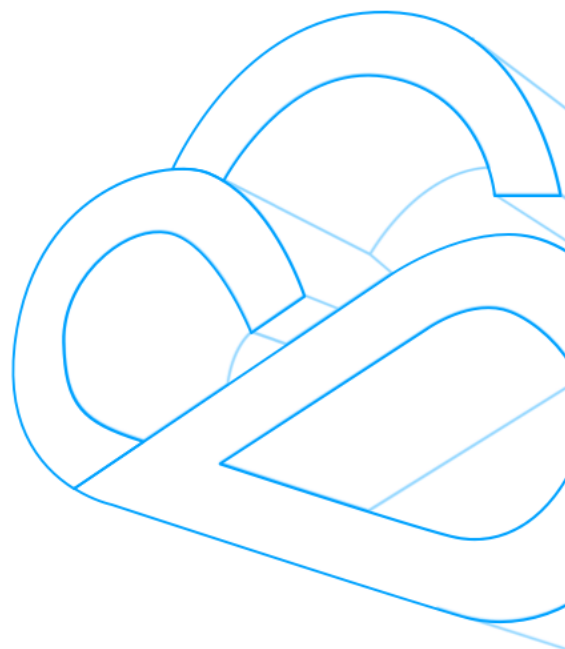




# 腾讯云数据库 TDSQL MySQL 版 分布式 V10.3.22.8/集中式 V8.0.22.8 简介



文档版本：

发布日期：

腾讯云计算（北京）有限责任公司

## 版权声明

本文档著作权归腾讯云计算（北京）有限责任公司（以下简称“腾讯云”）单独所有，未经腾讯云事先书面许可，任何主体不得以任何方式或理由使用本文档，包括但不限于复制、修改、传播、公开、剽窃全部或部分本文档内容。

本文档及其所含内容均属腾讯云内部资料，并且仅供腾讯云指定的主体查看。如果您非经腾讯云授权而获得本文档的全部或部分内容，敬请予以删除，切勿以复制、披露、传播等任何方式使用本文档或其任何内容，亦请切勿依本文档或其任何内容而采取任何行动。

## 商标声明



“腾讯”、“腾讯云”及其它腾讯云服务相关的商标、标识等均为腾讯云及其关联公司各自所有。若本文档涉及第三方主体的商标，则应依法由其权利人所有。

## 免责声明

本文档旨在向客户介绍本文档撰写时，腾讯云相关产品、服务的当时的整体概况，部分产品或服务在后续可能因技术调整或项目设计等任何原因，导致其服务内容、标准等有所调整。因此，本文档仅供参考，腾讯云不对其准确性、适用性或完整性等做任何保证。您所购买、使用的腾讯云产品、服务的种类、内容、服务标准等，应以您和腾讯云之间签署的合同约定为准，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

# 修订记录

文档版本	发布日期	修订人	修订内容
------	------	-----	------

# 目录

---

修订记录.....ii

目录.....iii

前言.....iv

1 产品概述..... 1

1.1 演进历史..... 1

1.2 核心特性..... 3

1.3 应用场景..... 4

2 产品架构..... 6

2.1 系统架构和模块说明..... 6

2.2 集中式和分布式的架构特性及异同..... 7

3 内核功能概览..... 9

4 使用限制说明..... 11

# 前言

## 文档目的

本文档用于帮助用户掌握云产品的操作方法与注意事项。





## 目标读者

本文档主要适用于如下对象群体：

- 客户
- 交付 PM
- 交付技术架构师
- 交付工程师
- 产品交付架构师
- 研发工程师
- 运维工程师

## 符号约定

本文档中可能采用的符号约定如下：

符号	说明
 说明：	表示是正文的附加信息，是对正文的强调和补充。
 注意：	表示有低度的潜在风险，主要是用户必读或较关键信息，若用户忽略注意消息，可能会因误操作而带来一定的不良后果或者无法成功操作。
 警告：	表示有中度的潜在风险，例如用户应注意的高危操作，如果忽视这些文本，可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
 禁止：	表示有高度潜在危险，例如用户应注意的禁用操作，如果不能避免，会导致系统崩溃、数据丢失且无法修复等严重问题。

