestamp 压缩率高达 33。

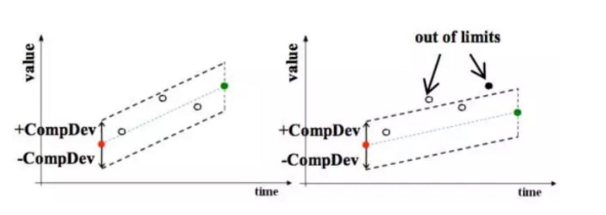
再进一步观察看 value 那一列，如果取差值，可以得到（6，-5，2，-5），全部都加 5 得到（11，0，7，0），这些数值都可以用 4bit 来表示。也就是用 [23,5,4,0xb0700000] 来表示（23，22，24，25，24）。其中的 4 代表后续一共有 4 个数。如果这样的规律一直维持到 100 个 Int 的 value，就可以用 16 个 Int 来代表，压缩率高达 6.3。

具体的情景会复杂很多，在此只是简单举个例子。InfluxDB 无损压缩算法在其页面上有完整的阐述（注 3），可以配合开源源码进行更加深入的理解。针对于浮点数类型，Facebook 在 Gorilla 论文中（注 4）提到的非常高效的无损压缩算法，已经有很多文章进行分析。InfluxDB 对于浮点型也采用这个算法。

## 有损压缩

有损压缩的意思是说解压后的数据和被压缩的数据在精度上有损失，主要针对于浮点数。通常都会设置一个压缩精度，控制精度损失。时序数据的有损压缩的思路是拟合。也就是用一条线尽可能的匹配到这些点，可以是直线，也可以是曲线。

最有名的时序数据有损压缩是 SOIsoft 公司的 SDA 算法，中文称为旋转门压缩算法。



在上图中，红色的点是上一个记录的点，空心的点是被丢掉的点，绿色的点是当前的点，黑色的点是当前要记录的点。

可以看到图左边，当前点和上一个记录点以及压缩精度的偏差值形成的矩形可以包含中间的点，所以这些点都是可以丢掉的。

再看图右边，当前点和上一个记录点形成的矩形无法包含中间的点，所以把上一个点记录下来。如此进行下去，可以看到，大部分的数据点都会被丢掉。查询的时候需要根据记录的点把丢掉的点在插值找回来。

有损压缩除了可以大幅减少存储成本。如果结合设备端的能力，甚至可以减少数据的写入，降低网络带宽。

## 总结

虽然判断压缩算法最优是不可计算的，但是设计好的压缩算法仍然是可计算的问题。可以看到，前面提到的时序数据的无损压缩有损压缩算法都会基于时序数据的特征采取方案，达到更好的压缩率。现在 deep learning 非常的火，让人很好奇它是不是可以给数据压缩带来新的方案。

注 1:[请点击](https://condense-in-sequential-databases/zh.wikipedia.org/zh-cn/%E6%9F%AF%E6%B0%8F%E5%A4%8D%E6%9D%82%E6%80%A7)

注 2：[请点击](https://github.com/inikep/lzbench/blob/v1.7.1/lzbench171_sorted.md)

注 3：[请点击](https://docs.influxdata.com/influxdb/v1.0/concepts/storage_engine/#compression)

注 4：[请点击](http://www.vldb.org/pvldb/vol8/p1816-teller.pdf)