TDSQL MySQL版

产品简介

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2025 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】

腾讯云

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。
您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，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。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

文档目录

产品简介
- 产品概述
- 产品优势
- 应用场景
  - InnoDB 引擎
  - TDStore 引擎
- 地域和可用区
- InnoDB 引擎
  - 实例架构
  - 基本原理
    - 水平分表
    - 读写分离
    - 弹性扩展
    - 强同步
- TDStore 引擎
  - 产品特性
  - 实例架构

版权所有：腾讯云计算（北京）有限责任公司

产品简介
产品概述

最近更新时间：2025-02-28 10:19:32

简介

TDSQL MySQL 版（TDSQL for MySQL）是部署在腾讯云上的一种支持自动水平拆分、Shared Nothing 架构的分布式数据库。TDSQL MySQL 版即业务获取的是完整的逻辑库表，而后端会将库表均匀的拆分到多个物理分片节点。
TDSQL MySQL 版默认部署主备架构，提供容灾、备份、恢复、监控、迁移等全套解决方案，适用于 TB 或 PB 级的海量数据库场景。
TDSQL MySQL 版提供不同的引擎供用户选择，两者均兼容 MySQL 标准协议：

- **InnoDB 版**：采用 InnoDB 作为数据存储引擎，是 MySQL 的默认存储引擎。
- **TDStore 版**：采用自研新敏态引擎 TDStore 作为数据存储引擎，该引擎可以有效解决客户业务发展过程中业务形态、业务量的不可预知性，适配金融敏态业务。

解决问题

单机数据库瓶颈

面对互联网类业务百万级以上的用户量，单机数据库由于硬件和软件的限制，数据库在数据存储容量、访问容量、容灾等方面都会随着业务的增长而到达瓶颈。
如果性能或容量不足以支撑业务发展时，TDSQL MySQL 版可通过控制台一键扩容。整个过程中您无需关心分布式系统内的数据迁移，均衡和路由切换。升级完成后访问 IP 不变，仅在自动切换时存在秒级闪断，您仅需确保有重连机制即可。

应用层分片开发工作量大

应用层分片将业务逻辑和数据库逻辑高度耦合，给当前业务快速迭代带来极大的开发工作量。
基于 TDSQL MySQL 版透明自动拆分的方案，开发者只需要在 建表 时指定分表键（shardkey）的值，后续迭代无需过多关注数据库逻辑，可以极大减少开发工作量。

开源方案或 NoSQL 难题

选择开源或 NoSQL 产品也能够解决数据库瓶颈，这些产品免费或者费用相对较低，但可能有如下问题：

- 产品 bug 修复取决于社区进度。
- 您的团队是否有能持续维护该产品的人，且不会因为人事变动而影响项目。
- 关联系统是否做好准备。
- 您的业务重心是什么，投入资源来保障开源产品的资源管控和生命周期管理、分布式逻辑、高可用部署和切换、容灾备份、自助运维、疑难排查等是否是您的业务指标。

TDSQL MySQL 版支持 Web 控制台，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。

版权所有：腾讯云计算（北京）有限责任公司

产品优势

最近更新时间：2025-06-11 14:44:52

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。
- TDStore 引擎支持超高压缩比存储，最高可达20倍压缩率，尤其适合海量数据的业务场景。

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%。
- 产品通过 安全可靠测评 ，为客户提供完整可靠的分布式数据库解决方案。

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。

版权所有：腾讯云计算（北京）有限责任公司

应用场景
InnoDB 引擎

最近更新时间：2024-12-17 10:45:42

> **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些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。

版权所有：腾讯云计算（北京）有限责任公司

TDStore 引擎

最近更新时间：2024-12-17 11:05:22

> **说明：**
> TDSQL TDStore 引擎版本仅适用于 OLTP 场景的业务。

历史归档

在电商、零售、金融、O2O、社交应用等行业中，随着时间的推移，大量历史数据会与最新数据一起存储在 MySQL 数据库中。这种冷数据和热数据混合存储会带来存储成本高、数据库性能下降和备份困难等问题。借助 TDStore 引擎的海量存储、低成本、高可靠、强扩展等特性，用户能够以极低成本实现大容量数据归档存储方案时，也支持事务处理和在线数据访问。腾讯通信充值、信用卡还款记录等业务都已经在使用 TDSQL TDStore 引擎。

海量数据

传统数据库在存储和管理海量数据时面临诸多挑战，尤其是在高并发读写和高吞吐场景下。在日志流水、物联网、智能家居等海量场景中，数据规模大、单表记录多、写入吞吐高、一致性要求高是常见的业务需求。TDStore 引擎具备良好的水平扩展能力，可以通过增加节点近乎线性地提升系统的读取和写入的吞吐和处理能力，支持数百 TB 或更多的数据处理。同时，引擎内置 Multi-Raft 协议，提供强一致和高可用性，并具备快速在线 DDL 和弹性伸缩能力，使用户可以稳定、高效地存储和管理海量数据。此外，TDStore 引擎还提供高效的数据压缩和存储策略，有效降低了存储成本。腾讯发票流水业务已经使用了 TDSQL TDStore 引擎。

敏态业务

随着现代市场需求越来越多变，业务研发越来越敏捷，灵活的 Schema 和不受限的弹性伸缩能力对游戏、SaaS 等行业的用户尤其重要。TDStore 引擎为应对敏态业务需求提供了强有力的支持，其强大弹性伸缩能力使其能够在需要时迅速扩展或缩减资源，既能支持水平伸缩，也能支持纵向伸缩，还能支持副本伸缩。同时，TDStore 引擎的快速在线 DDL 能力使用户能够在不影响业务运行的情况下，轻松进行数据库结构的调整，从而应对各种难以预料的业务变化，为应对现代复杂业务需求提供了强有力的支持。TDStore 引擎已经在腾讯手Q 游戏中心的用户活动数据场景采用。

数据中台

传统按照核心业务分片键分库分表 MySQL 方案虽然解决了 TP 场景的交易核心问题，但分库分表方案对数据运营或数据汇聚场景（Data Serving/Data Hub）并不友好。这些场景往往并不按照分片键进行查询，而是按照多个不同维度进行数据轻量统计分析。此时，无论广播查询还是通过索引表都难以简单、灵活、高效的实现多维查询，而 TDStore 引擎则非常适合此类场景。TDStore 引擎支持将多个分片的数据高速汇总到同一实例，支持创建全局唯一索引，支持多分片多维度数据的一致性查询，满足数据中台场景下需要的高性能的读写能力，并支撑用户基于海量数据做数据运营分析。TDStore 引擎已经被腾讯广告轻量数仓场景采用。

