## 时序数据库相关产品简要介绍及对比

### 时序数据库简要介绍

#### InfluxDB

InfluxDB是一个由InfluxData开发的开源时序型数据库，专注于海量时序数据的高性能读、高性能写、高效存储与实时分析等，在DB-Engines Ranking时序型数据库排行榜上排名第一，广泛应用于DevOps监控、IoT监控、实时分析等场景。

InfluxDB部署简单、使用方便，在技术实现上充分利用了Go语言的特性，无需任何外部依赖即可独立部署。提供类似于SQL的查询语言，接口友好，使用方便。丰富的聚合运算和采样能力，提供灵活的数据保存策略（Retention Policy）来设置数据的保留时间和副本数，在保障数据可靠性的同时，及时删除过期数据，释放存储空间，提供灵活的连续查询（Continues Query）来实现对海量数据的采样。支持协议种类多，除了HTTP、UDP等原生协议，还兼容CollectD、Graphite、OpenTSDB、Prometheus等组件的通讯协议。

官网：[InfluxDB Time Series Data Platform | InfluxData](https://www.influxdata.com/)

##### 特点

* 无系统环境依赖，部署方便。
* 无结构化（SchemaLess）的数据模型，灵活强大。
* 原生HTTP管理接口，免插件配置和免第三方依赖。
* 强大的类SQL查询语句的操作接口，学习成本低，上手快。
* 丰富的权限管理功能，精细到“表”级别。
* 丰富的时效管理功能，自动删除过期数据，自定义删除指标数据。
* 低成本存储，采样时序数据，压缩存储。
* 丰富的聚合函数，支持AVG、SUM、MAX、MIN等聚合函数。

#### IoTDB

一体化收集、存储、管理与分析物联网时序数据的软件系统。 Apache IoTDB 采用轻量式架构，具有高性能和丰富的功能，并与Apache Hadoop、Spark和Flink等进行了深度集成，可以满足工业物联网领域的海量数据存储、高速数据读取和复杂数据分析需求。

官网：[IoTDB Website (apache.org)](https://iotdb.apache.org/zh/)

##### 特点

* 灵活的部署策略。IoTDB为用户提供了一个在云平台或终端设备上的一键安装工具，以及一个连接云平台和终端上的数据的数据同步工具。
* 硬件成本低。IoTDB可以达到很高的磁盘存储压缩比。
* 高效的目录结构。IoTDB支持智能网络设备对复杂时间序列数据结构的高效组织，同类设备对时间序列数据的组织，海量复杂时间序列数据目录的模糊搜索策略。
* 高吞吐量读写。IoTDB支持数以百万计的低功耗设备的强连接数据访问、高速数据读写，适用于上述智能网络设备和混合设备。
* 丰富的查询语义。IoTDB支持跨设备和测量的时间序列数据的时间对齐、时间序列字段的计算(频域转换)和时间维度的丰富聚合函数支持。
* 学习成本非常低。IoTDB支持类似sql的语言、JDBC标准API和易于使用的导入/导出工具。
* 与先进的开放源码生态系统的无缝集成。IoTDB支持分析生态系统，如Hadoop、Spark和可视化工具(如Grafana)。

#### TDengine

一款开源、高性能、云原生的[时序数据库](https://www.taosdata.com/" \o "时序数据库" \t "https://www.taosdata.com/_blank)，具有极强的弹性伸缩能力，同时带有内建的缓存、流式计算、数据订阅等功能

官网：[TDengine 产品 - TDengine | 涛思数据 (taosdata.com)](https://www.taosdata.com/products)

##### 特点

* 高效写入，支持 SQL 写入和 Schemaless 写入
* 高效查询，支持 SQL，支持嵌套查询、UDF 等
* 提供一系列时序数据特有查询和窗口函数
* 支持集群、云原生，支持水平扩展
* 带有内建的缓存功能，可以不用部署 Redis
* 带有内建的流式计算，可以不用部署 Spark/Flink
* 带有内建的数据订阅，可以不用部署Kafka
* 支持各种编程语言，与很多第三方工具无缝集成