# 1 产品概述

---

## 1.1 演进历史

### 分布式数据库发展简史

分布式数据库理论成型于 20 世纪 70 年代，当时集中式数据库刚刚从理论向业务系统落地，从在 1990 年以前，分布式数据库尚处于理论阶段，并无大型实际应用。而 1991 年由 M. Tamer Özsu, Patrick Valduriez 两位学者的撰写的《Principles of Distributed Database Systems》，则是分布式数据库理论和实践第一次系统性的介绍。

随着 20 世纪 90 年代，中美互联网产业的快速发展，各个互联网厂商将分布式数据库逐步应用于真实的业务系统，进而彻底改变了分布式数据库主要存在于理论阶段的命运。

首先，互联网产业的快速发展完全改变了原有应用系统的使用场景——从仅服务于公司内部员工转变为服务大众，后者用户是前者的几万倍，且呈现指数级快速增长趋势。以腾讯、阿里、Amazon.com（亚马逊）、Yahoo.com（雅虎）等互联网先驱为例，大多数应用在推出后一或两年后就能达到百万或千万级用户量，后期业务量更是呈指数级增长。因此传统面向企业内部应用的集中式数据库无法满足互联网产业的性能和容量需求。

其次，当时全球数据库市场被 Oracle、IBM 和微软占据，虽然其商用数据库提供了诸多可靠的解决方案，但是众多互联网厂商当时缺乏可靠的盈利来源，很难去购买昂贵的商用数据库。绝大多数互联网厂商从成本考虑，选择基于开源数据库为技术为基础（当时唯一可靠选择是 MySQL，PostgreSQL 在 1995 年才刚开源），并使用手工方式拆分数据，分散压力（如下图）；这就是产业对分布式数据库使用的雏形。

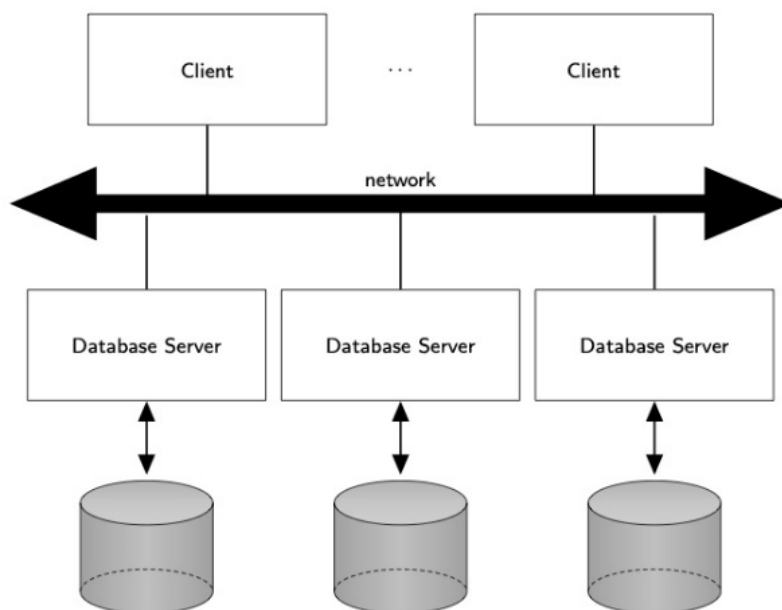


图 1.1-1 图片来自于《Principles of Distributed Database Systems》

然后，2000 年前后（虽然 2000 年互联网泡沫破裂，但仍有大量的互联网公司存活下来），互联网公司意识到服务的高效、稳定是获取用户的重要手段。但基于手工方式拆分数据的方式带来的是极其复杂的程序逻辑和繁重的运维。因此，各互联网厂商纷纷研究更加高效的分布式数据库方案（基于 MDBS Layer 方式，如下图），如早期的 Google 的 BigTable、Yahoo 的 PNUTS、阿里 TDDL、腾讯的 HOLD、开源的 MyCat。而随着分布式数据库实际应用的广泛和深入，业务的发展，《Principles of Distributed Database Systems》第二版出版，这本书的更新也意味着分布式数据库迎来新纪元。

