

# 基于 influxDB 的工业时序数据库引擎设计

徐化岩<sup>1</sup> 初彦龙<sup>2</sup>

<sup>1</sup>(冶金自动化研究设计院混合流程工业自动化系统及装备技术国家重点实验室 北京 100071)

<sup>2</sup>(辽宁警察学院治安管理学系 辽宁 大连 116036)

**摘要** 工业时序数据具有测点多、采样频率快、读取性能要求高等特点。influxDB 提出时间结构合并树 TSM,解决了数据的读写性能优化问题,并针对整数、浮点数、布尔、字符串、时间五种数据类型采用不同的压缩算法。它在读写性能、存储空间占用方面取得了较好的效果,在开源时序数据库软件中排名第一。针对 influxDB 元数据结构相对于工业时序数据过于复杂的问题,提出简化后的 TSM 文件结构,开发工业时序数据库的引擎,并进行了读写性能测试。结果显示,该引擎一次 5 万点整型数据写入平均耗时约 310 ms,读取 1 000 点共计 100 万条数据耗时约 626 ms,并且具有很大的性能提升潜力。

**关键词** 时序数据库 influxDB TSM 压缩 索引

中图分类号 TP311.13 文献标识码 A DOI: 10.3969/j.issn.1000-386x.2019.09.006

## DESIGN OF INDUSTRIAL TIME SERIES DATABASE ENGINE BASED ON INFLUXDB

Xu Huayan<sup>1</sup> Chu Yanlong<sup>2</sup>

<sup>1</sup>(State Key Laboratory of Hybrid Process Industry Automation Systems and Equipment Technology, Automation Research and Design Institute of Metallurgical Industry, Beijing 100071, China)

<sup>2</sup>(Department of Public Security Management, Liaoning Police College, Dalian 116036, Liaoning, China)

**Abstract** Industrial time series data have many characteristics, such as many measuring points, fast sampling frequency and high requirement of reading performance. The TSM proposed by influxDB solves the problem of data read-write performance optimization. Different compression algorithms are adopted for five data types: integer, floating point, Boolean, string and time. It has achieved good results in reading and writing performance and storage space occupancy, ranking first in open source sequential database software. Aiming at the problem that influxDB metadata structure is too complex compared with industrial time series data, this paper proposed a simplified TSM file structure, developed an engine of industrial time series database, and tested its read-write performance. The results show that the engine takes about 310 ms to write 50 000 integer data at a time, and about 626 ms to read 1 million data at 1 000 points, and has great potential for performance improvement.

**Keywords** Time series database InfluxDB TSM Compression Index

## 0 引言

时序数据即时间序列数据,指按照时间先后顺序变化、带时间标签的数据。时序数据主要由电力、化工、冶金等各类型实时监测、检查与分析设备所采集、产生的数据,这些数据的典型特点是:(1) 采样频率

快,每秒甚至几十毫秒采样一次;(2) 测点多信息量大,一个千万吨级钢铁联合企业共有约 50 万个在线监测点,如果每秒采集后不压缩存储,一天需要 800 GB 存储空间;(3) 数据按照时间顺序产生,一经产生不再变化;(4) 一次读取的数据量大,绘制一条 24 小时秒级曲线需要读取 86 400 条数据;(5) 没有关系数据的并发写入/更改情况,没有读写共享冲突问题;(6) 单

收稿日期:2019-01-23。国家重点研发计划项目(2017YFB0304102)。徐化岩,高工,主研领域:钢铁企业信息化。初彦龙,副教授。

点数据价值小,对数据一致性要求没有业务数据严格。

目前,工业领域普遍的做法是使用商业实时数据库软件,如 OSIsoft PI、Wonderware Insql、GE IHistorian 等,作为时序数据库的存储和读取工具。随着大数据技术的发展,近年来也有一些基于 HBase、MongoDB<sup>[1]</sup> 等大数据平台的研究。

2013 年,Errplane 公司将 influxDB<sup>[2]</sup> 开源,经过不断改进现已在开源时序数据库中排名第一<sup>[3]</sup>。influxDB 参考 O'Neil P 等<sup>[4]</sup> 提出应用于 BigTable<sup>[5]</sup> 的日志结构合并树(Log Structured Merge Tree, LSM),提出了时间结构合并树(Time Structured Merge Tree, TSM),针对时序数据的读写分别进行了优化设计。在写入时,将数据追加写入到日志文件(Write Ahead Log, WAL)中,并在内存中进行缓存,当内存缓存达到一定大小时创建新的日志文件,并启动压缩线程,将数据按照读取优化的方式压缩成 TSM 数据文件。根据官方提供的性能<sup>[2]</sup>,influxDB 与其他几个时序数据库的性能比较如表 1 所示。