#### KairosDB

KairosDB最初是从OpenTSDB 1.x版本fork出来的一个分支，目的是在OpenTSDB的代码基础上进行二次开发来满足新的功能需求。其改造之一就是支持可插拔式的存储引擎，例如支持H2可以方便本地开发和测试，而不是像OpenTSDB一样与HBase强耦合。在其最初的几个版本中，HBase也是作为其主要的存储引擎。但是在之后的存储优化中，慢慢使用Cassandra替换了HBase，它也是第一个基于Cassandra开发的时序数据库。在最新的几个版本中，已不再支持HBase，因为其存储优化使用了Cassandra所特有而HBase没有的一些特性。

在整体架构上，和OpenTSDB比较类似，都是采用了一个比较成熟的**数据库来作为底层存储引擎**。自己的主要逻辑仅仅是在存储引擎层之上很薄的一个逻辑层，这层逻辑层的部署架构是一个无状态的组件，可以很容易的水平扩展。

官网： [KairosDB](https://kairosdb.github.io/)

##### 特点

* 数据采集：数据可以通过多种协议写入KairosDB，比如Telnet的按行写入，HTTP API，Graphite以及批处理导入。此外，还可以使用或者自己编写插件。
* 存储：KairosDB 采用了 Cassandra 作为数据存储方式，Cassandra 也是一个比较流行的NoSQL数据库，很多开源软件基于此数据库。
* Rest API：KairosDB提供了REST API，已完成对metric名称，tag等的查询，当然，也少不了存储和查询数据点（data points）。
* 自定义数据类型（Custom Data）：KairosDB支持存储和聚合自定义数据类型。默认情况下KairosDB支持long、double和字符串的value，这比OpenTSDB要丰富一些。
* 分组和聚合：作为数据分析系统，分组和聚合则是必不可少的功能。 KairosDB的聚合（也就是down samples）功能，支持的标准函数有min、max、sum、count、mean、histogram、gaps等，而且都非常实用。比如percentile，可以计算一个指标值大概的百分比位置，非常适合存储类似“你打败了xx%的人”这种需求场景。
* 支持工：KairosDB提供了进行数据导入导出的命令行工具。根据官方文档的说明，在一台分配了2Gig内存的SSD Cassandra上，1秒钟能导入 13万条数据。
* 插件机制：KairosDB也提供多种基于Guice的插件机制来进行扩展（data point监听器，数据存储，协议处理等。）KairosDB是从OpenTSDB fork过来的，因此最初它是支持HBase的，不过现在HBase已经不能完全支持KairosDB所需的特性，将来会取消对HBase的支持。

#### TimescaleDB

基于 PostgreSQL 打造的一款时序数据库，采用插件的形式，可以跟随社区主版本，融合 PostgreSQL 生态，使得PostgreSQL 也一跃成为一款优秀的时序数据库。TimescaleDB 在 PostgreSQL 的总时间的十五分之一内加载十亿行数据库，并且在更大数据写入量时，吞吐量甚至超过 PostgreSQL 20 倍。