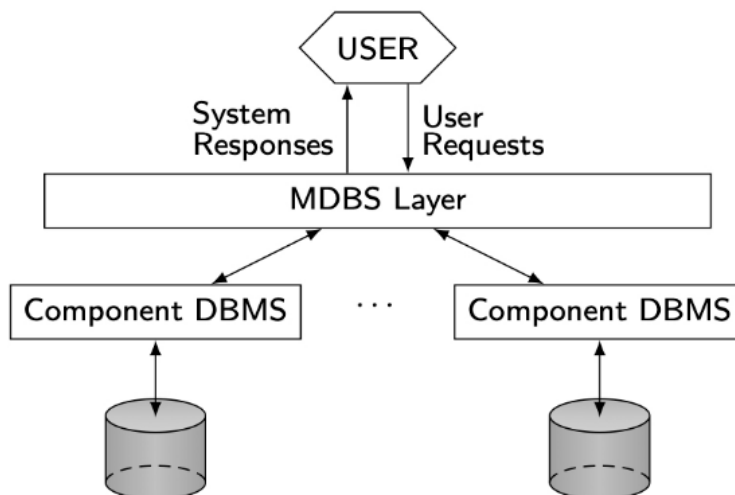


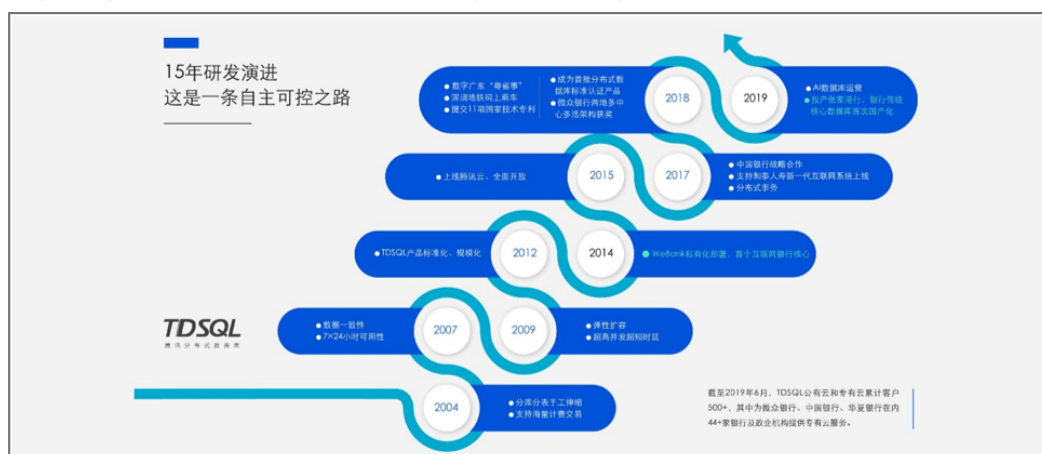
图 1.1-2 图片来自于《Principles of Distributed Database Systems》

而来到 21 世纪，分布式数据库开始迎来他的快速发展期，这期间基于硬件、Web 技术、云计算等快速发展，分布式数据库的应用日益广阔。分布式关系型数据库进一步演进，数据库引入全局事务管理器（GTM）和全局协调器（Keeper），进行分布式事务处理和必要的元数据管理工作，并将 SQL 引擎和存储引擎前置到专门的计算层，进而支持更加复杂的应用场景。同时，基于数据模型的变化，分布式数据库家族也衍生出更多更复杂类型，例如 NOSQL（KV 存储、文档存储、宽列数据库、图数据库）、NewSQL（Spanner、多模数据库）、以及基于全球部署云化的分布式数据服务等。

## TDSQL 发展简史

腾讯云数据库 TDSQL (Tencent Database SQL，简称 TDSQL) 是随着腾讯业务规模不断扩大而发展起来的，其定位是基于互联网分布式架构的金融级数据库。