表 1 开源时序数据库性能对比表

序号	对比数据库	写入速度	读取速度	存储空间占用
1	Cassandra	快 4.5 倍	快 45 倍	少 2.1 倍
2	Elasticsearch	快 6.1 倍	快 8.2 倍	少 2.5 倍
3	MongoDB	快 2.4 倍	快 5.7 倍	少 20 倍
4	OpenTSDB	快 5 倍	快 3.65 ~ 4 倍	少 16.5 倍

influxDB 起源于对服务器运行情况进行监控这一特定领域,设计了较为复杂的元数据模型以便能够保存除测点、时间、取值之外的附加属性信息。数据存储时的键为含有测点名、属性类型、取值类型的字符串,不仅占用空间多,而且索引时需要根据字符串长度计算数据位置,增加了检索计算量,这一复杂设计对于工业时序数据而言是多余的。本文参考 influxDB 的引擎代码,改进了 TSM 文件结构,实现了工业时序数据库引擎,并进行了性能测试。

## 1 时序数据库引擎结构

引擎自顶而下包括数据库实例、存储策略、存储分片几个层次,如图 1 所示。每个数据库实例包含一个或更多存储策略,用于定义存储时长,存储策略包含多个存储分片,每个存储分片负责一段时间的数据,超过时长的过期数据会以存储分片为单位进行整体删除,以提高删除效率。每个存储分片里包含一个内存缓存区、一个或更多日志文件、多个数据文件,以及用于处

理写入、压缩、读取的线程。

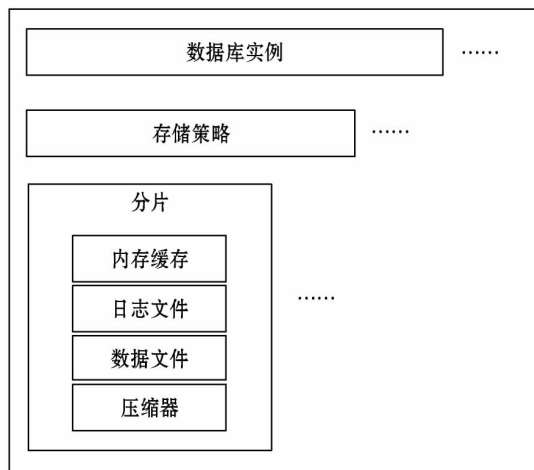


图 1 时序数据库引擎结构示意图

在写入时,系统根据数据点所属数据库实例名找到数据库实例,由数据库实例找到数据点所属数据存储策略,然后由存储策略根据数据的时间范围找到命中的存储片区,如果片区不存在则创建片区,并将数据分发给片区执行写入。存储片区负责将数据同时写入到内存缓存与日志文件中,日志文件定时压缩为数据文件,数据文件再逐步压缩为更大的数据文件,直到数据文件大小达到上限不再压缩。写入流程如图 2 所示。