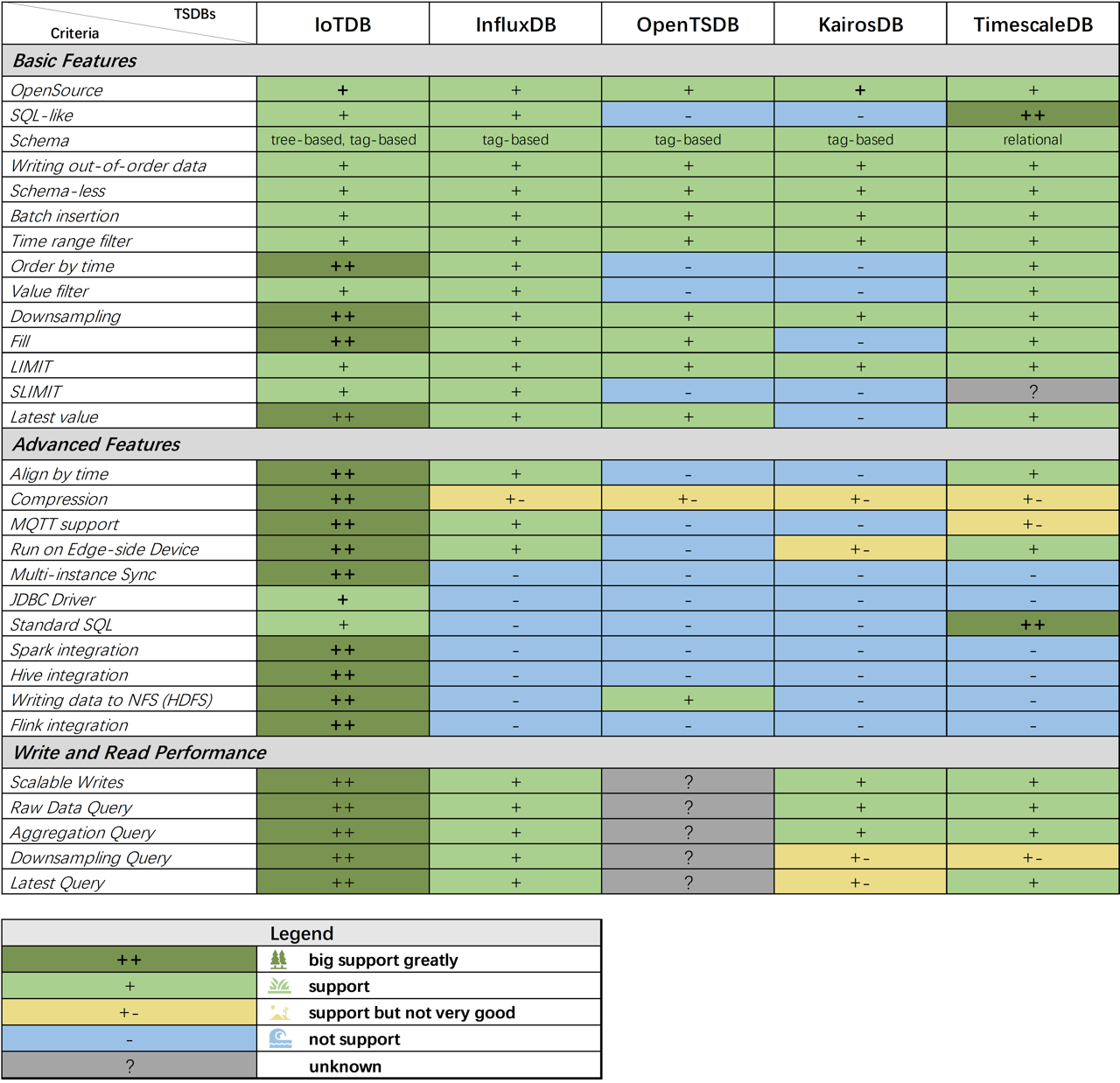
官网：[Timescale Docs](https://docs.timescale.com/)