从 2004 年开始，核心业务便大规模使用分布式架构数据库，经过 15 年的发展，TDSQL 已经覆盖公有云、专有云部署模式，长期为财付通、腾讯充值等业务提供底层支撑，并广泛应用于党政机关、银行、保险、证券等行业，服务客户超过 600。



## 1.2 核心特性

### 超高性能

- 单节点最大性能可达超数十万 QPS，整个实例性能随着分片数量增加线性扩展。
- 计算层每个节点均可读写，单实例轻松支撑千万级 QPS 流量，适合大批量写入的业务场景。

### 专业可靠

- 经过各类客户核心业务超十年的大规模生产系统验证，行业覆盖社交、电商、支付、音视频等多领域。
- 提供完善的数据备份、容灾、一键升级等功能。
- 完善的监控和报警体系，大部分故障都通过自动化程序处理恢复。



- 提供数据加密能力，支持 AES 算法和国密 SM4 算法，可满足静态数据加密的合规性要求。
- 支持分布式数据库领域领先功能，如分布式多表 JOIN、小表广播、分布式事务、SQL 透传等。
- 数据库实例可用性可达到 99.95%；数据的可靠性可达到 99.99999%。

## 简单易用

- 除少量语法与原生 MySQL、MariaDB 不同外，使用起来如使用单机数据库，分片过程对业务透明且无需干预。
- 兼容 MySQL 协议（支持 MySQL、MariaDB 等内核）。
- 支持 Web 控制台，读写分离能力、专有运维管理指令等。

## 1.3 应用场景

### 说明:

TDSQL MySQL 版目前仅适用于 OLTP 场景的业务，例如，交易系统、前台系统；不适用于 ERP、BI 等存在大量 OLAP 业务的系统。

### 大型应用（超高并发实时交易场景）

电商、金融、O2O、社交应用、零售、SaaS 服务提供商，普遍存在用户基数大（百万级或以上）、营销活动频繁、核心交易系统数据库响应日益变慢的问题，制约业务发展。

TDSQL MySQL 版提供线性水平扩展能力，能够实时提升数据库处理能力，提高访问效率，峰值 QPS 达 1500 万+，轻松应对高并发的实时交易场景。微信支付、财付通、腾讯充值等都是使用的 TDSQL MySQL 版架构的数据库。

### 物联网数据（PB 级数据存储访问场景）

在工业监控和远程控制、智慧城市的延展、智能家居、车联网等物联网场景下，传感监控设备多、采样率高、数据规模大。通常存储一年的数据就可以达到 PB 级甚至 EB，而传统基于 x86 服务器架构和开源数据库的方案根本无法存储和使用如此大的数据量。

TDSQL MySQL 版提供水平扩展能力，TDStore 引擎还支持超高压缩比存储，可以有效的帮助用户以低成本存储海量数据。

### 文件索引（万亿行数据毫秒级存取）

一般来说，作为云服务平台，存在大量的图片、文档、视频数据，数据量都在亿级 - 万亿级，服务平台通常需要将这文件的索引存入数据库，并在索引层面提供实时的新增、修改、读取、删除操作。

由于服务平台承载着其他客户的访问，服务质量和性能要求极高。传统数据库无法支撑如此规模的访问和使用，TDSQL MySQL 版超高性能和扩展能力并配合强同步能力，有效的保证平台服务质量和数据一致性。

