



TDSQL架构及原理

sevenwzhang(张文)

2018-12-12

目录

- 产品简介以及使用场景
- TDSQL架构分析及模块介绍
- 数据一致性保障
- 分布式TDSQL实践
- 部署实践



产品介绍及使用场景

TDSQL (Tencent Distributed MySQL)
是腾讯针对**联机交易场景**推出的高一致性,
分布式数据库集群解决方案。

定位

- 在技术栈中的位置
 - 位于数据层位于数据层
 - 标准TDSQL适用于OLTP联机交易场景，不适用于OLAP复杂分析类SQL

X 统计、分析类场景

建议：类似需求放入T-SPARK平台

T-SPARK是TDSQL用来支持
OLAP场景的组件



MySQL兼容性



默认支持

MariaDB

10.1.9

10.0.10

兼容MySQL 5.6

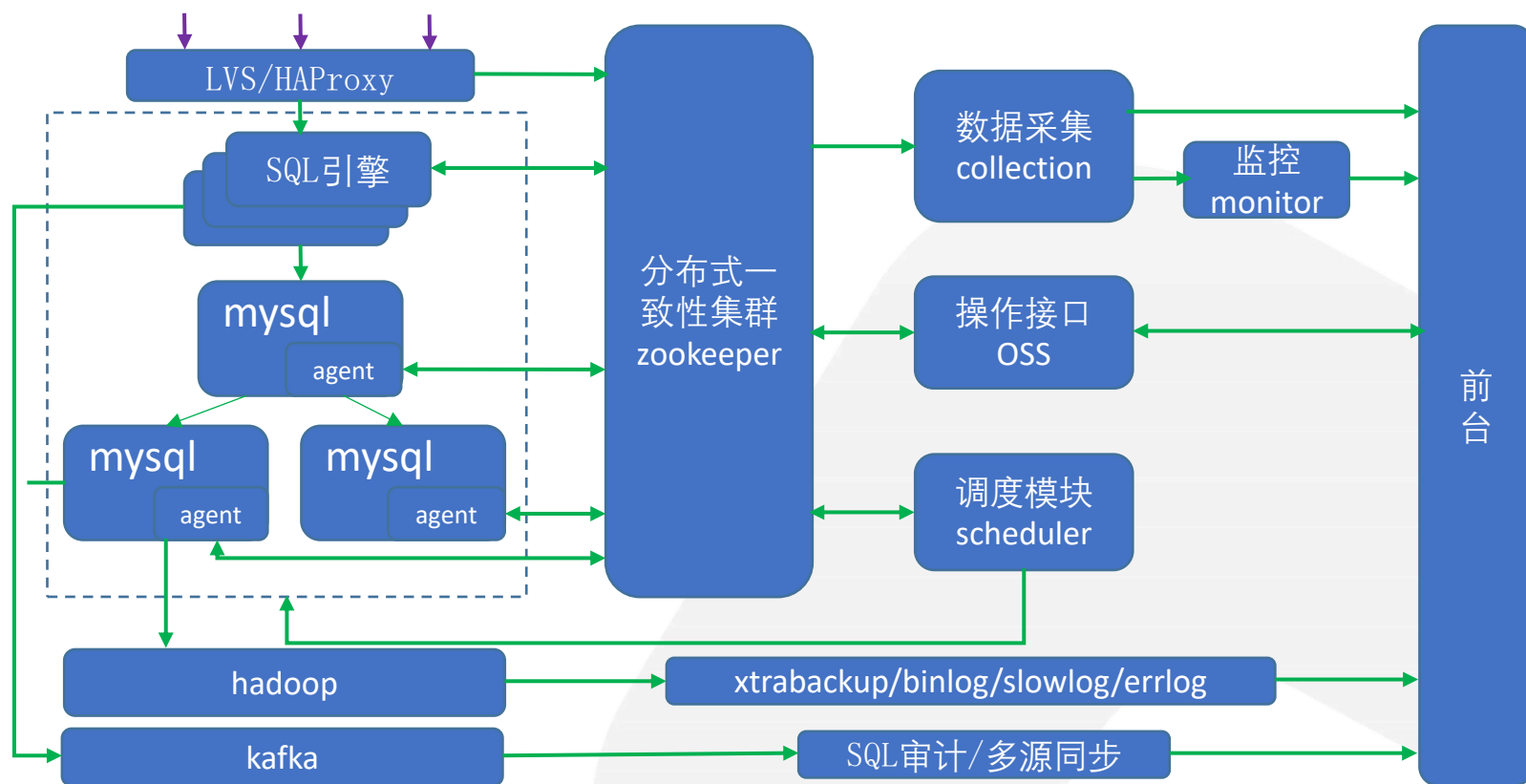
Percona/MySQL