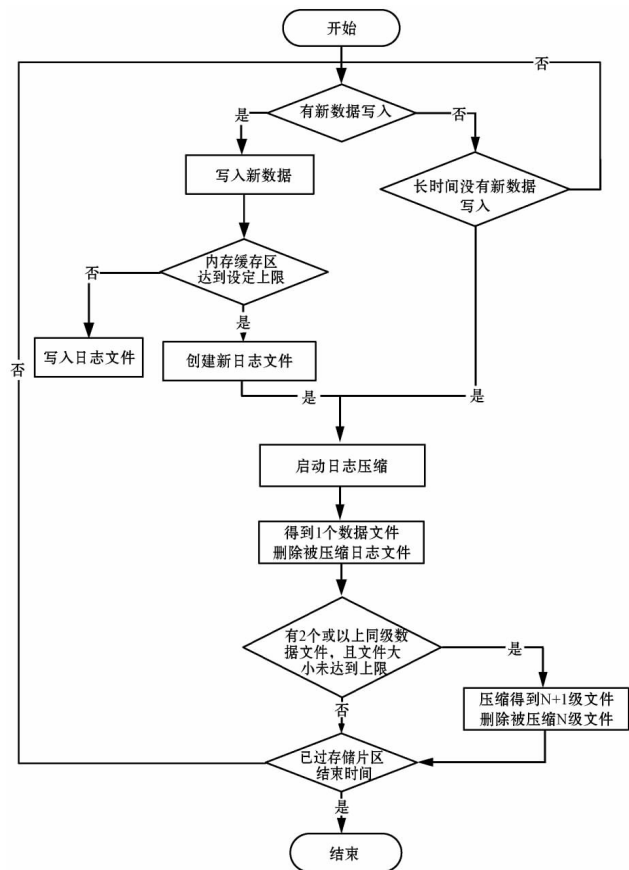


图 2 数据写入及文件压缩流程图

在读取时,系统首先找到所有命中的存储片区,由片区判断是从其内存缓存中读取还是从其数据文件中

读取。如果是从数据文件中读取,利用文件中的索引快速定位数据位置,读取后由系统将所有命中片区返回的结果按照时间先后顺序合并,返回最终查询结果。读取流程如图3所示。

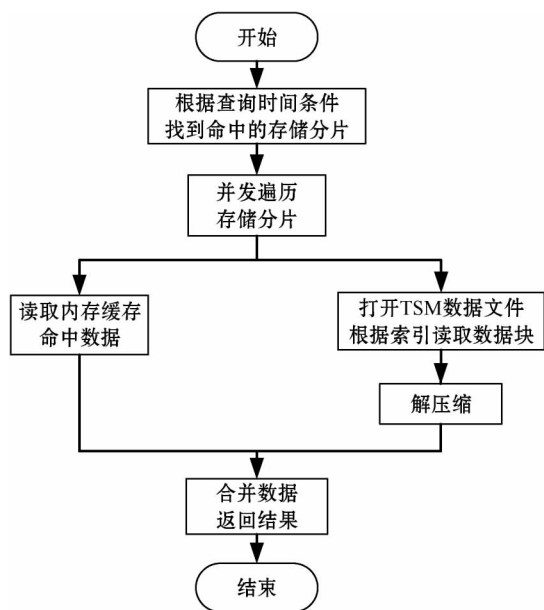


图3 数据读取流程图

## 2 TSM 文件结构设计

TSM 采用内存缓存和 WAL 日志文件来保证高效写入,内存缓存目的是代替日志文件提供对临近数据的高效读取,日志文件的目的是在数据库重启后能够重建内存缓存,日志文件不用于读取数据。由于磁盘的顺序写入速度快而随机写入速度慢(寻道和旋转延迟),而时序数据的特点是大量数据实时采集、实时写入,为了提高写入效率将一批数据(一般为5 000到10 000点)按照数据类型、点位编号、点数、时间、值、…、数据类型等顺序编码为字节流,如图4所示,再利用 snappy 算法<sup>[6]</sup>压缩后写入 WAL 日志文件。同时,将写入日志文件的数据以点位编号、时间、值的结构存入内存缓存区,内存缓存区与日志文件保持同步。日志文件大小固定,当达到规定大小时自动生成一个新的日志文件。

数据类型	点位编号	点数	时间	值	...	数据类型	...
1字节	8字节	4字节	8字节	N字节	...	1字节	...

图4 WAL 日志文件结构图

TSM 数据文件中包含5字节文件头、多个数据块、一个索引块和8字节文件尾,如图5所示。文件头存放数据库引擎的幻数和版本号;数据块区由若干组4字节校验和与N字节数据块组成;校验和是数据块的32位循环冗余校验码;数据块为数据点的一组按时间排序后的数据经过压缩后得到的字节数组;索引块由

点位编号、数据类型、数据块数量、数据块索引1、数据块索引2等构成,每个数据块索引包含数据块起始时间、数据块结束时间、数据块在文件中的起始位置、数据块字节数构成;文件尾8字节用于保存索引块在TSM数据文件中的起始字节位置。