## 高性价比商业数据库解决方案

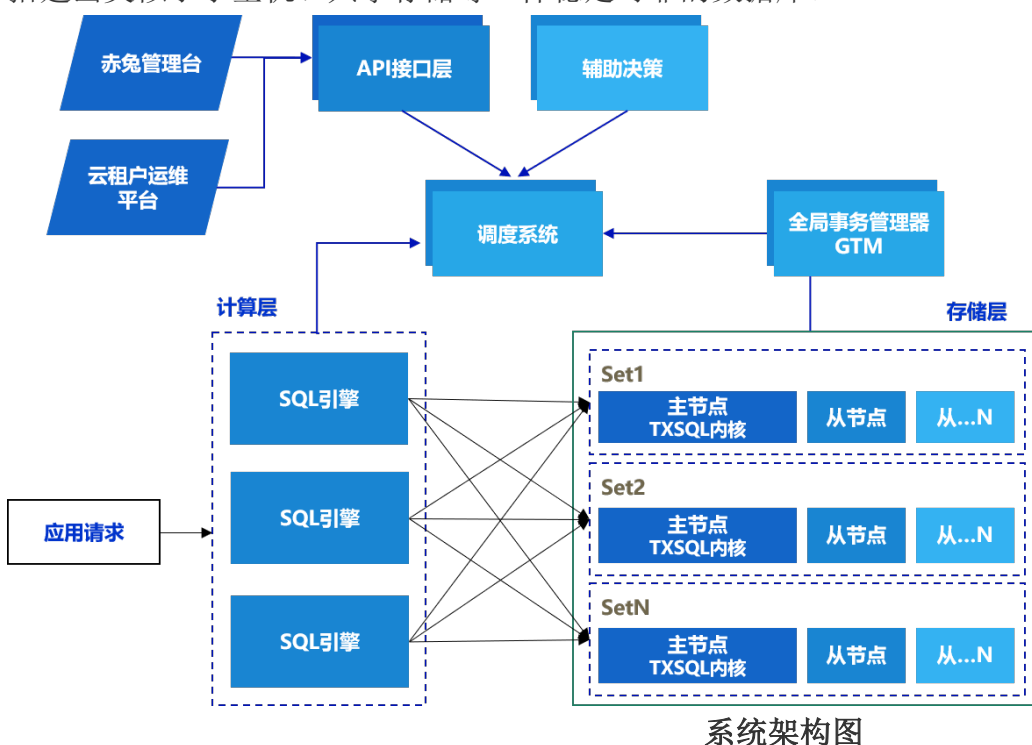
大型企业、银行等行业为了支持大规模数据存储和高并发数据库访问，对小型机和高端存储依赖极强。而互联网企业通过低成本 x86 服务器和开源软件即可达到与商业数据库相同甚至更高的能力。

TDSQL MySQL 版适用于诸如国家级或省级业务系统汇聚、大型企业电商和渠道平台、银行的互联网业务和交易系统等场景。

## 2 产品架构

### 2.1 系统架构和模块说明

TDSQL 采用分布式集群架构，如下图所示。该架构具有较高的灵活性，不仅简化了各个节点之间的通信机制，也简化了对于硬件的需求。使得 TDSQL 的集中式实例、分布式实例、分析型实例不仅可以混合部署在同一集群中，而且即使配置简单的 x86、ARM 服务器，也可以搭建出类似于小型机、共享存储等一样稳定可靠的数据库。



数据库存储层、数据库计算层形成了 TDSQL 集群最核心的部分，除此之外还包括其他系统。

- 数据库存储层，是由数据库节点组（SET）组成。
  - 数据库存储层属于 IO 密集型的服务，通常对磁盘性能要求较高；如果在虚拟化情况下，一台物理机可能会存在多个实例的读写，因此我们建议配置较高性能的磁盘。
  - SET 存储数据库核心数据，采用主从高可用（HA）架构，节点数通常 $\geq 2$ ，根据业务实际需求可扩展。
  - SET 是一种逻辑概念，可以由一组物理设备（物理的 SET），也可以是在物理设备上虚拟出的节点组成（虚拟的 SET）。

