TDengine

高性能、分布式、支持SQL的时序数据库

张玮绚（Wade Zhang）

涛思数据研发VP

Agenda

- *TDengine诞生的行业背景*
- *TDengine的产品特性*
- *开源和商业策略*
- *客户案例*

TDengine 诞生的行业背景

什么是时序数据库

对时序数据进行存储、查询和计算服务的数据库

什么是时序数据

- 按时间顺序产生的结构化数据
- 同一数据源所产生的时序数据是完全同构的
- 同一类数据源所产生的时序数据也往往是同构的或高度相似的

时序数据十大特征

1. 所有采集的数据都是时序的
2. 数据都是结构化的
3. 一个采集点的数据源是唯一的
4. 数据很少有更新或删除操作
5. 数据一般是按到期日期来删除的
6. 数据以写操作为主，读操作为辅
7. 数据流量平稳，可以较为准确的计算
8. 数据都有统计、聚合等实时计算操作
9. 数据一定是指定时间段和指定区域查找的
10. 数据量巨大，取决于采集点的数量和采集间隔

时序数据（库）的应用领域

电信 上网记录，通话记录，用户行为，设备监测

电网 智能电表，设备监测

银行 交易记录，ATM/POS监测

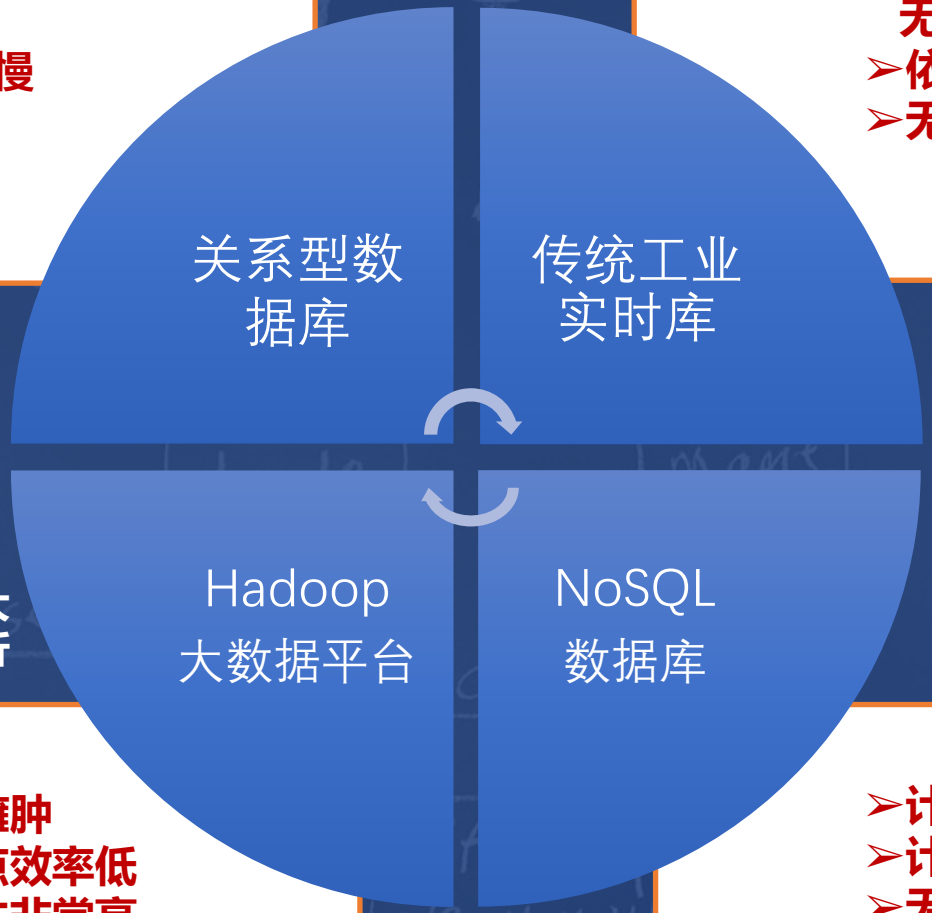
交通物流 交通工具和集装箱货物的位置追踪

IT运维 服务器和业务系统的监测

传统制造业 生产设备的实时监控

其它 所有能产生时序数据的领域

时序数据存储方案的各种选择



- ➢海量时序数据读写性能低
- ➢分布式支持差
- ➢数据量越大，查询越慢
- ➢报表分析慢，T+N

典型场景：用电、能耗监控

- ➢架构陈旧无分布式方案，无法水平扩展
- ➢依赖windows等环境
- ➢无法云化部署

典型场景：scada系统、生产监控系统

典型场景：多矿区集中监控大数据平台，海量历史数据分析

- ➢组件多而杂，架构臃肿
- ➢支持分布式但单节点效率低
- ➢硬件、人力维护成本非常高

典型场景：井下状态监控平台，设备数据存储

- ➢计算实时性差，查询慢
- ➢计算内存、CPU开销巨大
- ➢无时序针对性优化

TAOS DATA

TDengine 的产品特性

- 高性能：插入和查询性能都优于InfluxDB
- 分布式：集群方案开源