版权所有：腾讯云计算（北京）有限责任公司

地域和可用区

最近更新时间：2024-05-22 14:19:31

InnoDB 引擎

公有云

腾讯云目前提供多个可选地域，TDSQL MySQL 版支持的地域和可用区可在 TDSQL MySQL 版购买页 查看。

金融云

针对金融行业监管要求定制的合规专区，具有高安全，高隔离性的特点；已认证通过的金融行业客户可提工单申请使用专区，详见 金融专区介绍 。

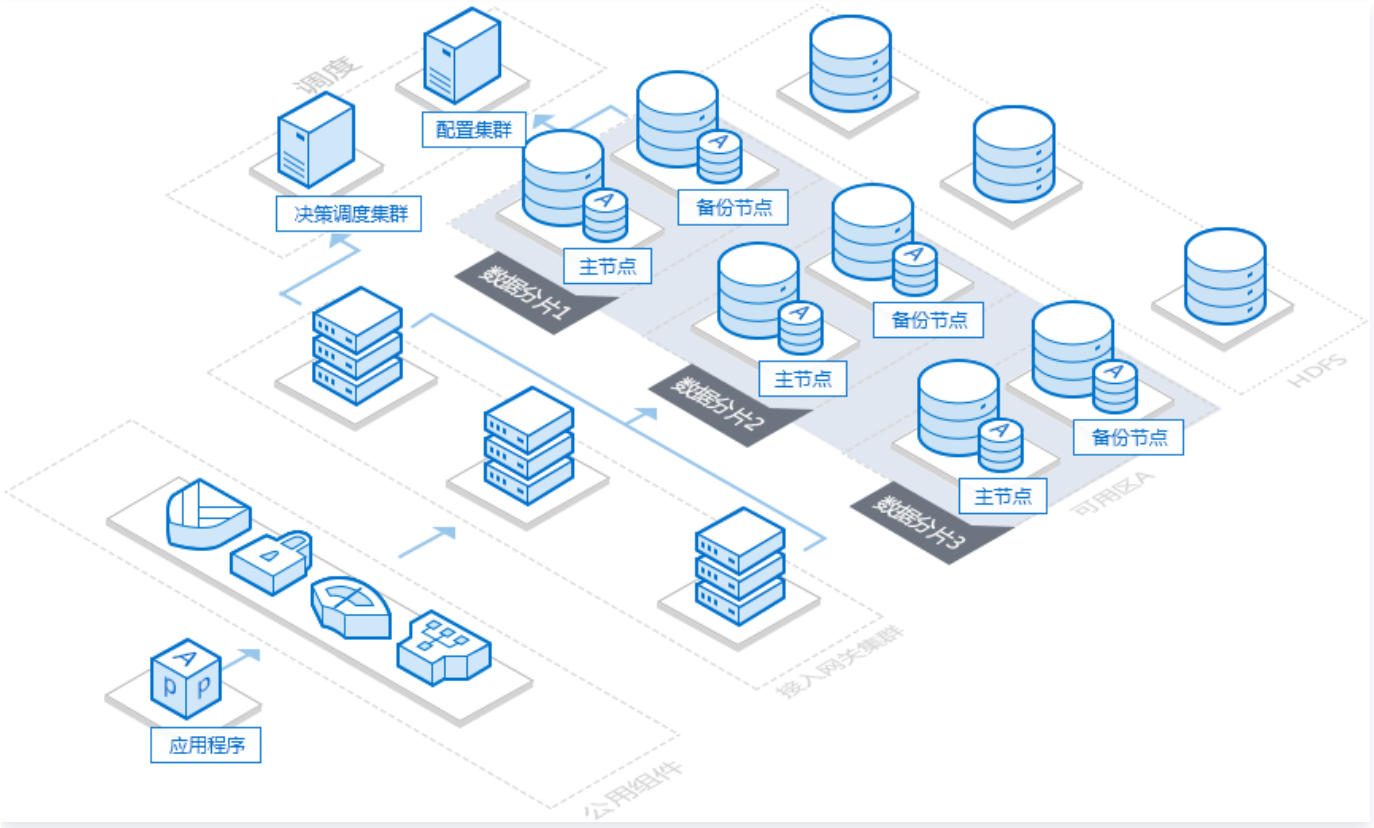
TDStore 引擎

目前支持公有云华南（广州）、华北（北京）、华东（上海）地域。金融云专区稍后开放支持，敬请期待。

InnoDB 引擎
实例架构

最近更新时间：2024-05-22 15:12:11

InnoDB 引擎



注意：
数据库审计功能重构升级中，敬请期待；在此期间数据库新购实例不再开放审计功能。

实例架构	定义	节点	特点
标准版（一主一从）	每个分片提供主从双活部署的高可用架构	两个节点：一个 Master 节点，一个 Slave 节点	支持从机只读 **说明：** 一主一从架构从机只读仅可用于轻量只读任务，请避免大事务等较高负载任务，影响从机备份任务及可用性。 故障后节点自动恢复 默认监控采样粒度：5分钟/次 最大可配备份时长：7天 操作日志备份：7天
标准版（一主二从）	每个分片提供主从多活部署的高可用架构	三个节点：一个 Master 节点，两个 Slave 节点	

			支持数据库审计，审计日志存储15天，规则配置个数暂无限制
金融定制版（一主一从）	每个分片提供主从双活部署的高可用架构	两个节点：一个 Master 节点，一个 Slave 节点	• 支持其他部署方案，需联系对应商务 • 支持从机只读，从机只读时智能负载 • 故障后节点自动恢复 • 默认监控采样粒度：1分钟/次 • 最大可配备份时长：3650天，需 提交工单 • 操作日志备份：默认60天，归档存储1年 • 支持数据库审计：审计日志存储15天 • 监管配合：有
金融定制版（一主二从）	每个分片提供主从多活部署的高可用架构	三个节点：一个 Master 节点，两个 Slave 节点	

基本原理
水平分表

最近更新时间：2023-02-21 11:41:08

概述

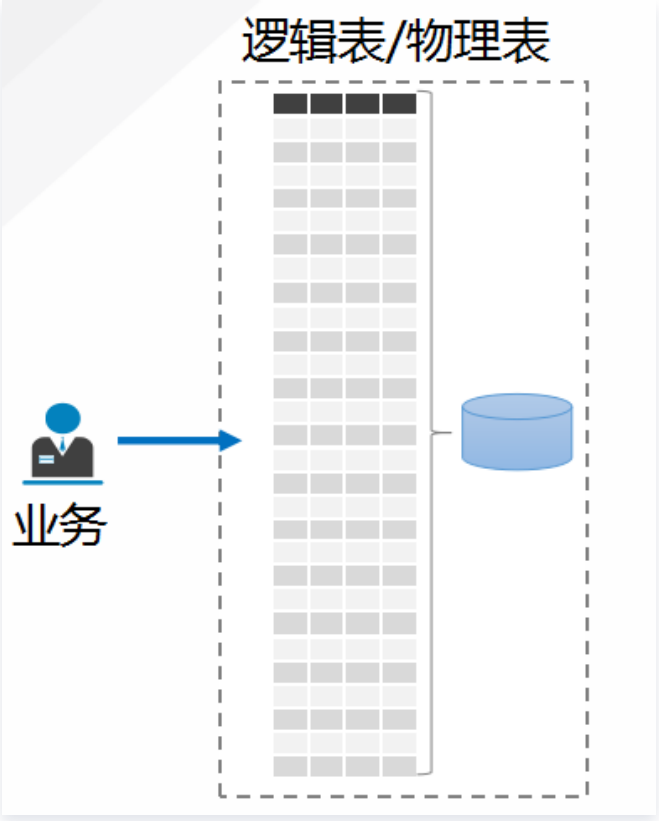
水平拆分方案是 TDSQL MySQL版 的基础原理，它的每个节点都参与计算和数据存储，且每个节点都仅计算和存储一部分数据。因此，无论业务的规模如何增长，我们仅需要在分布式集群中不断的添加设备，用新设备去应对增长的计算和存储需要即可。通过如下视频，您可以了解水平拆分的过程与原理：