- 每个节点（Node）信息采集与管理模块（Agent），互相检测心跳以确保集群的健壮性。
- 数据库计算层，是由 SQL 引擎（SQL-Engine，又名 Proxy）组成。
  - SQL 引擎属于 CPU 密集型，内存消耗性。这是因为 SQL 引擎作为计算层，它需要管理链接，如果是大量的短链接或者长链接，非常占内存；如果存在复杂 SQL，它又要进行组合计算，所以它对 CPU 和内存的要求比较高。
  - SQL 引擎需要存储和处理数据库元数据，需要负责做词法、语法分析，以及作为查询引擎等工作，在分布式的场景下，SQL 引擎复杂的功能性就会凸显，比如要处理分布式事物，还要维护全局自增字段，保证多个数据、多个存储节点共享一个保证全局自增的序列。还要限制一些非法语法（SQL 防火墙），包括词法和语法的解析；在复杂计算上，它还要做一些 SQL 下推，以及最后数据的聚合。
  - SQL 引擎是无状态的，本身不存储数据，也没有主备之分可以同时与业务系统通讯。
  - SQL 引擎为主主高可用架构，节点数通常 $\geq 2$ ，根据业务实际需求可扩展。
  - 每个 SQL 引擎的 IP 不同，前端通常需要部署负载均衡，可以支持 F5、腾讯 CLB、腾讯 TGW、LVS、DNS 等；通过负载均衡为分布式或集中式实例提供唯一的访问地址。
- 配置管理决策调度集群，是由 Zookeeper、Scheduler、Manager、GTM（Metacluster）等组件组成。
  - 配置管理决策调度集群通常需要奇数台，基于高可用通常建议 3 台起配，以基于 raft 协议实现对实例高可用切换的第三方选举。
  - GTM（Metacluster）：提供分布式事务全局一致性的事务管理器，主要由 Metacluster 中心时钟源为事务的开启、提交阶段提供一个全局唯一且严格递增的事务时间戳以及事务管理器（TM）组成，为了提高并发效率 TM 模块目前内置于计算层。
- 运维支撑系统，是由 OSS 运营调度管理模块、监控采集与分析模块、赤兔管理系统组成。
- 其他支撑系统。

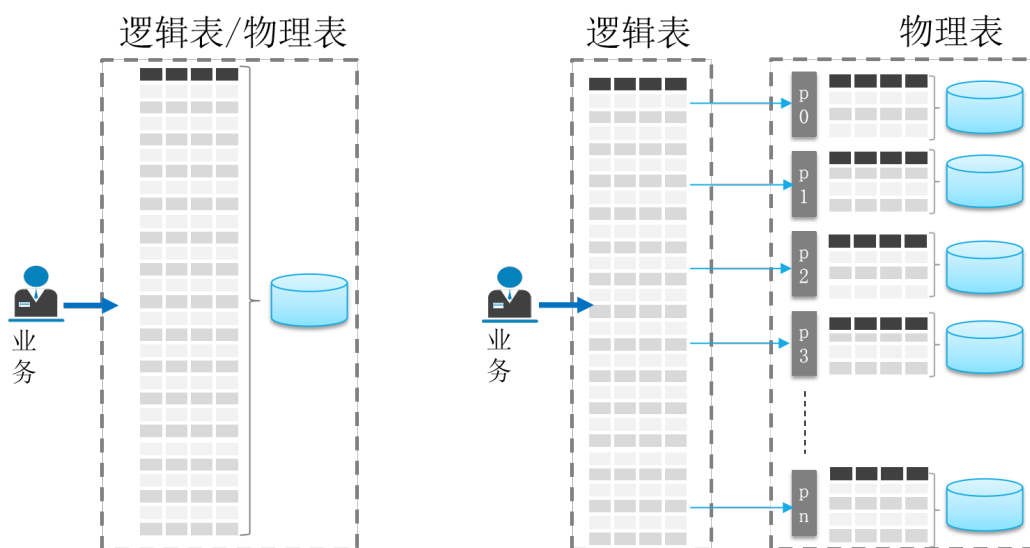
基于上述架构，分布式实例和集中式实例可以有机的结合在一套物理集群中，实现混合部署：

- 一个分片（SHARDING，一组独立的数据库计算节点和数据库存储节点组成），相当于一个集中式数据库实例。
- 多个分片，即组成（group）分布式数据库实例。