### 数据库相关对比

如下图为相关时序数据从 **基础功能**、**高级功能**、**性能**三个方面的对比：

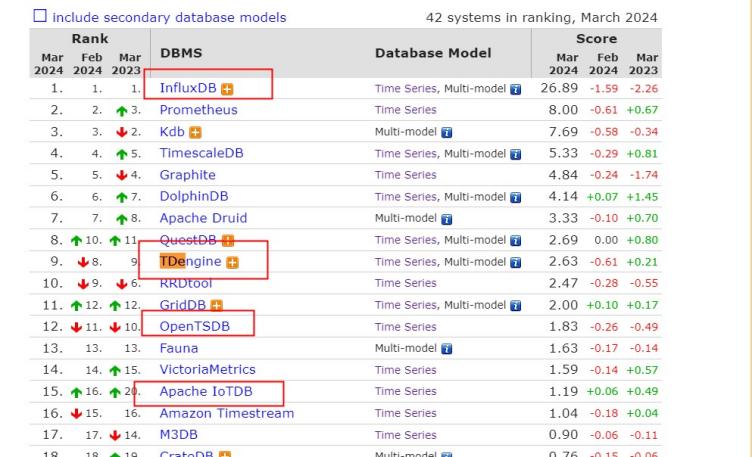
表格中符号的含义：

* ++：强大支持
* +：支持
* +-：支持但欠佳
* -：不支持
* ?：未知



#### 排名

几款时序数据库，在db-engines网站上的排名对比



#### 功能

通过对基础功能的比较，我们可以发现：

* OpenTSDB 和 KairosDB 缺少一些重要的查询功能。
* TimescaleDB 不能被企业免费使用。
* IoTDB 和 InfluxDB 可以满足时间序列数据管理的大部分需求，同时它俩之间有一些不同之处。

##### Schema

* IoTDB：IoTDB 提出了一种 基于树的 schema。这和其它时间序列数据库很不一样。这种 schema 有以下优点：
  + 在许多工业场景里，设备管理是有层次的，而不是扁平的。因此我们认为基于树的 schema 比基于 tag-value 的 schema 更好。
  + 在许多现实应用中，tag 的名字是不变的。例如：风力发电机制造商总是用风机所在的国家、所属的风场以及在风场中的 ID 来标识一个风机，因此，一个 4 层高的树（“root.the-country-name.the-farm-name.the-id”）来表示就足矣。你不需要重复告诉 IoTDB”树的第二层是国家名”、“树的第三层是风场名“等等这种信息。
  + 这样的基于路径的时间序列 ID 定义还能够支持灵活的查询，例如：”root.\*.a.b.\*“，其中、\*是一个通配符。
* InfluxDB, KairosDB, OpenTSDB：使用基于 tag-value 的 schema。现在比较流行这种 schema。
* TimescaleDB 使用关系表。

##### **order by time**

事实上，所有时间序列数据库都支持单条时间序列的按时间戳排序。但是，OpenTSDB 和 KairosDB 不支持多条时间序列的按时间戳排序。

下面考虑一个新的例子：这里有两条时间序列，一条是风场 1 中的风速，一条是风场 1 中的风机 1 产生的电能。如果我们想要研究风速和产生电能之间的关系，我们首先需要知道二者在相同时间戳下的值。也就是说，我们需要按照时间戳对齐这两条时间序列。因此，结果应该是：

|  |  |  |
| --- | --- | --- |
| 时间戳 | 风场1中的风速 | 风场1中的风机1产生的产能 |
| 1 | 5.0 | 13.1 |
| 2 | 6.0 | 13.3 |
| 3 | Null | 13.1 |

或者：

|  |  |  |
| --- | --- | --- |
| 时间戳 | 时间序列名 | 值 |
| 1 | 风场1中的风速 | 5.0 |
| 1 | 风场1中的风机1产生的产能 | 13.1 |
| 2 | 风场1中的风速 | 6.0 |
| 2 | 风场1中的风机1产生的产能 | 13.3 |
| 3 | 风场1中的风机1产生的产能 | 13.1 |