观看视频

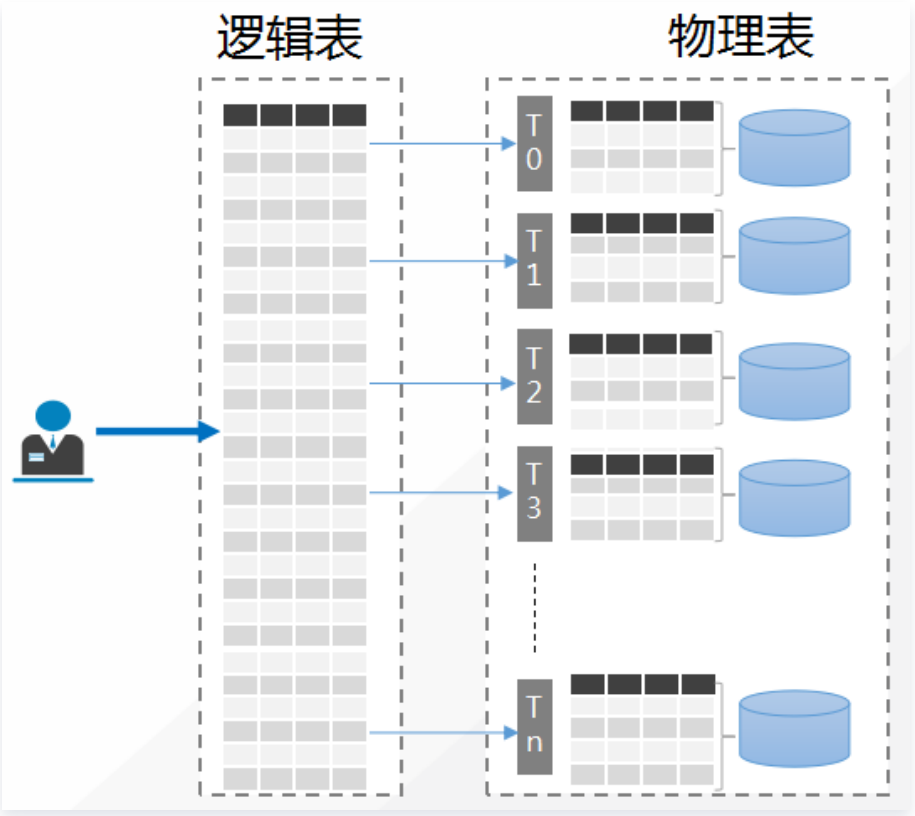
水平切分

水平切分（分表）：是按照某种规则，将一个表的数据分散到多个物理独立的数据库服务器中，形成“独立”的数据库“分片”。多个分片共同组成一个逻辑完整的数据库实例。

- 常规的单机数据库中，一张完整的表仅在一个物理存储设备上读写。



- 分布式数据库中，根据在建表时设定的分表键，系统将根据不同分表键自动分布数据到不同的物理分片中，但逻辑上仍然是一张完整的表。



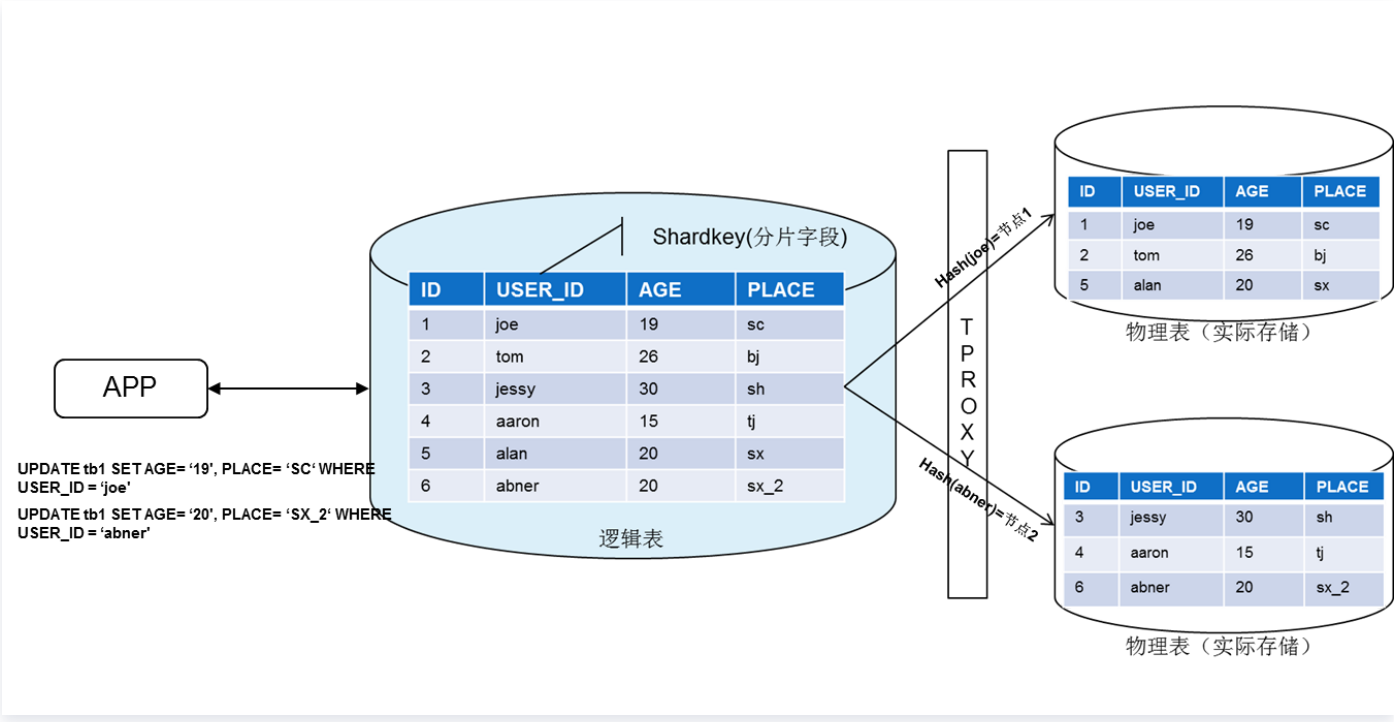
- 在 TDSQL MySQL版 中，数据的切分通常就需要找到一个分表键（shardkey）以确定拆分维度，再采用某个字段求模（HASH）的方案进行分表，而计算 HASH 的某个字段就是 shardkey。HASH 算法能够基本保证数据相对均匀地分散在不同的物理设备中。

写入数据（ SQL 语句含有 shardkey ）

1. 业务写入一行数据。
2. 网关对 shardkey 进行 hash，得出 shardkey 的 hash 值。
3. 不同的 hash 值范围对应不同的分片（调度系统预先分片的算法决定）。

版权所有：腾讯云计算（北京）有限责任公司

4. 数据根据分片算法，将数据存入实际对应的分片中。



数据聚合

数据聚合：如果一个查询 SQL 语句的数据涉及到多个分表，此时 SQL 会被路由到多个分表执行，TDSQL MySQL版 会将各个分表返回的数据按照原始 SQL 语义进行合并，并将最终结果返回给用户。

> **注意**
> 执行 SELECT 语句时，建议您在 where 条件带上 shardKey 字段，否则会导致数据需要全表扫描然后网关才对执行结果进行聚合。全表扫描响应较慢，对性能影响很大。

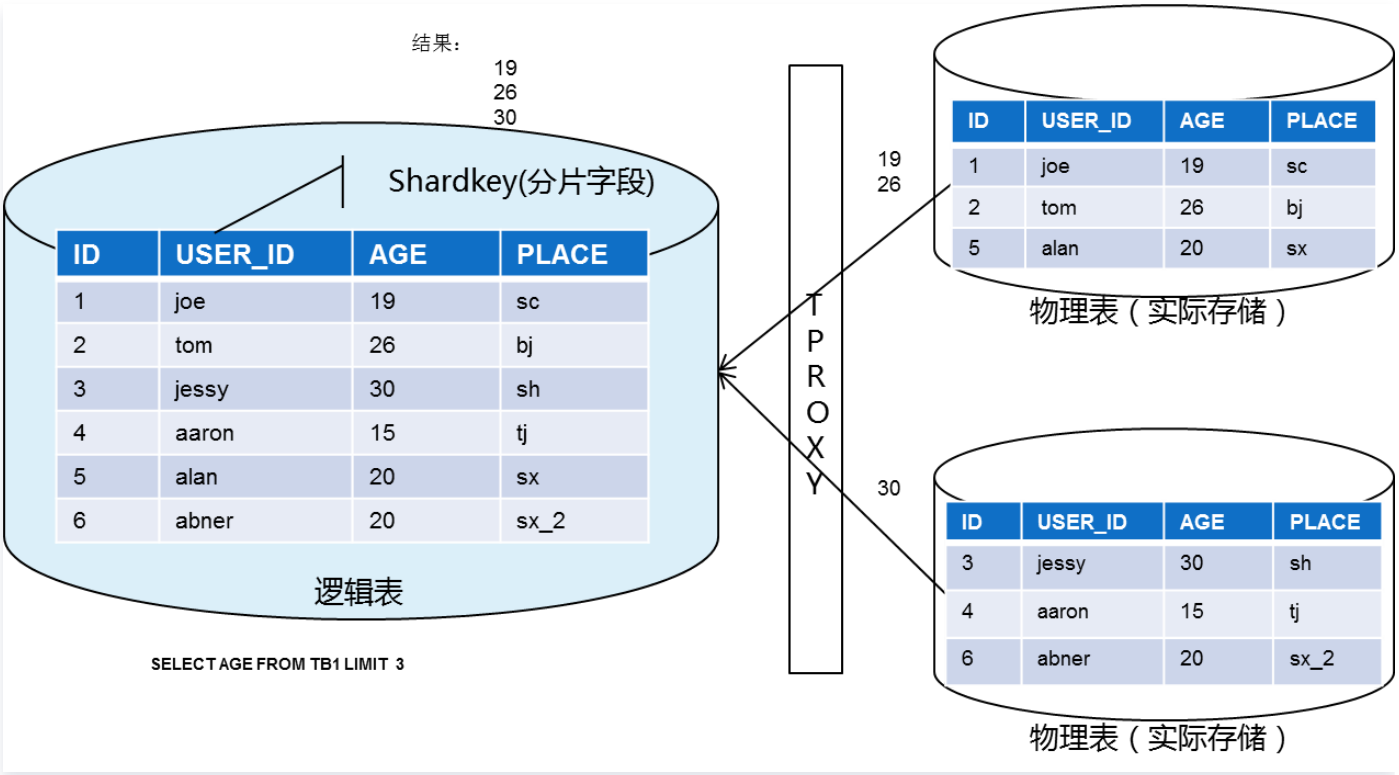
读取数据（有明确 shardkey 值）

1. 业务发送 select 请求中含有 shardkey 时，网关通过对 shardkey 进行 hash。
2. 不同的 hash 值范围对应不同的分片。
3. 数据根据分片算法，将数据从对应的分片中取出。