## 2.2 集中式和分布式的架构特性及异同

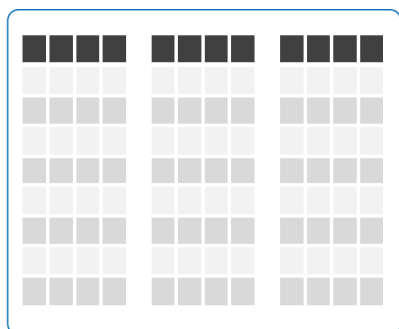
集中式实例和分布式实例异同：

- 在集中式（NoShard）模式下，一张库表分布在一个 MySQL 实例上。
- 在分布式（Shard）模式下，一张表根据分片的数量分布在不同的 MySQL 节点上。

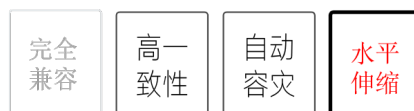
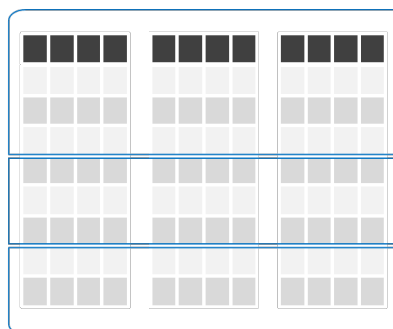


- 架构上都支持金融级别的容灾和数据一致性保障等要求。
- 功能上 NoShard 完全兼容 MySQL，Shard 模式下对 SQL 能力有一些约束。

NoShard



Shard



- 中小规模适用于 NoShard，大规模适用于 Shard。



## 3 内核功能概览

---

- 引入线程池，创建一定数量的服务线程服务所有的用户请求，确保大量用户连接的增加数据库性能也能维持在高位。
- 支持强同步复制，为了满足金融级用户对数据库可用性要求，在主从同步高可用架构上实现主从复制的强一致性，只有在从库成功复制了主库事务产生的日志后主库才能给客户端返回成功。
- 支持热点更新功能，在引擎层实现的热点更新模块采用排队的思想对业务更新操作串行化，避免了行锁冲突，能大幅度提升性能。
- 移除 InnoDB 表级锁，InnoDB 引擎自身包含了一套表级锁功能，但同时在 Server 层又有 MDL 锁来保护表和库等元数据信息，InnoDB 的表级锁不仅冗余，还产生了更高的锁系统冲突，因此 TXSQ 将其移除，从而获得性能提升。
- 实现无锁化的事务系统。
- 支持将 client ip 透传到 DB。
- InnoDB 使用独立的 Purge 线程来对 undo log 及废弃的索引记录进行清理，单个线程的效率在高负载下较低，TXSQ 实现多 Purge 线程调度无锁化优化。
- 支持 InnoDB 事务锁系统分区，优化 DML 场景下，mutex 的激烈冲突，提升性能。
- 优化 fil\_write\_zeros 分配/释放大内存，优化 innodb 在分配新的数据页面的时候的性能。
- group commit 三阶段优化，事务提交需要经过三个阶段（write binlog,flush,innodb commit），将这个过程拆成三个阶段，使用三个队列来实现，提升 IO 效率。
- mysqlbinlog 支持 flashback，利用 binlog 来进行数据恢复。
- 新增 murmurhashcodeandmode 函数，用来支持 shard 模式下分区表的创建。
- 新增屏蔽分区功能，用来实现水平扩容快速对不需要被访问到的分区进行屏蔽。
- 新增语法 xa rollback " force;强制回滚当前链接的 xa 事务。
- 新增 txsql\_skip\_xa\_recover\_admin 参数，实现所有用户都能执行 xa recover，方便实现分布式事务。
- 新增 pseudo\_server\_id 会话变量，将 binlog 用指定的 server id 去写，实现跨城容灾 binlog 不环绕。
- 新增 xa recover with time;语法，打印分布式事务的开始时间。
- 新增一系列参数，用于对云上用户各种高危敏感操作进行限制。
- 增加异步删除大表功能，主要用于删除数据文件很大的表，避免 IO 的抖动。
- 增加 binlog 复制限速功能，主要用于限制 binlog 复制的速度，避免主从差距过大。
- 增加状态变量 Iothreadreport\_newstime，用来上报 IO 线程的延迟。