- 支持SQL：语法尽量与标准SQL兼容

优势一： 高性能写入

测试代码路径:
https://github.com/taosdata/timeseriesdatabase-comparisons

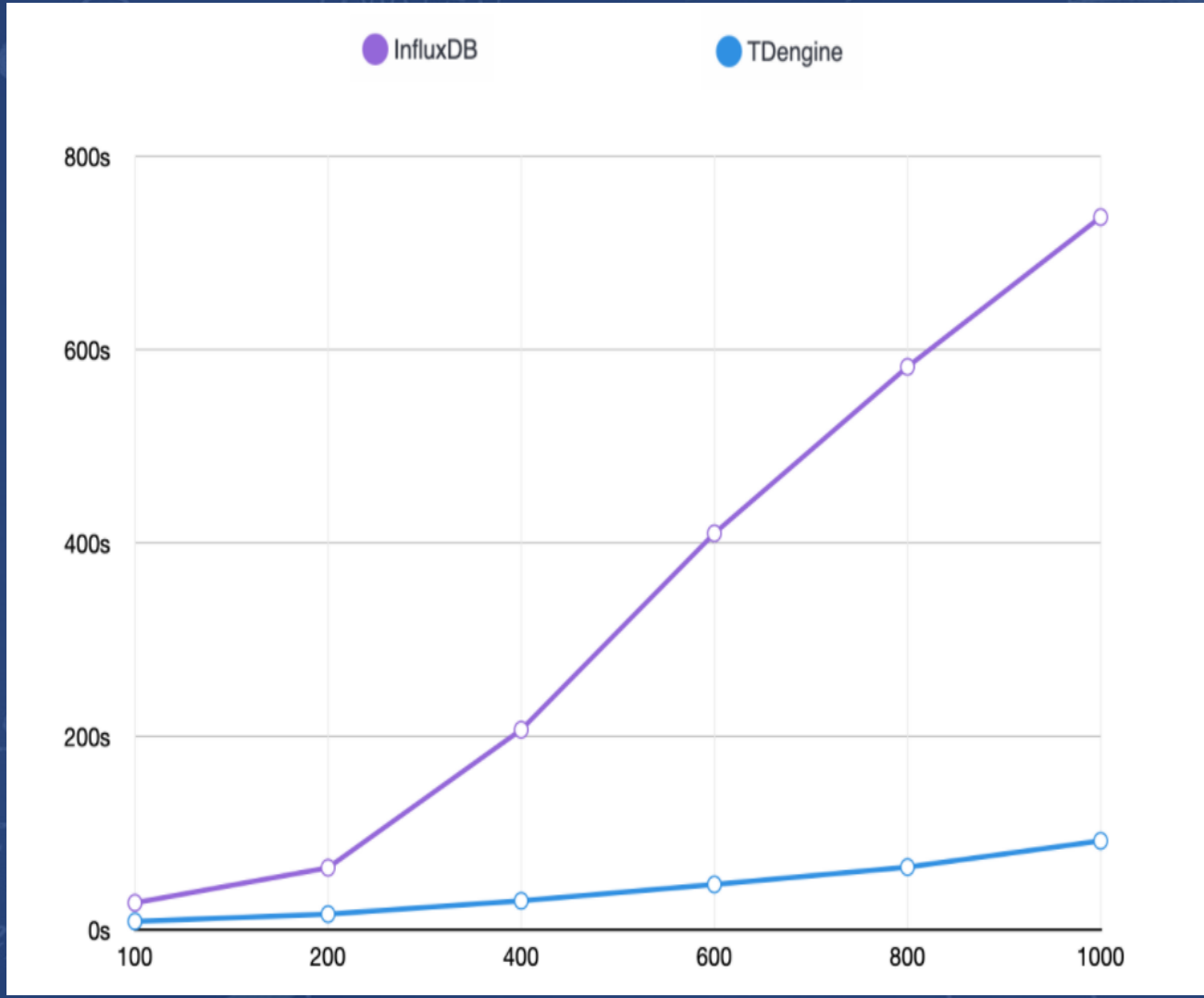
写入场景 1:
在batchsize和workers的所有组合下，TDengine都全面胜出， 有2到9倍的优势

模拟的设备数量	batch size	workers	Vnode	TDengine WAL2		InfluxDB结果	
				总用时（秒）	每秒记录条数	总用时（秒）	每秒记录条数
1000台设备 9000张表 总共 77760000条记录	1	1	1	-	-	-	-
		16		4,089.88	19,013	7,599.54	10,232
		50		1,841.58	42,225	4,784.04	16,254
		100		1,554.64	50,018	3,837.48	20,263
	1000	1		-	-	-	-
		16		118.15	658,119	515.51	150,840
		50		87.63	887,324	485.07	160,306
		100		82.48	942,729	486.20	159,934
	2500	1		-	-	-	-
		16		98.98	785,602	561.09	138,587
		50		77.09	1,008,741	511.34	152,072
		100		76.46	1,016,970	494.99	157,094
	5000	1		-	-	-	-
		16		91.58	849,053	737.04	105,503
		50		73.08	1,064,089	641.12	121,287
		100		71.09	1,093,761	633.49	122,748

优势一： 高性能写入

测试代码路径:
https://github.com/taosdata/timeseries-database-comparisons

写入场景2:
固定batchsize 500, workers 16，固定每台设备的数据写入量，但不断增加设备数量，发现**InfluxDB的写入耗时增加速度明显高于TDengine**。