读取数据（无明确 shardkey 值）

1. 业务发送 select 请求没有 shardkey 时，将请求发往所有分片。
2. 各个分片查询自身内容，发回 Proxy 。

3. Proxy 根据 SQL 规则，对数据进行聚合，再答复给网关。



读写分离

最近更新时间：2024-05-17 12:56:01

功能简介

当处理大数据量读请求的压力大、要求高时，可以通过读写分离功能将读的压力分布到各个从节点上。
TDSQL MySQL 版默认支持读写分离功能，架构中的每个从机都能支持只读能力，如果配置有多个从机，将由网关集群（TProxy）自动分配到低负载从机上，以支撑大型应用程序的读取流量。

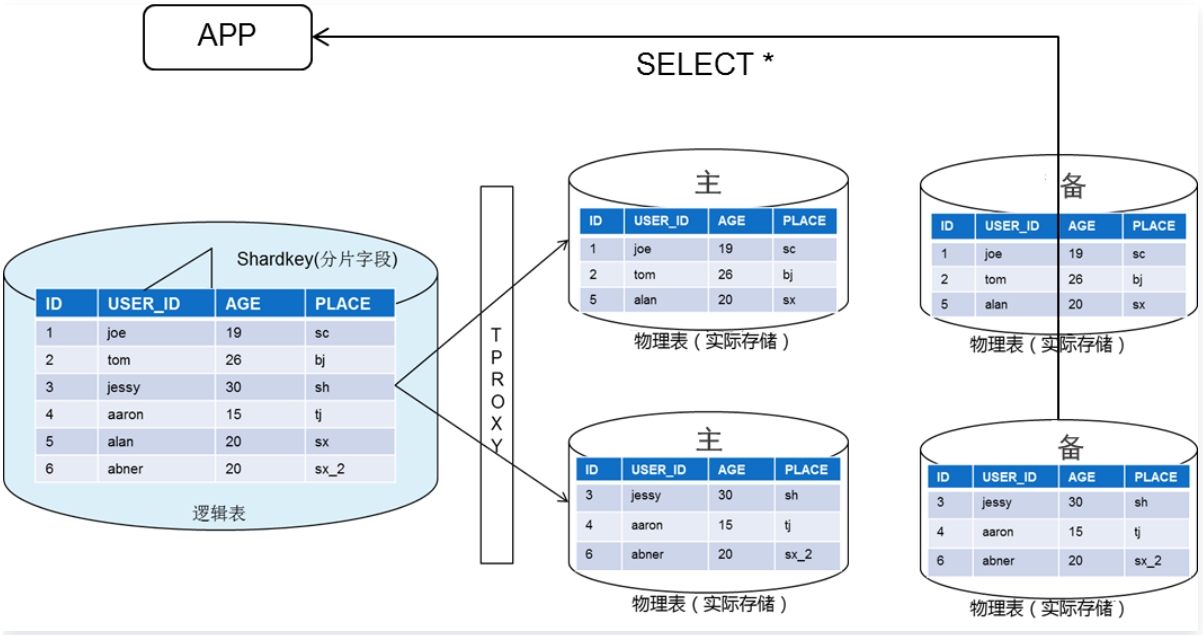
基本原理

读写分离基本的原理是让主节点（Master）处理事务性增、改、删操作（INSERT、UPDATE、DELETE），让从节点（Slave）处理查询操作（SELECT）。

只读账号

> **说明：**
> 若您的实例架构为一主一从，只读分离功能仅可用作低负载只读任务，请避免大事务等较高负载任务，影响从机备份任务及可用性。

只读账号是一类仅有读权限的账号，默认从数据库集群中的从机（或只读实例）中读取数据。
通过只读账号，对读请求自动发送到备机，并返回结果。



弹性扩展

最近更新时间：2022-11-24 11:05:54

概述

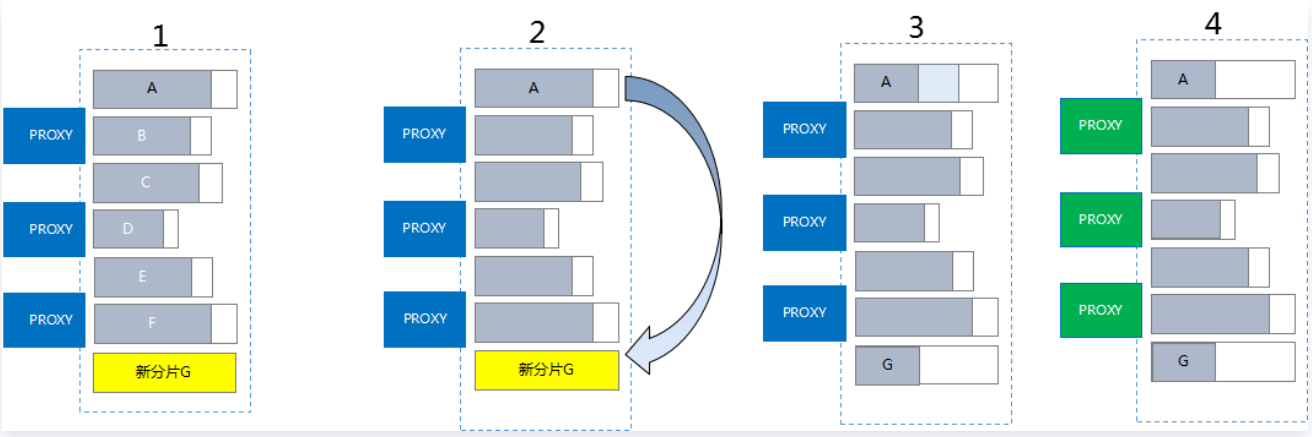
TDSQL MySQL版 支持在线实时扩容，扩容方式分为新增分片和现有分片扩容两种方式，整个扩容过程对业务完全透明，无需业务停机。扩容时仅部分分片存在秒级的只读或中断，整个集群不会受影响。

扩容过程

TDSQL MySQL版 主要是采用自研的自动再均衡技术保证自动化的扩容和稳定。

新增分片扩容

1. 在 TDSQL MySQL版控制台 对需要扩容的 A 节点进行扩容操作。
2. 根据新加 G 节点配置，将 A 节点部分数据搬迁（从备机）到 G 节点。
3. 数据完全同步后，A、G 节点校验数据库，存在一至几十秒的只读，但整个服务不会停止。
4. 调度通知 proxy 切换路由。