- 增加 `default_collation_for_utf8mb4`，为 `utf8mb4` 配置默认的字符序。
- 增加 `show detail processlist` 语法，展示更多的会话信息。
- 通过插件，提供 SQL 防火墙的支持。
- 通过设置带内存池的 `history`，优化在设置 `binlog_transaction_dependency_tracking = WRITESET` 的情况下，`tpcc` 性能提升 30%。
- 支持国密算法。
- 支持列压缩。
- 支持企业级审计功能，主要用于记录用户各种指定类型的操作。
- 支持 `txsql_returning` 语法，使得 `update+select` 通过一条 SQL 就可以完成，提升执行效率。
- 支持 GB18030-2022 字符集。
- 支持二级分区功能。
- 支持执行计划缓存功能。
- 支持轻量级 `check` 约束以及自增序列的特性。
- 支持 MDL 死锁信息记录到 `errorlog`，可以在 `mdl` 死锁发生的时候记录下来。
- 支持 `sequence` 功能增强。
- 支持全局索引功能。
- 支持回收站功能。
- 支持全局一致性读。
- 支持 `Profile` 记录到慢日志功能。
- 兼容部分 `oracle` 常用的语法和函数，比如 `add_month number()`、`to_number()` 内置函数等。

## 4 使用限制说明

### 说明:

TDSQL MySQL 版支持分布式实例和集中式实例。其中集中式实例使用可以参考社区 MySQL 语法规则。分布式实例也高度兼容 MySQL 的协议和语法，但由于架构的差异，对于 SQL 有一定的限制，具体限制说明如下。

### 大特性限制

- 不支持自定义函数、事件、表空间。
- 不支持存储过程、触发器、游标。
- 不支持外键、自建分区。
- 不支持复合语句，如 BEGIN END、LOOP。

### 小语法限制

#### DDL

- 不支持 CREATE TABLE ... SELECT
- 不支持 CREATE TEMPORARY TABLE
- 不支持 CREATE/DROP/ALTER SERVER/LOGFILE GROUP
- 不支持 ALTER 对分表键（shardkey）进行改名。

#### DML

- 不支持 SELECT INTO OUTFILE/INTO DUMPFILE/INTO var\_name
- 不支持 query\_expression\_options，如：  
HIGH\_PRIORITY/STRAIGHT\_JOIN/SQL\_SMALL\_RESULT/SQL\_BIG\_RESULT/SQL\_BUFFER\_RESULT/SQL\_CACHE/SQL\_NO\_CACHE/SQL\_CALC\_FOUND\_ROWS。
- 不支持不带列名的 INSERT/REPLACE
- 不支持全局的 DELETE/UPDATE 使用 ORDER BY/LIMIT（版本 $\geq$ 1.14.4 支持）
- 不支持不带 WHERE 条件的 UPDATE/DELETE
- 不支持 LOAD DATA/XML
- 不支持 SQL 中使用 DELAYED 和 LOW\_PRIORITY，没有效果。
- 不支持 INSERT ... SELECT（版本 $>$ 1.14.4 支持）
- 不支持 SQL 中对于变量的引用和操作，如：SET @c=1, @d=@c+1; SELECT @c, @d
- 不支持 index\_hint
- 不支持 HANDLER/DO

### 管理 SQL



- 不支持 ANALYZE/CHECK/CHECKSUM/OPTIMIZE/REPAIR TABLE，需要用透传语法。
- 不支持 CACHE INDEX
- 不支持 FLUSH
- 不支持 KILL
- 不支持 LOAD INDEX INTO CACHE
- 不支持 RESET
- 不支持 SHUTDOWN
- 不支持 SHOW BINARY LOGS/BINLOG EVENTS
- 不支持 SHOW WARNINGS/ERRORS 和 LIMIT/COUNT 的组合。
- 不支持 SHOW GRANTS FOR CURRENT\_USER 语句，需使用 SHOW GRANTS FOR <用户名称>进行查询。