优点二： 高性能查询

测试代码路径：https://github.com/taosdata/timeseriesdatabase-comparisons

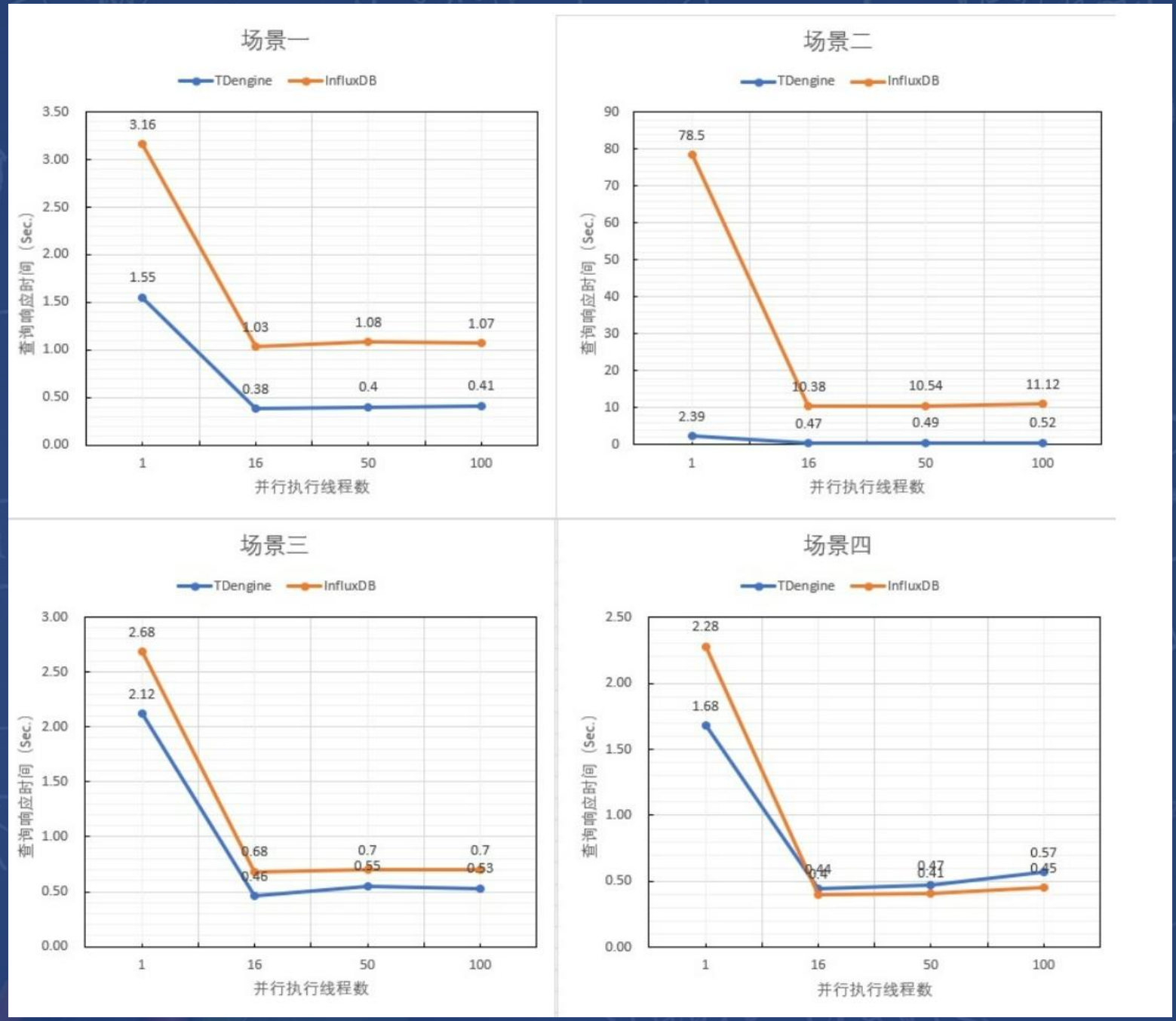
待查询数据集：1000 台设备 ， 9000 个子表，每个设备间隔 10 秒生成一条记录，总记录数是 **77,760,000** 条记录。

查询性能测试包括四个测试场景：

- 场景一：通过标签过滤随机筛选出 8 个时间线后，取其中的最大值。
- 场景二：随机选取1个小时时间区间，通过标签随机筛选出 8 个时间线，取其中的最大值。
- 场景三：随机选取12小时的时间区间，通过标签随机筛选出 8 个时间线，使用 10 分钟作为一个时间窗口，获取每个时间窗口的最大值。
- 场景四：随机选取 1 小时时间区间，通过标签过滤随机筛选出 8 个时间线，使用 1 分钟为一个时间窗口，获取每个时间窗口的最大值。