虽然第二个表格没有按照时间戳对齐两条时间序列，但是只需要逐行扫描数据就可以很容易地在客户端实现这个功能。

IoTDB 支持第一种表格格式（叫做 align by time），InfluxDB 支持第二种表格格式。

##### Latest value

最基础的时间序列应用之一是监视最新数据。因此，返回一条时间序列的最新点是非常重要的查询功能。

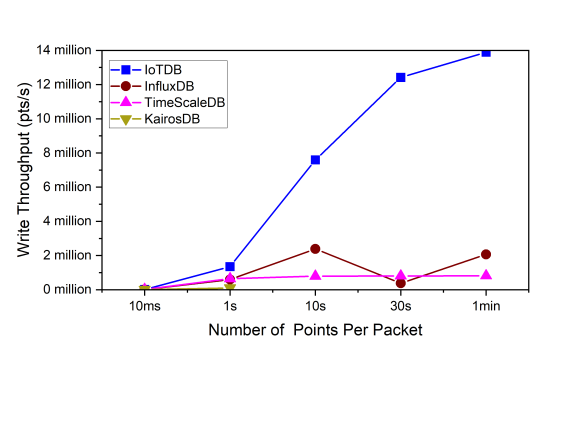
IoTDB 和 OpenTSDB 使用一个特殊的 SQL 或 API 来支持这个功能，而 InfluxDB 使用聚合函数来支持。

IoTDB 提供一个特殊的 SQL 的原因是 IoTDB 专门优化了查询。

#### 性能

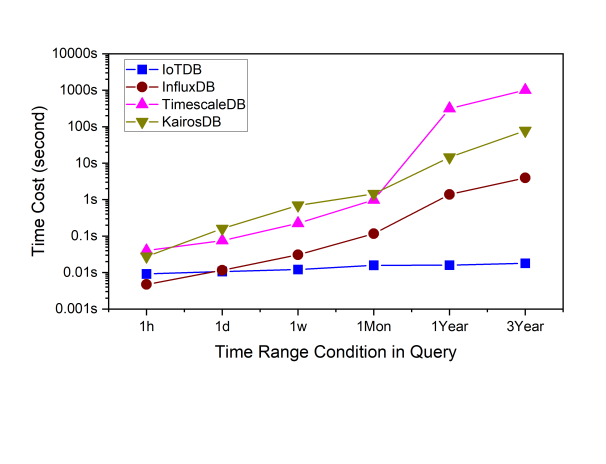
##### 写入性能

批量写入



##### 查询性能

聚合查询



### 小结

IoTDB 具有最小的写入延迟。批处理大小越大，IoTDB 的写入吞吐量就越高。这表明 IoTDB 最适合批处理数据写入方案。

在高并发方案中，IoTDB 也可以保持吞吐量的稳定增长。 （每秒 1200 万个点可能已达到千兆网卡的限制）

在原始数据查询中，随着查询范围的扩大，IoTDB 的优势开始显现。因为数据块的粒度更大，列式存储的优势体现出来，所以基于列的压缩和列迭代器都将加速查询。

在聚合查询中，我们使用文件层的统计信息并缓存统计信息。因此，多个查询仅需要执行内存计算（不需要遍历原始数据点，也不需要访问磁盘），因此聚合性能优势显而易见。

降采样查询场景更加有趣，因为时间分区越来越大，IoTDB 的查询性能逐渐提高。它可能上升了两倍，这对应于 2 个粒度（3 小时和 4.5 天）的预先计算的信息。因此，分别加快了 1 天和 1 周范围内的查询。其他数据库仅上升一次，表明它们只有一个粒度统计。

总的来说，对比现有的时序数据库，已IoTDB、InfluxDB、TDengine三种为主，他们之间的差异也相对较小，有各自适用的场景，比如，IoTDB在数据计算上较为优势，支持MapReduce，可以支持Hadoop和Spark。在db-engines上，排名前后依次为 InfluxDB > TDengine > IoTDB。

### 参考

* [时间序列数据库比较 | IoTDB Website (apache.org)](https://iotdb.apache.org/zh/UserGuide/Master/stage/TSDB-Comparison.html" \l "overview)
* [Apache IoTDB vs. InfluxDB vs. TDengine Comparison (db-engines.com)](https://db-engines.com/en/system/Apache+IoTDB;InfluxDB;TDengine)