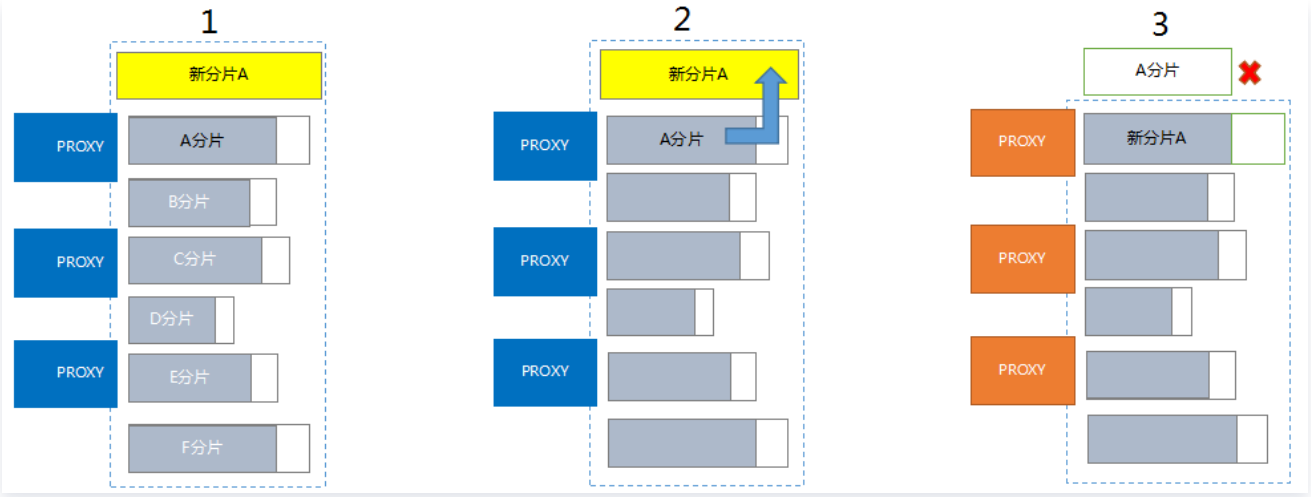
现有分片扩容

基于现有分片的扩容相当于更换了一块更大容量的物理分片。

> **说明**
> 基于现有分片的扩容没有增加分片，不会改变划分分片的逻辑规则和分片数量。

1. 按需要升级的配置分配一个新的物理分片（以下简称新分片）。
2. 将需要升级的物理分片（以下简称老分片）的数据、配置等同步数据到新分片中。

3. 同步数据完成后，在腾讯云网关做路由切换，切换到新分片继续使用。



相关操作

分布式数据库由多个分片组成，如您需要将现有 TDSQL MySQL版 实例的规格升级到更高规格，请参见 升级实例 。

强同步

最近更新时间：2022-08-22 11:31:36

背景

传统数据复制方式有如下三种：

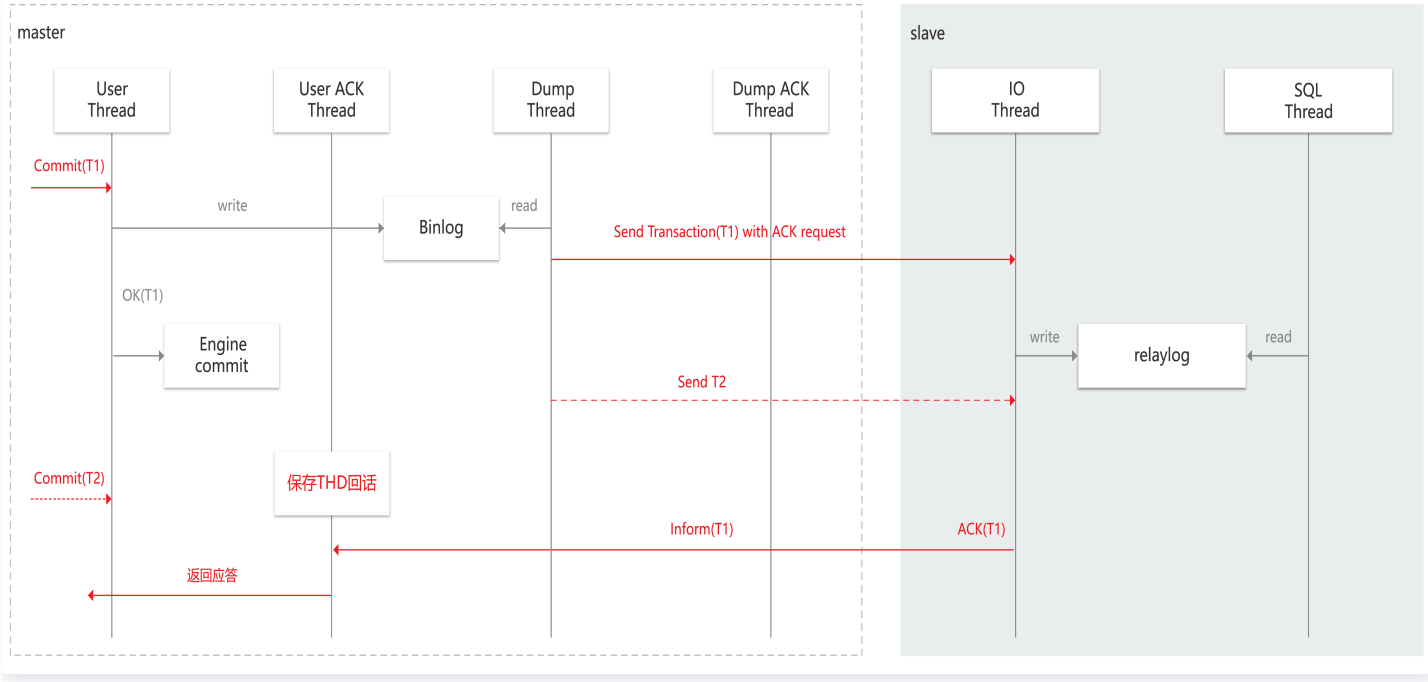
- 异步复制：应用发起更新请求，主节点（Master）完成相应操作后立即响应应用，Master 向从节点（Slave）异步复制数据。
- 强同步复制：应用发起更新请求，Master 完成操作后向 Slave 复制数据，Slave 接收到数据后向 Master 返回成功信息，Master 接到 Slave 的反馈后再应答给应用。Master 向 Slave 复制数据是同步进行的。
- 半同步复制：应用发起更新请求，Master 在执行完更新操作后立即向 Slave 复制数据，Slave 接收到数据并写到 relay log 中（无需执行）后才向 Master 返回成功信息，Master 必须在接受到 Slave 的成功信息后再向应用程序返回响应。

存在问题

当 Master 或 Slave 不可用时，以上三种传统数据复制方式均有几率引起数据不一致。
数据库作为系统数据存储和服务的核心能力，其可用性要求非常高。在生产系统中，通常都需要用高可用方案来保证系统不间断运行，而数据同步技术是数据库高可用方案的基础。

解决方案

MAR 强同步复制方案是腾讯自主研发的基于 MySQL 协议的并行多线程强同步复制方案，只有当备机数据完全同步（日志）后，才由主机给予应用事务应答，保障数据正确安全。
原理示意图如下：



在应用层发起请求时，只有当从节点（Slave）返回信息成功后，主节点（Master）才向应用层应答请求成功，以确保主从节点数据完全一致。
MAR 强同步方案在性能上优于其他主流同步方案，具体数据详情可参见 强同步性能对比数据 。主要特点如下：

- 一致性的同步复制，保证节点间数据强一致性。
- 对业务层面完全透明，业务层面无需做读写分离或同步强化工作。
- 将串行同步线程异步化，引入线程池能力，大幅度提高性能。
- 支持集群架构。
- 支持自动成员控制，故障节点自动从集群中移除。
- 支持自动节点加入，无需人工干预。
- 每个节点都包含完整的数据副本，可以随时切换。
- 无需共享存储设备。

版权所有：腾讯云计算（北京）有限责任公司

TDStore 引擎
产品特性

最近更新时间：2024-05-22 15:12:11

TDStore 是腾讯云自研的金融级新敏态引擎，该引擎可以有效解决客户业务发展过程中业务形态、业务量的不可预知性，适配金融敏态业务。

透明分布式

TDSQL TDStore 引擎版计算节点基于 MySQL 8.0 实现，除个别受限的系统操作。TDStore 可以兼容99%的原生 MySQL 语法。使用时无需指定分区键，无须手动分库分表。单机 MySQL 的业务可以无损迁移到 TDStore 上，真正实现对业务应用无入侵。

多节点高性能读写

不同于传统的主-从模式，TDStore 计算层为多主模式，每个计算节点均可读写；用户可以随着业务量的增长而弹性扩展计算节点，单实例可支撑千万级 QPS 流量，帮助用户应对突如其来的业务峰值压力。

低成本海量存储

TDStore 存储层基于 LSM-Tree + SSTable 结构存放和管理数据，具有较高的压缩率，能有效降低海量数据规模下的存储成本。对于一些数据行重复度较大的业务场景，对比 InnoDB 存储引擎，TDStore 版最高可实现高达20倍的压缩率，能有效降低海量数据规模下的存储成本，单实例可支撑 PB 级别的存储量。

原生 Online DDL

TDStore 支持原生 Online DDL 操作，用户在业务运行过程中，有动态更改表结构的需求时，无须依赖如 pt 或 ghost 等外部工具组件，直接使用原生 MySQL DDL 语句便可完成。 TDStore 覆盖 MySQL 原生可支持的 instant 类型 DDL 操作，并且对于大部分类型（除涉及主键外的）DDL，均能以不阻塞业务的正常 DML 请求下完成。同时，TDStore 的 Online DDL 可以在多个计算节点之间保持一致性，不同表对象的 Online DDL 可以并行执行。