优点二： 高性能查询

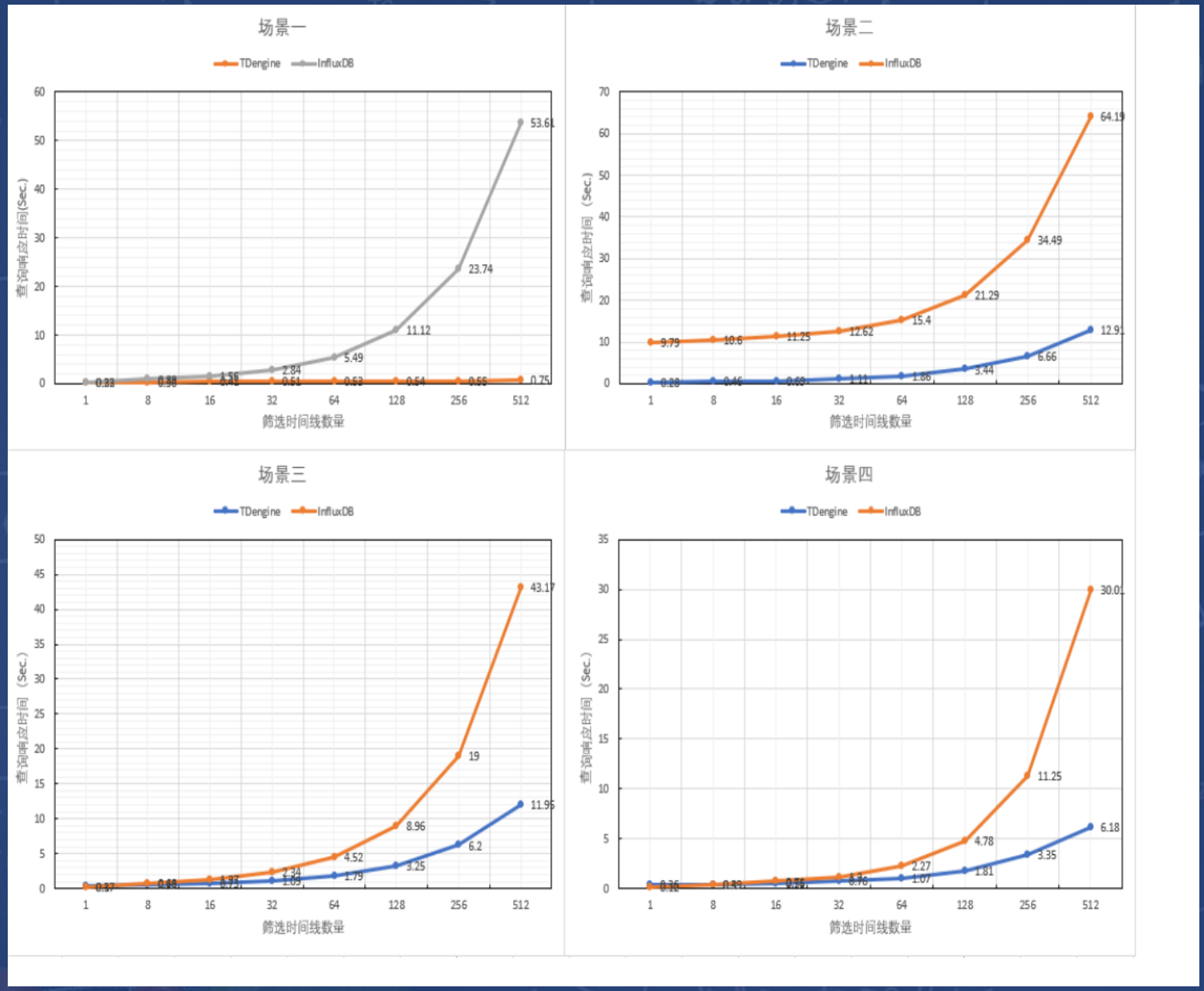
在绝大多数测试场景和batchsize, worker的组合下，TDengine的查询性能都明显优于InfluxDB，只有场景四中在workers超过16之后略弱于InfluxDB．