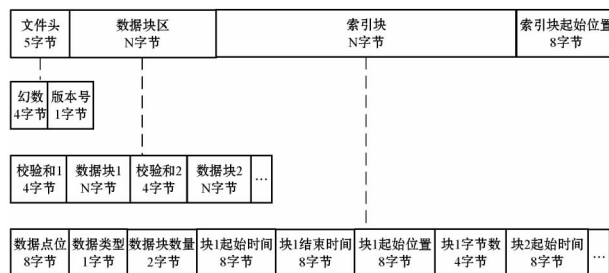


图5 TSM 数据文件结构图

## 3 数据的无损压缩方法

时序数据包含点名、时间和数值信息,写入一组时序数据时,如图6所示,压缩器遍历每个点,判断数值的数据类型是整数、浮点数、布尔值还是字符串,分别调用相应的压缩方法进行压缩,时间类型调用时间压缩方法压缩,将两者压缩得到的字节数组合并后写入到数据文件中。整数、浮点数、布尔、字符串、时间五种数据类型的压缩算法如下所述。

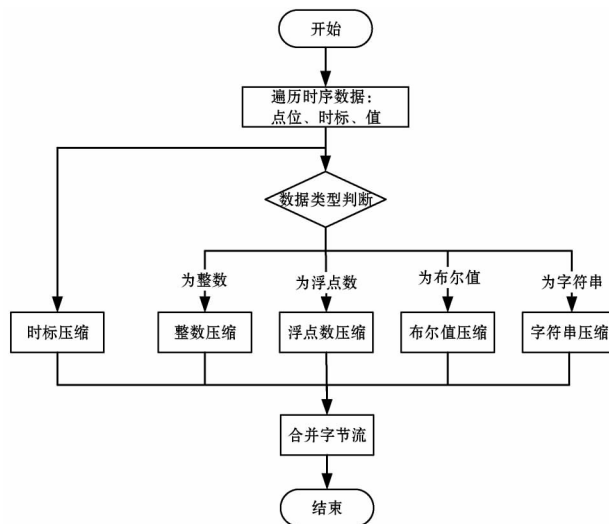


图6 数据压缩流程图

整数的压缩方法为: 第一个整数不压缩,从第二个整数开始计算与前一个数的差值,并对差值通过 Zig-Zag 编码<sup>[7]</sup>将差值为负数的变为正数,然后分三种情况进行压缩:(1) 如果连续两个以上差值相等,则只存储差值及差值出现的次数;(2) 如果差值超过十六进制数值 0x0800000000000000,不压缩;(3) 否则,将差值采用 simple8b 算法<sup>[8]</sup>进行压缩。

浮点数的压缩方法<sup>[9]</sup>为: 第一个浮点数不压缩,从第二个浮点数开始与前一个数进行异或计算,得到一

个 64 位无符号整数作为差值,当两个浮点数数值接近时得到的差值很小。当差值为 0 时仅存 1 位 0;不为零时存 1 位 1,然后用 5 位存储 64 位中位于左端的 0 的数量,用 6 位存储居右端的 0 的数量,再将非零位截取出来存储。

布尔值的压缩方法为:直接将布尔值用 1 位存储到字节末尾(每个字节有 8 位),当存满 1 个字节后分配一个新字节。

字符串的压缩方法为:将字符串顺序添加到字节流后用 snappy 算法<sup>[6]</sup>压缩。

时间类型的压缩方法为:第一个时间不压缩,从第二个时间开始与前一个时间进行差值计算。第一个差值不压缩,然后从第二个差值开始计算与前一个差值的差值。如果差值的差值为 0(当数据的存储间隔相同时),仅存储 0 和 0 出现的次数;否则采用 simple8b 算法<sup>[8]</sup>存储该差值的差值。

## 4 读写性能测试

influxDB 开源代码<sup>[10]</sup>采用 GO 语言实现,本文将引擎部分用 .net framework 4.5 重新实现,通过随机数发生器每秒生成 50 000 点整型随机数,进行了性能测试。测试硬件设备为一台 ThinkPad T550 笔记本电脑, CPU: i5 5200U 2.2 G 双核,内存: 8 GB,硬盘: 256 GB 固态硬盘。

写入测试是每秒随机发生 5 万点整型数据,重复 3 600 次写入,写入平均耗时 310 毫秒。读取性能测试进行了两组,一组测试从 1 条/点到 200 条/点,分 200 次读取 5 万点的数据,读取的数据条目数从 5 万条到 1 000 万条,如图 7 所示,读取耗时从 2.8 秒到 9.2 秒,随着条数增加耗时呈线性增长趋势;另一组测试分 50 次固定读取 100 万条数据,点数从 1 000 开始以 1 000 点步长一直增长到 5 万点,得到的曲线如图 8 所示,随着点数增加,用时从 0.6 秒到 3.3 秒呈线性增长趋势。