完整分布式事务支持

TDStore 以原生分布式的架构完整支持事务 ACID 特性，支持全局一致性读特性，整体事务并发控制框架基于 MVCC + Time-Ordering 的方式实现。
分布式事务协调者由分布式存储层节点担任，而当存储节点在线扩容遇到数据分裂或切主等状态变更的场景时，TDStore 均可实现不中断事务，将底层数据状态的变更对事务请求的影响降到最低，从而做到无感知的集群扩缩容。

数据位置感知

用户可根据业务逻辑灵活调整数据物理位置，有效的打散热点数据，减少分布式事务，优化查询下推，指定容灾策略，从而提升性能和可靠性，实现数据智能调度。

容器化管控系统

TDStore 的管控部分采用了容器化的实现方式，借助云原生的能力，能够快速且方便地管理 TDStore 实例，免除了繁琐的物理机上架，配置等资源管理运维操作，同时也无需关心资源的使用率情况，即买即用，支持高效弹性扩缩容。

版权所有：腾讯云计算（北京）有限责任公司

实例架构

最近更新时间：2024-05-24 00:45:11

TDStore 实例分为集群版和基础版两种：

- **集群版**：由多个（≥3个）节点构成，以三副本 Raft 集群的形态提供高性能可用的数据库服务，适用于企业生产环境。
- **基础版**：由单个节点构成，以较低的成本提供完整的数据库功能，适用于个人用户。

> **说明：**
> 基础版实例创建后可以通过控制台升级为集群版实例，集群版实例创建后不可以降级为基础版实例。

TDStore 实例内的节点分为对等架构和分离架构两种：

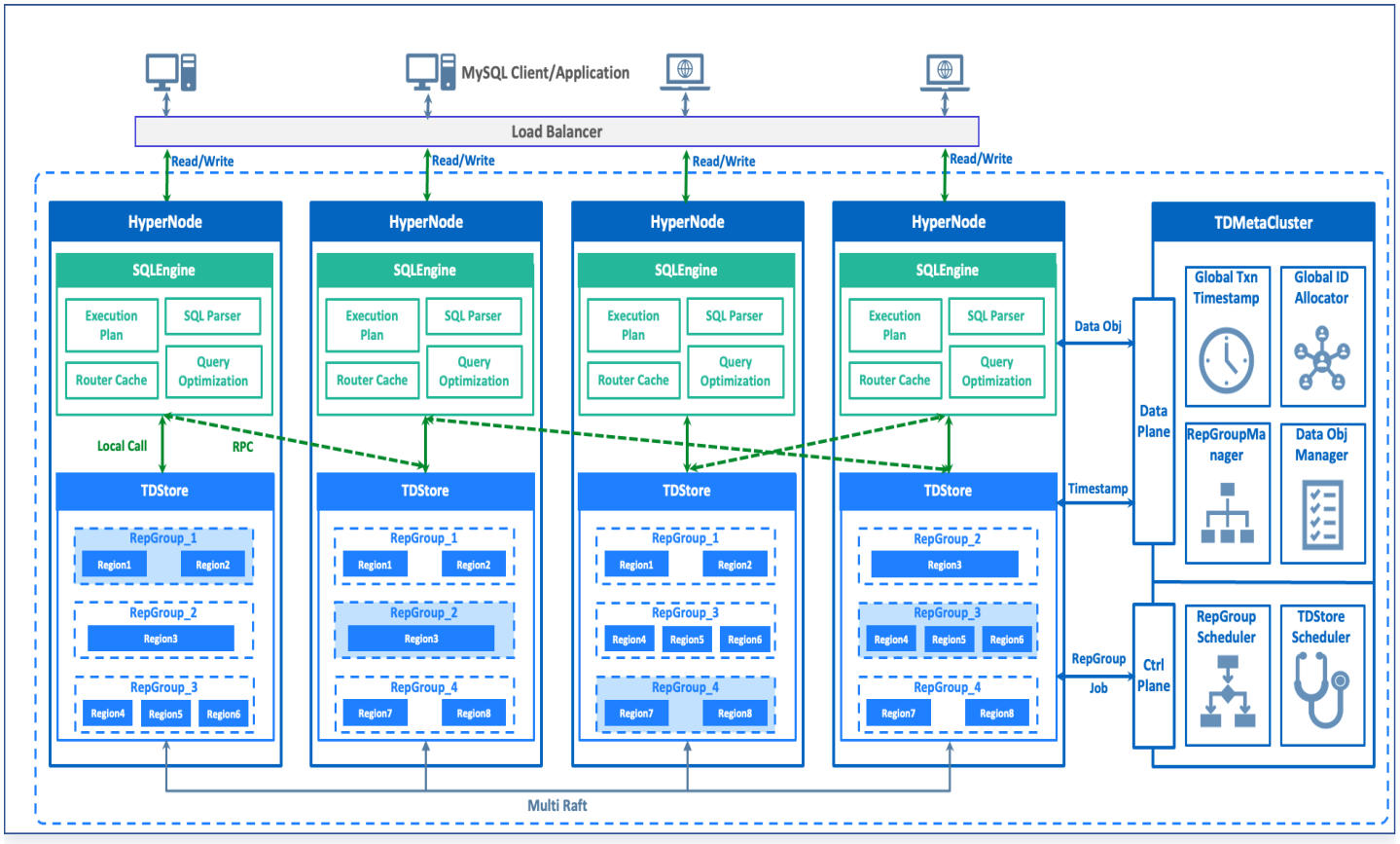
- **对等架构**：计算层 SQL Engine 与数据层 TDStore 合并在一个物理节点中，减少硬件节点数量和跨节点通信，从而降低成本并提高性能。
- **分离架构**：计算层 SQL Engine 与数据层 TDStore 分别在不同的物理节点中。

TDStore 技术架构

无论是集中式单机数据库还是分布式数据库，功能模块通常可以分为三大组件：

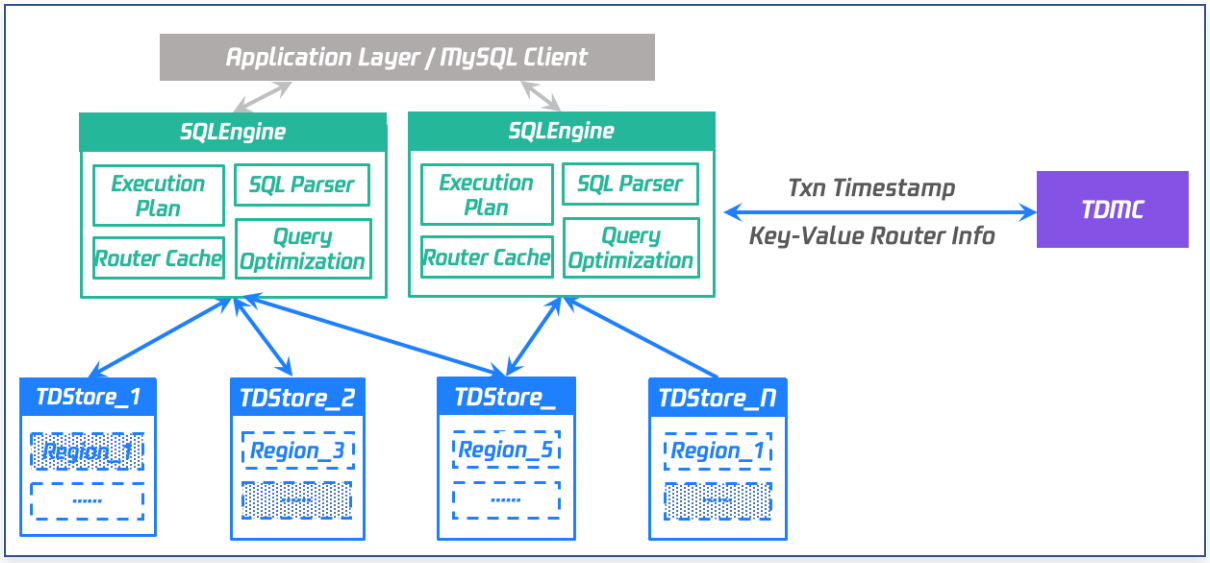
- 计算引擎：主要包括 SQL 解析、优化器、执行器等。
- 存储引擎：主要包括事务处理、数据存储等。
- 元数据服务：主要包括全局逻辑时钟服务、全局 ID 生成器、元数据存储、调度引擎（数据/容灾调度）以及负载采集等。