优点二： 高性能查询

在四种测试场景下通过**不断增加查询条件中的时间线数量**，可以看出TDengine的查询性能随着时间线数量的增加下降得比InfluxDB要缓慢得多。

换句话说，**查询条件中的时间线越多，TDengine的查询性能优势越明显。**



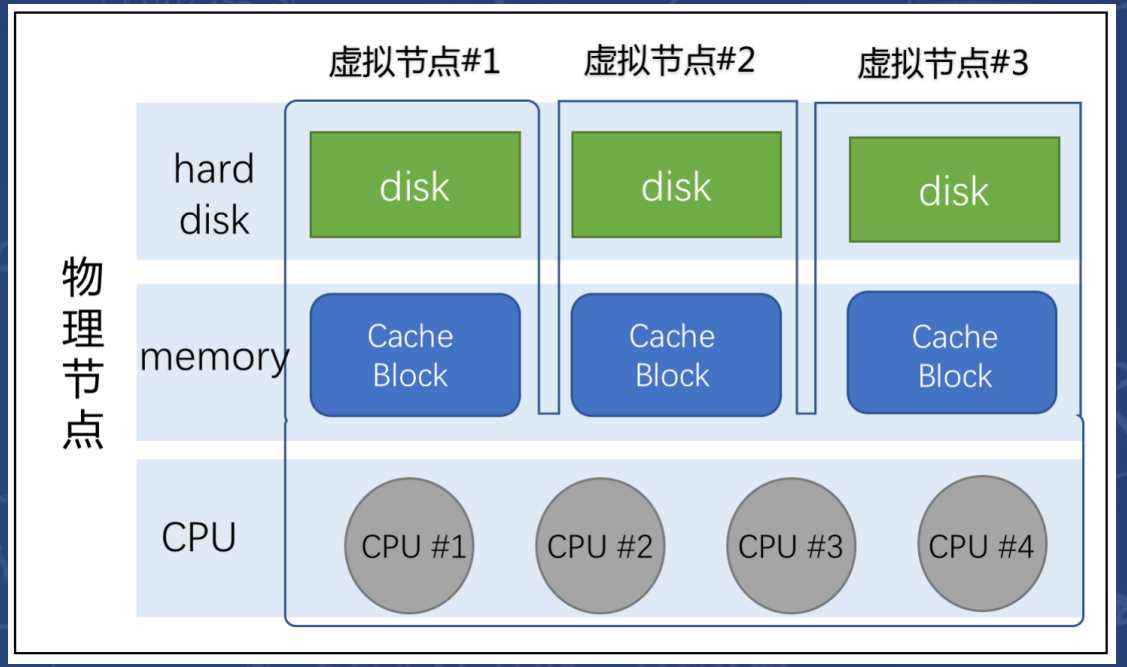
优点三： 开源的分布式集群解决方案

- InfluxDB 集群方案未开源
- Prometheus没有集群方案
- TDengine的集群方案能够很好地水平扩展，轻松支持海量级的时间线和数据量，**而且是开源的**。

集群架构

虚拟节点(VNODE): 物理节点包含多个虚拟节点。各虚拟节点是相对独立的工作单元，是数据管理的基本单元，具有独立的运行线程、内存空间与持久化存储路径。虚拟节点包含若干表（数据采集点）的数据。一个物理节点上能创建的虚拟节点的数量取决于该节点的硬件资源。

管理节点(MNODE): 负责所有虚拟节点运行状态的监控、维护，以及虚拟节点之间的负载均衡。同时，管理节点也负责元数据(包括用户、数据库、表、标签等)的存储和管理，也称为元数据节点(Meta Node)。不同的管理节点强一致方式进行数据同步，任何数据更新操作只能在主管理节点上进行。



物理节点(DNODE): 一个运行实例，一个工作的系统包含至少一个物理节点。物理节点包含零到多个虚拟节点，以及至多一个管理节点(mnode)。

集群架构

管理节点（mnode）集群：管理节点负责所有数据节点运行状态的监控和维护，以及节点之间的负载均衡。同时，管理节点也负责元数据(包括用户、数据库、表、标签等)的存储和管理。整个集群中有不多于三个mnode组成了元数据管理集群。元数据集群采用master/slave结构，而且采取强一致方式进行数据同步，任何数据更新操作只能在 Master 上进行。**整个集群中只有一个管理节点集群**。

存储节点(vnode)集群: vnode是存储和管理时序数据的基本单元，具有独立的运行线程、内存空间与持久化存储路径。不同物理节点上的vnode可以构成一个集群，vnode集群内采用master/slave方式，写入操作只能在master vnode上进行，异步复制到slave vnode。**整个集群中可以有多个存储节点集群**，其数量上限主要取决于mnode的元数据管理能力和监控能力。

集群的横向扩展：TDengine集群的横向扩展主要指随着物理节点的增加，存储节点相应增加，存储节点集群数量也会相应地增加，由此能够存储更多的时间线和数据量。**一个TDengine物理集群内部由一个管理节点集群和N个存储节点集群构成。**

集群工作机制

多副本：集群内的每个节点保存了相应数据的一个副本。

选主机制：每个内部集群（mnode集群，vnode集群）都会按照选主机制选举出master节点，数据写入操作只能通过相对应集群的master进行。选主机制简单地说是由各节点都维护的一个持久化的版本号依照选主规则进行。

数据复制: 从master到slave的数据复制**默认为异步方式**，以多副本方式保障数据安全性。但如果对数据安全性要求较高的场景，也可以选择配置为**同步数据复制**，相应地对写入性能会有一定影响。

数据分片：按照表级别的粒度进行数据分区（Sharding），把每张表都映射到一个特定的存储节点集群。

数据分区：按照时间段对数据进行分区。每个数据文件包含一个固定长度时间段的时序数据。除了对查询和写入性能的提升之外，也能够高效地实现数据保留策略。

优点四： 支持SQL，易用性高

- SQL语句写入和查询 – 学习成本低
- 多语言连接器：Java, Go, Python, Rust, nodeJS, C#
- RESTful接口：与语言无关的标准HTTP接口

优点五： 安装部署快速简单

- 标准Linux安装包 apt/apt-get/rpm
- 一键解压缩即可用：tar.gz
- 从源代码编译安装
- 容器化部署：单机Docker/Docker Swarm/K8S

优点六： 省空间

- **存储方案：** 在数据块内部按列存储
- **压缩算法:**

数据写入磁盘时，根据系统配置参数comp决定是否压缩数据。TDengine提供了三种压缩选项：无压缩、一阶段压缩和两阶段压缩，分别对应comp值为0、1和2的情况。