根据测试结果,引擎能够实现单实例每秒 5 万点的数据写入,读取相同记录数的情况下,点数越少速度越快,这主要是在 TSM 文件中,一段时间内相同点的数据是按时间顺序连续存储的,而不同点的数据是间隔存储的,因此点数越多文件位置的计算越多。考虑到工业时序数据库应用以单点历史曲线读取为主, TSM 文件结构有利于提高读性能。

引擎的时间消耗中,用于集合运算和比较约占 1/3,五种数据类型的压缩和解压缩约占 2/3。由于 .net framework 相较 .net core 集合运算性能差距较大,加上浮点数、整数压缩算法性能还有提升空间,引擎通过

.net core 重新编译并采用性能更高的压缩算法后,读写性能会有较大提升。

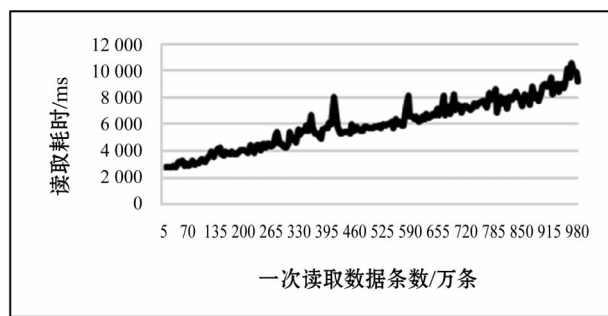


图 7 5 万点数据读取速度曲线

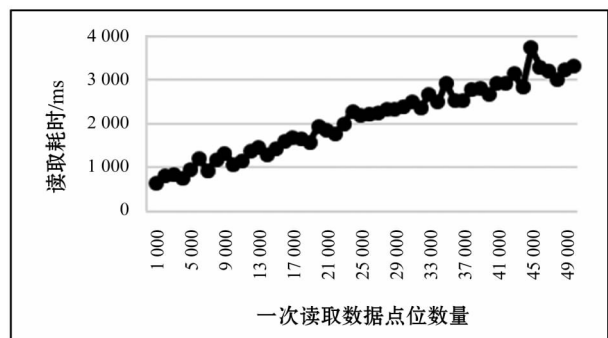


图 8 100 万条数据读取速度曲线

## 5 结 语

本文按照工业时序数据的特点,参考 influxDB 思想设计了工业时序数据库的引擎。引擎自顶而下包括数据库实例、存储策略、存储分片几个层次,每个存储分片里包含一个内存缓存区、一个或更多日志文件、多个数据文件,以及用于处理写入、压缩、读取的线程。本文简化了 TSM 文件结构,数据点位关键字由 influxDB 的字符型改为整型,降低了存储占用和索引计算量。最后,进行了引擎的开发和性能测试。

未来,还需要做三个方面的改进:一是利用 .net core 重新编译提高性能;二是压缩解压缩部分替换为性能更高的算法库;三是增加分布式冗余机制,以便实现更大规模的数据吞吐量,并提高数据容错能力。

## 参 考 文 献

- [1] 肖子达,朱立谷,冯东煜,等. 分布式数据库聚合计算性能优化[J]. 计算机应用,2017,37(5): 1251-1256.
- [2] Influxdata [OL]. 2018. <https://www.influxdata.com/>.
- [3] DB-Engines [OL]. 2018. <https://db-engines.com/en/ranking/time+series+dbms>.
- [4] O'Neil P, Cheng E, Gawlick D, et al. The log-structured merge-tree (LSM-tree) [J]. Acta Informatica, 1996, 33(4): 351-385.