TDStore 的总体架构如图所示：



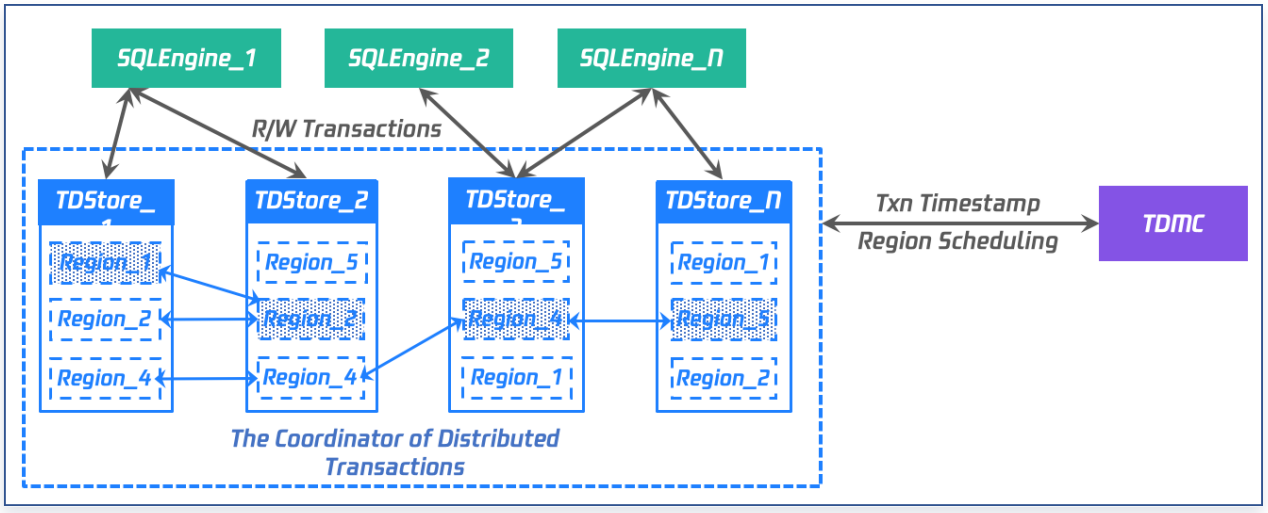
计算引擎-SQLEngine

- 内核：基于 MySQL 8.0 实现，对 MySQL 兼容度高。
- 架构：计算层为多主架构，无状态化设计，每个 SQLEngine 节点均可读写，可以承受千万级别 QPS 写入压力。
- 交互：从管控节点获取全局事务时间戳和数据路由信息，然后与存储节点进行事务的交互，向客户端返回结果。



存储引擎-TDStore

- 架构：基于 LSM-Tree 和 Multi-Raft 的分布式 KV 存储引擎。
- 数据：基于 Raft 同步的多副本的存储，数据根据 Key 范围分布在不同 Region 上。
- 交互：TDStore 接收来自计算节点的请求，处理后返回结果；每个 Region 的主副本负责接收和处理读写请求。

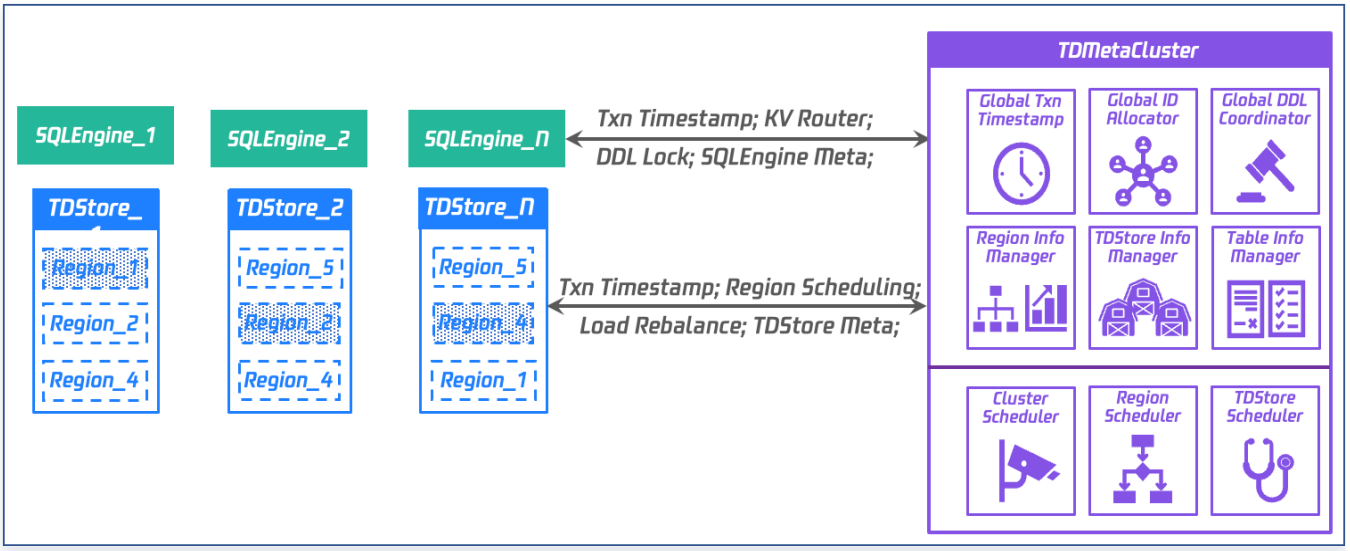


元数据服务-TDMC

- 架构：基于 Raft 的一主两备的元数据管理集群，由 Leader 提供服务。
- 数据：
  - 分配全局唯一且递增的事务 ID。
  - 管理 TDStore 和 SQLEngine 元数据。
  - 管理 Region 数据路由信息。
  - 全局 MDL 锁管理。

版权所有：腾讯云计算（北京）有限责任公司

- 管控：
  - 调度 Region 的分裂、合并、迁移、切主。
  - 存储层的扩缩容调度。
  - 存储层的负载均衡调度。
  - 各维度的异常事件告警。



存储模型

TDStore 的存储引擎会将所有用户数据映射到（-∞，+∞）的线性有序无限的 Key 空间中，每行数据对应到 key 空间中的某个点。

```
//示例
create table t1 (
  f1 varchar(50),
  f2 varchar(20),
  f3 varchar(20),
  f4 varchar(20),
  primary key(f1,f2),
  index idx_f3(f3)
);
insert into t1 values('a', 'b', 'c', 'd');
```

在上图的示例中，insert 一行数据后，会产生两个 KV 键值对：
主键：pk_encode('t1', 'a', 'b') -> pk_value_encode('c', 'd')
索引：sk_encode('idx_f3', 'c', 'a', 'b') -> sk_value_encode()
因为每个 key 的最前面部分是该数据对象（表/索引/分区表）的 ID，故该数据对象的所有数据会连续分布在 Key 空间的某一段上。