一阶段压缩根据数据的类型进行了相应的压缩，压缩算法包括delta-delta编码、simple 8B方法、zig-zag编码、LZ4等算法。

二阶段压缩在一阶段压缩的基础上又用通用压缩算法进行了压缩，压缩率更高。

优点七： 易集成

- TDengine –> RESTful proxy -> Grafana
- TDengine –> RESTful proxy -> Prometheus
- TDengine -> TDengine exportor -> Anywhere
- collectd/telegraf/.... –> RESTful proxy -> TDengine
- Note:
- RESTful proxy i.e. taosAdapter
- TDengine exportor i.e. taosKeeper

优点八： 易运维

- 一个时间线一张表
- 独创的超级表概念，只需为一类采集点创建一张超级表，则所有该类采集点的子表都可以被自动创建

TDengine的开源和商业策略

github.com/taosdata

核心代码全部开源

TAOS DATA

github.com/taosdata/TDengine

社区版开源

2022.02

雄踞趋势榜榜首

开源一周

17,760+
Star

4,300+
Fork

5,200+
PR & Issue

项目	开源时间	Star	Fork
InfluxDB	2013	22,930	3,100
OpenTSDB	2011	4,600	1,300
TimeScale	2017	12,500	660

TAOS DATA

TDengine 商业模式

- **开源社区版**：核心代码完全一样，用来打造品牌，建立开发者社区，建立生态
- **企业版**：支持独立部署，年度订阅模式销售，根据数据量计费
  - **辅助功能**：安全（异地容灾、加密、审计等），运维工具等
  - **技术服务**：任何软件都有BUG，有升级，有维护，需要强力的技术支持
- **云服务版**：在阿里云、AWS等云平台上直接提供PaaS服务，根据数据量和时长计费

TDengine的客户案例

客户案例 – 顺丰 （旧方案及痛点）

- 采用HBase+OpenTSDB 作为大数据监控平台全量监控数据的存储方案
- 平均日数据量约数十亿条
- 主要痛点
  1. 依赖多，稳定性差：依赖Kafka, Spark, HBase，同时数据处理链路长导致平台稳定性降低
  2. 使用成本高：4节点OpenTSDB+21节点HBase，每天1.5T数据量（三副本，压缩后）
  3. 性能差：写入性能基本可接受，但查询经常需要十几秒且OpenTSDB在数据量大的情况下极易崩溃