(下转第 40 页)



图5 设备状态实时监视

故障诊断系统生成的诊断报告如图6所示。可以引导操纵员判断道岔安全状态出现异常的具体原因和部位,同时针对每一个诊断结果给出相应的处置意见和操作步骤。

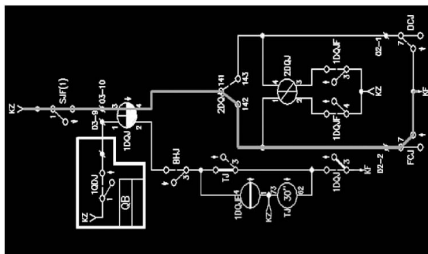
名称	【14063】道岔定到反操纵后未动作: 1DQJ 未吸起(单操)
报警描述	当计算机联锁不送 DCJ、FCJ 给监测时,使用该逻辑。如果送了 DCJ、FCJ 优先用 DCJ、FCJ 码位逻辑。 1) 人工单扳道岔(联锁送的单操按钮); 2) 道岔和所在区段均不处于锁闭的状态; 3) 本机 1DQJ 不吸起; 4) 多机联动的道岔,例如同尖/心轨的前机 1DQJ 没有吸起,后机不再报警
相关电路	
维护建议	1) 请查看 DCJ/FCJ 是否动作; 2) 如 DCJ/FCJ 未动作,则是联锁系统问题,MMI 单操行为未驱动 DCJ/FCJ; 3) 如 DCJ/FCJ 动作,则是 1DQJ 励磁电路问题,请从 KZ 至 KF 依次逐段查找电路中故障点。→更换故障继电器或配线

图6 故障诊断报告

### 3 结 语

本文分析了道岔装置对运行支持系统的需求,提出了针对性的设计方案。主要特点体现在以下3方面:

1) 对数据采集通话管理进行了状态设计,将系统功能模块化、状态化,保障了系统的可用性;

2) 采用机器学习方法避免了人工方式提取特征并不能包含原始曲线的所有信息导致很多很重要的信息丢失的缺点,具有提取一些和最终结果具有很强相关性的隐含特征的优点,以便最大程度地提高准确性;

3) 采用符合操纵员运行操纵经验的表达方式,快速、直观、敏锐地感知装置出现的异常运行状态,对报警进行描述并给出相关电路和维修建议,提高对装置运行状态的评判效率,减少误判断。

### 参 考 文 献

- [1] Roberts C, Dassanayake H P B, Lehasab N, et al. Distributed quantitative and qualitative fault diagnosis: railway junction case study [J]. Control Engineering Practice, 2002, 10 (4): 419-429.
- [2] Mchutchon M A, Staszewski W J, Schmid F. Signal Processing for Remote Condition Monitoring of Railway Points [J]. Strain, 2005, 41(2): 71-85.
- [3] 张帆, 黄世泽, 郭其一, 等. 基于故障树分析法的道岔故障诊断与可靠性评估方法 [J]. 城市轨道交通研究, 2018, 21(10): 58-62.
- [4] 许庆阳, 刘中田, 赵会兵. 基于隐马尔科夫模型的道岔故障诊断方法 [J]. 铁道学报, 2018, 40(8): 102-110.
- [5] 程雯丽. 判别和聚类方法在针灸临床数据分析中的应用研究 [D]. 上海: 东华大学, 2013.
- [6] 皮勇泽, 苏厚勤, 黄琴峰. 中医针灸临床治疗专家系统的研究与实现 [J]. 计算机应用与软件, 2015, 32(6): 99-103.
- [7] 付琴. DNA 优化的 BP 神经网络在故障诊断中的应用 [J]. 科学技术与工程, 2012, 12(29): 7592-7597.
- [8] 关琼. 基于 FOA-LSSVM 的高速铁路道岔故障诊断 [J]. 科技通报, 2015, 31(4): 230-232.
- [9] 唐维华. 基于 LSTM/NN 的道岔故障特征提取与识别研究 [J]. 计算机应用与软件, 2019, 36(1): 159-163.

### (上接第36页)

- [5] Chang F, Dean J, Ghemawat S, et al. Bigtable: A distributed storage system for structured data [J]. Acm Transactions on Computer Systems, 2008, 26(2): 1-26.
- [6] Snappy [OL]. 2018. <http://code.google.com/p/snappy/>.
- [7] ZigZag [OL]. 2018. <https://developers.google.com/protocol-buffers/docs/encoding>.
- [8] Anh V N, Moffat A. Index compression using 64-bit words [J]. Software Practice and Experience, 2010, 40(2): 131-147.
- [9] Pelkonen T, Franklin S, Teller J, et al. Gorilla: A Fast, Scalable, In-Memory Time Series Database [J]. Proceedings of the Vldb Endowment, 2015, 8(12): 1816-1827.
- [10] influxDB [OL]. 2018. <https://github.com/influxdata/influxdb>.