5.7.17

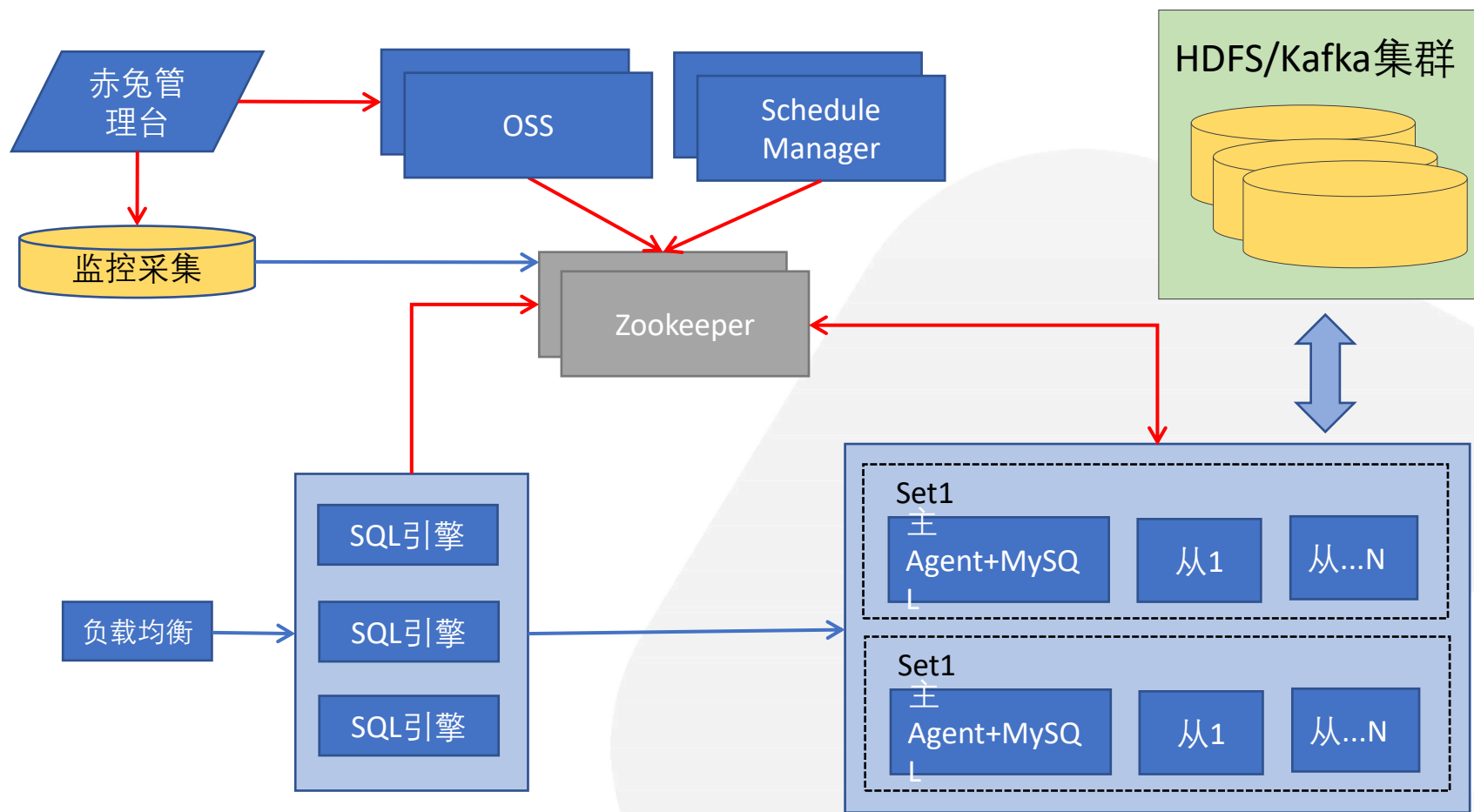


架构分析及模块介绍

系统总览



核心架构

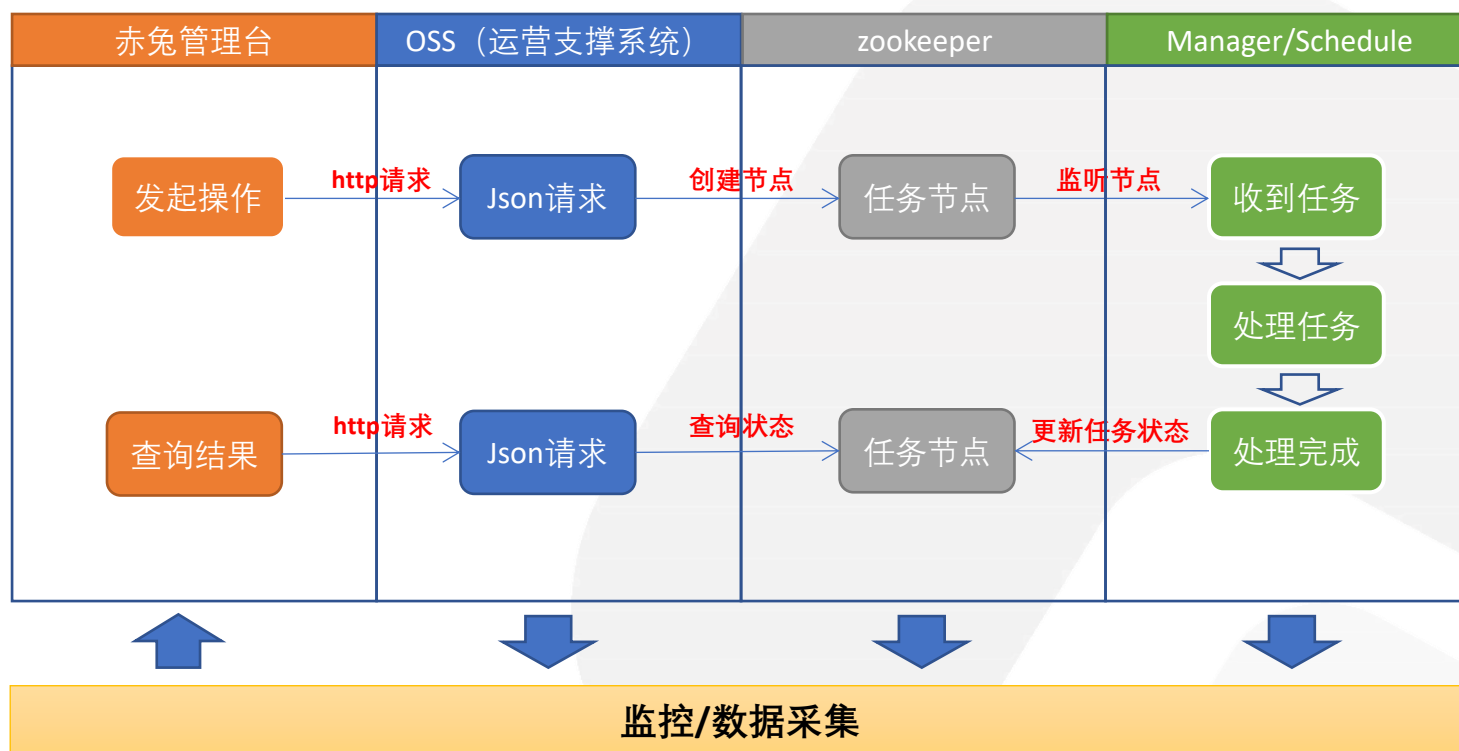


模块划分：管控模块

- 管理模块

- 一个集群搭建一套即可，一般是复用一组机器；
- 对机器的性能要求相对于DB机器和网关机器要低；