客户案例 – 顺丰 （技术选型）

- IoTDB

  刚孵化的Apache顶级项目，由清华大学贡献，单机性能不错，不支持集群模式，单机模式在容灾和扩展方面，不能满足需求

- Druid

  性能强大，可扩展的分布式系统，自修复、自平衡、易于操作，但是依赖ZooKeeper和Hadoop作为深度存储，整体复杂度较高

- ClickHouse

  性能最好，但是运维成本太高，扩展特别复杂，使用的资源较多

- TDengine

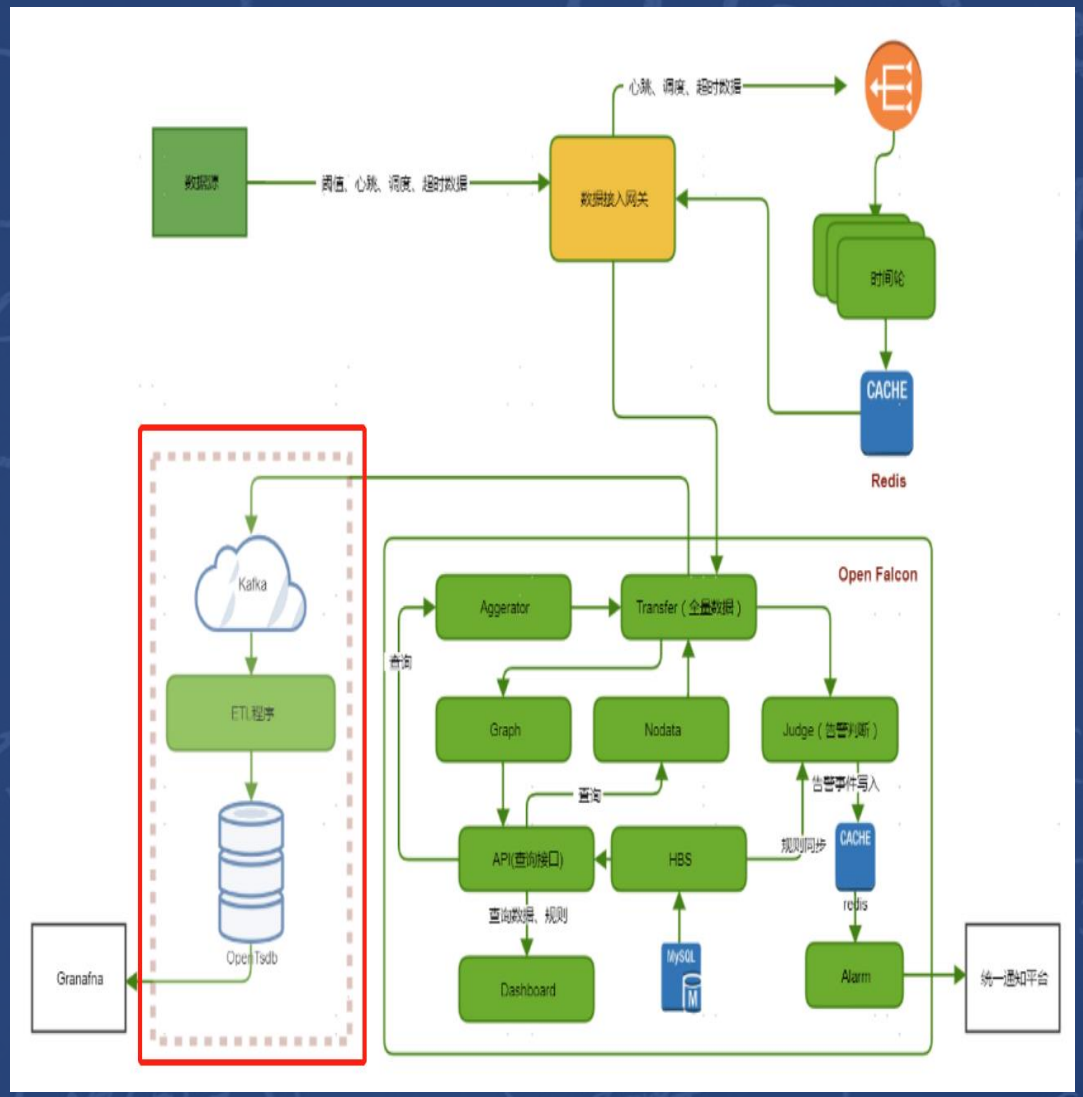
  性能、成本、运维难度都满足，支持横向扩展，且高可用

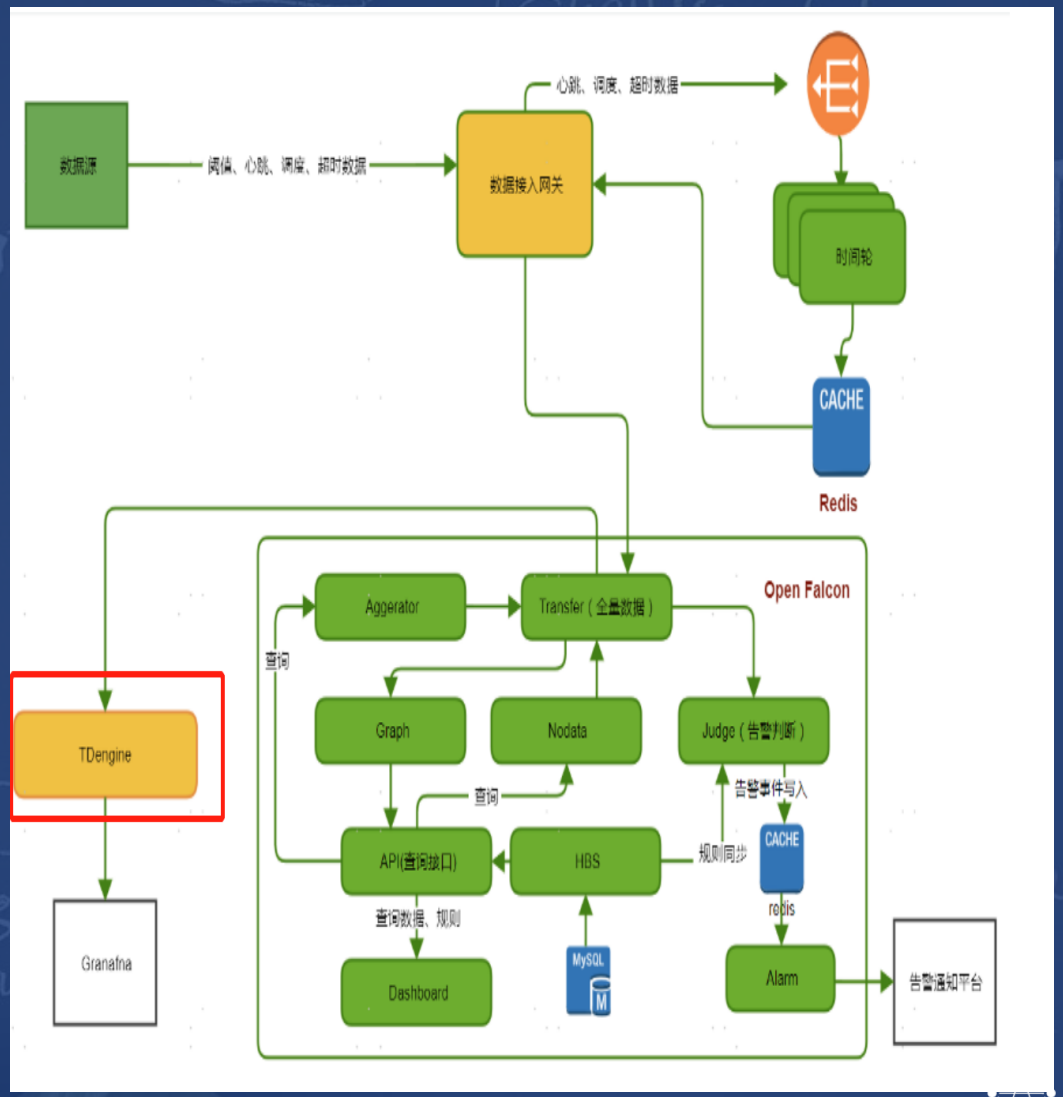
客户案例 – 顺丰 （采用TDengine）

- 数据建模
  - 每一种类型的数据采集点需要建立一个超级表，例如磁盘利用率，方便对同一类型的数据进行聚合分析计算
  - 监控数据本身包括标签信息，直接将标签信息作为超级表的标签列，相同的标签值组成一个子表
- 改造效果
  - 稳定性方面：完成改造后，大数据监控平台**摆脱了对大数据组件的依赖，有效缩短了数据处理链路**。自上线以来，一直运行稳定，后续我们也将持续观察
  - 写入性能：TDengine的写入性能跟写入数据有较大关系，在根据容量规划完成相关参数调整后，在理想情况下，集群**写入速度最高达到90w条/**s的写入速度。在通常情况下（存在新建表，混合插入），写入速度在20w条/s
  - 查询性能：在查询性能方面，在使用预计算函数情况下，**查询p99都在0.7秒以内**，已经能够满足我们日常绝大部分查询需求；在做大跨度（6个月）非预计算查询情况下，首次查询耗时在十秒左右，后续类似查询耗时会有大幅下降**（2-3s）**，主要原因是TDengine会缓存最近查询结果，类似查询先读取已有缓存数据，再结合新增数据做聚合
  - 成本方面：**服务端物理机由21台降至3台，每日所需存储空间为93G（2副本），同等副本下仅为OpenTSDB+HBase的约1/10**，在降低成本方面相对通用性大数据平台有非常大的优势

TAOS DATA

客户案例 – 顺丰 （改造前后架构对比）





客户案例 – 理想汽车 （旧方案及痛点）

- 场景：信号上报业务，将标记时间戳和采集点的信息写入后端数据库，有聚合查询需求，7万QPS，预计未来3年达到20万QPS

- 痛点（先是MongoDB，后迁移到TiDB）
  1. 持续高并发写入，带有tag，时间戳有时会乱序；
  2. 对业务有侵入性，底层库表要按照月份来建表
  3. 业务数量级膨胀极快，写入性能不足，需要不断进行资源扩容
  4. 对基于时间戳范围的聚合查询效果较差；
  5. 对硬件配置要求很高

- 期望