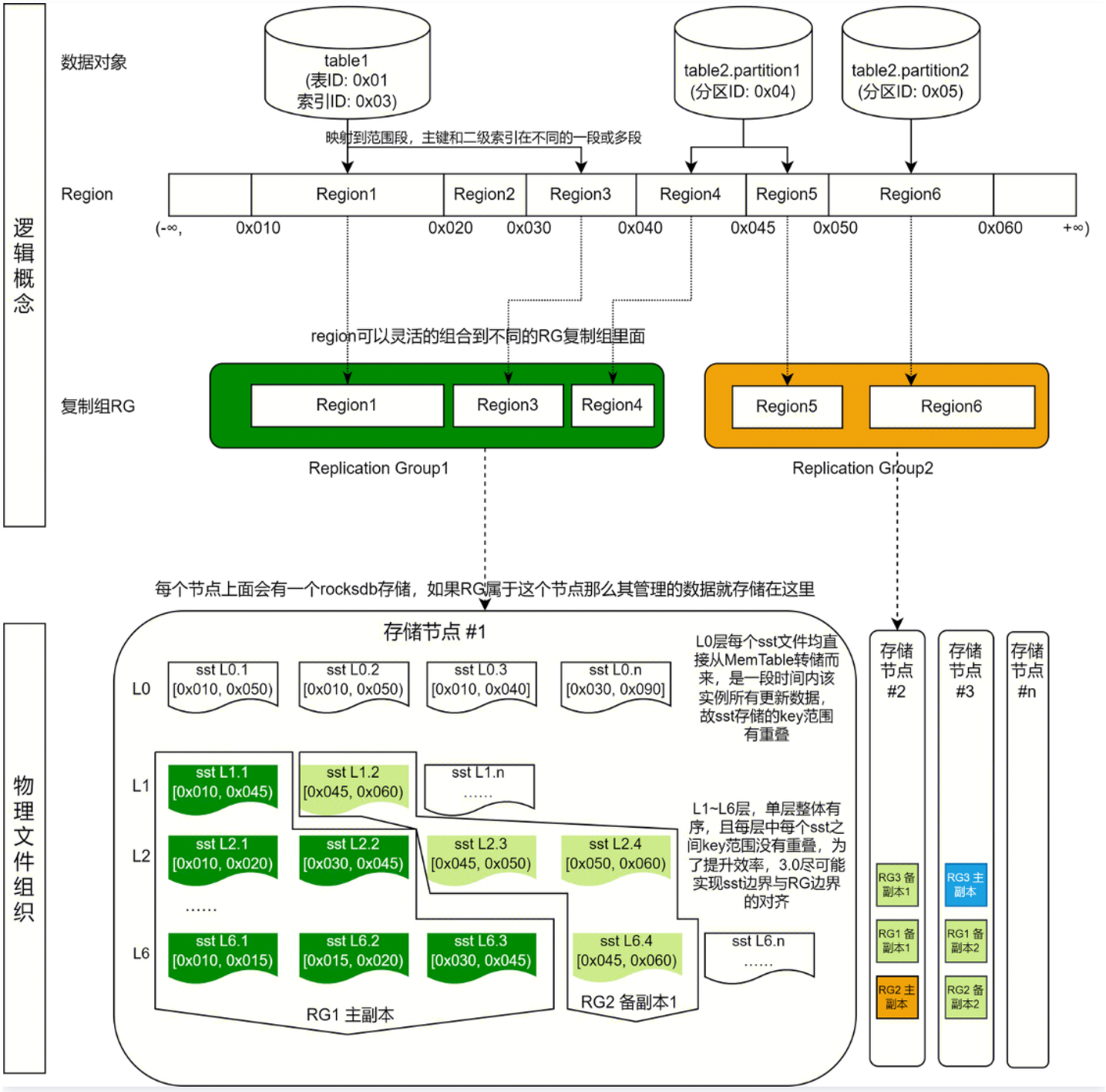
三层组织模式（DO-Region-RG）

数据对象（Database Object）：例如表（table），分区（Partition），索引（Index），通常会为这些数据对象分配一个全局唯一的 ID（index_id）。
数据分片（Region）：

①：一段连续的，左闭右开的 key 空间，[startkey, endkey)。
②：一个数据库对象对应一个或多个 Region，且在运行过程中，可以根据数据调度策略，对 Region 进行分裂，或将多个 Region 合并为一个 Region。
③：数据调度的基本单位，不同的 region 可以分布在该实例中的任何节点。
④：Region 是一个逻辑概念，在底层存储上，并不是完全独立的一段或一个文件。

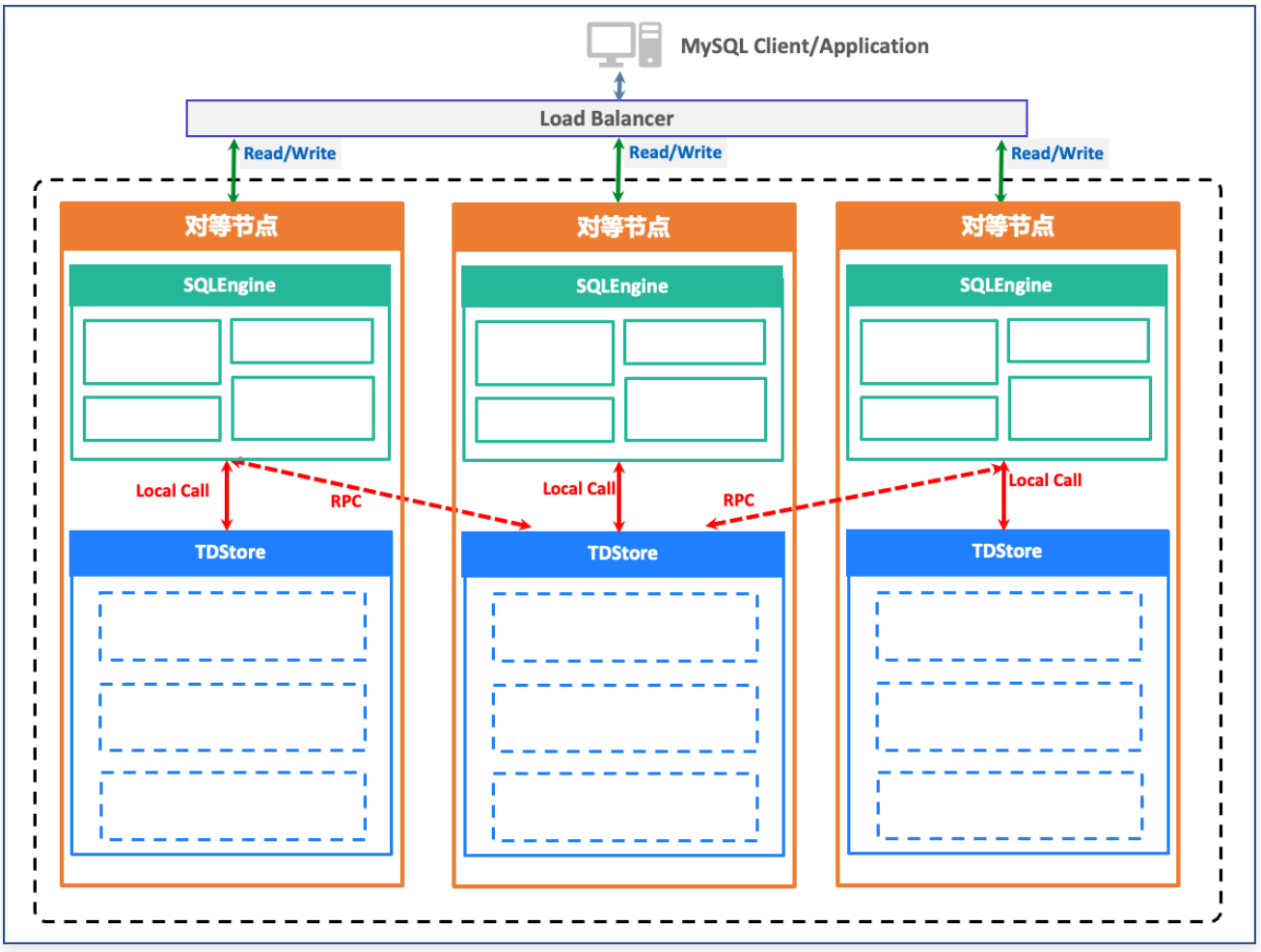
复制组（Replication Group）：

①：一个复制组中可以包含一个或多个 Region，这些 region 属于一个或多个数据对象。
②：在运行过程中，可以合并 RG，也可以分裂 RG（具体实现中通过创建 RG，迁移 region 进入或离开 RG 等来实现）。
③：一个 RG，对应一个 Raft 日志流（Redolog/WAL），如果一个事务（transaction）所涉及到的 key 都在一个 RG 内，则该事务可转换为单机事务，如果一个事务涉及数据跨 RG，则需要走两阶段的分布式事务协议。



对等架构节点

在 TDStore 的架构设计中，我们既要让计算离存储（缓存/存储）尽可能的近，也要能做到计算与存储分离实现高弹性，因此 TDStore 的设计为：

- 采用对等架构（HyperNode）设计，每个对等节点（进程）均包含有完整的计算、存储、日志三个功能引擎。
- 计算层与本地存储采用本地访问模式，访问远端存储则采用网络 RPC 访问模式，尽可能的确保访问本地数据的性能。
- 根据不同业务场景的需要，通过元数据及调度模块，也可以实现对节点进行角色指定，例如支持全功能节点（计算存储日志三个服务都提供），计算节点（仅支持计算，所有数据通过 Remote 访问），存储节点（仅支持数据存储），日志服务（仅提供日志订阅服务）等。

> **说明：**
> 节点角色指定的功能特性目前在公有云未开放。