  1. 希望可以不用针对月份数据进行分库分表，需求TTL机制；
  2. 希望可以针对采集点单独建表；
  3. 希望支持批量数据写入，且业务期望写入延时较低

客户案例 – 理想汽车 （技术选型）

- TDengine的技术优点
  1. 两级存储结构，数据插入性能高，资源利用率高；
  2. 对时序数据压缩率极高；
  3. 针对采集点单独建表，匹配业务场景；
  4. 支持大批量数据写入；
  5. 无感知的scale-out和scale-in；
  6. 支持TTL

- 迁移方案
  1. 先切写流量到TDengine，历史读流量在TiDB的方案
  2. 逐步将历史数据格式化导入到TDengine

客户案例 – 理想汽车 （效果对比）

对比	TiDB	TDengine	备注
资源配置	共计17个Node：5TiDB Server，3PD server(与TiDB混布)，12个TiKV Server，32C64G SSD(单独部署)	32C128G SSD	除内存加大外，其余配置与TiKV完全一致
资源使用	IO:70% CPU:40% MEM:20%	IO:40% CPU:10% MEM:70%	TDengine：调整分配内存，优化性能，TDengine的设计为内存行存，落盘列存。
业务写入latency	400ms	4-10ms	整体业务延迟降低非常明显
存储量级	3replica-1.3T/month	2replica-87G/month	存储成本降低十几倍
测试极限(相同业务Server写入)	写入速度提高需要TiDB集群增加节点资源	Server端多个K8s Node极限压测，TDengine仍没有到达性能极限	业务压测目前集群已经可以支持20万QPS

使用成本对比表

更多客户案例

电力	国网河北电力调控云平台	华润电力智慧风场大数据平台	绿能青海发电集控大数据平台
	山西电力IDC机房监控系统	青岛日电新能源智慧电厂管理系统	中国电信理想信息电力测功系统
煤炭	华锐风电大数据平台	厦门矿通科技智慧矿上平台	国网电动汽车充电桩联网大数据平台
	山西科达自控省煤炭安全监控平台	华夏天信智慧矿山RED-MOS系统	成都博威煤炭井下设备实时监控系统
车联网	浪潮新汶智慧矿山物联网大数据平台	山东八五信息智慧化工园区物联网平台	发那科数控机床状态监控云平台
	经纬恒润车联网大数据平台	星诺“好管车”车联网平台	科大智能SCARM堆垛机监控服务平台
智慧城市	广联达智慧城市环境监控系统	时代凌宇智慧城市物联网平台	陕西智慧水务大数据平台
	中科惠软南京智慧水务数据平台	辰安智能电梯应急管理物联网平台	金恒科技南钢智能冶金数据平台
制造业	玉溪MES系统	曲靖卷烟厂制丝制丝IO数据平台	昆明卷烟厂制丝IO监控系统
	顺丰IDC运维监控数据平台	得物运维监控数据平台	中国地震台网中心地震数据归档系统
IT运维			