包括：Zookeeper、Scheduler、Manager、OSS、监控采集程序、赤兔管理控制台

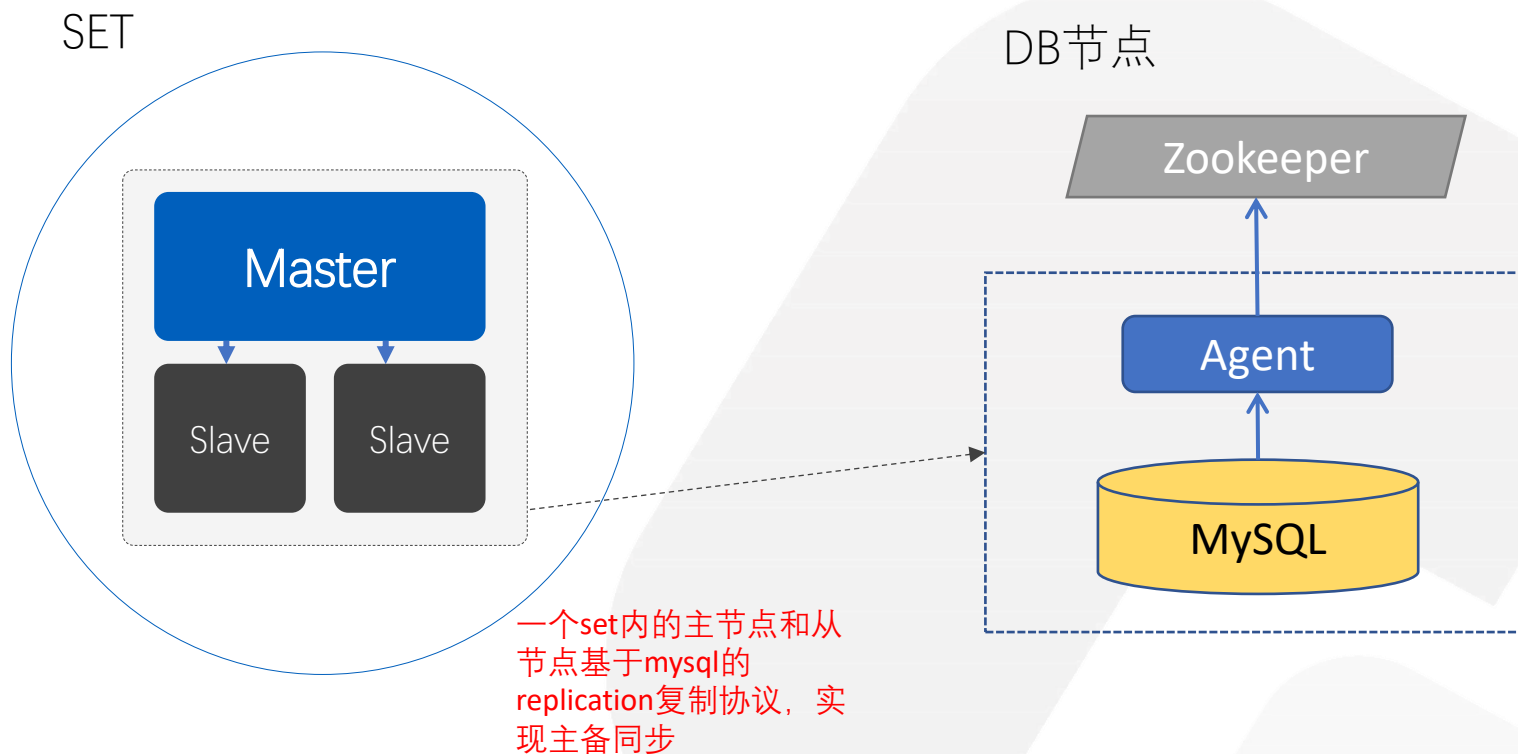


模块划分：DB节点

- DB节点

- DB节点上部署的数据库服务，属于IO密集型服务，对机器的IO要求较高，一般建议配置SSD硬盘；
- Agent属于旁路模块，主要承担DB的状态监控，存活检测以及其他功能性任务的执行。

注：单节点的MySQL实例通过Agent和TDSQL集群建立联系

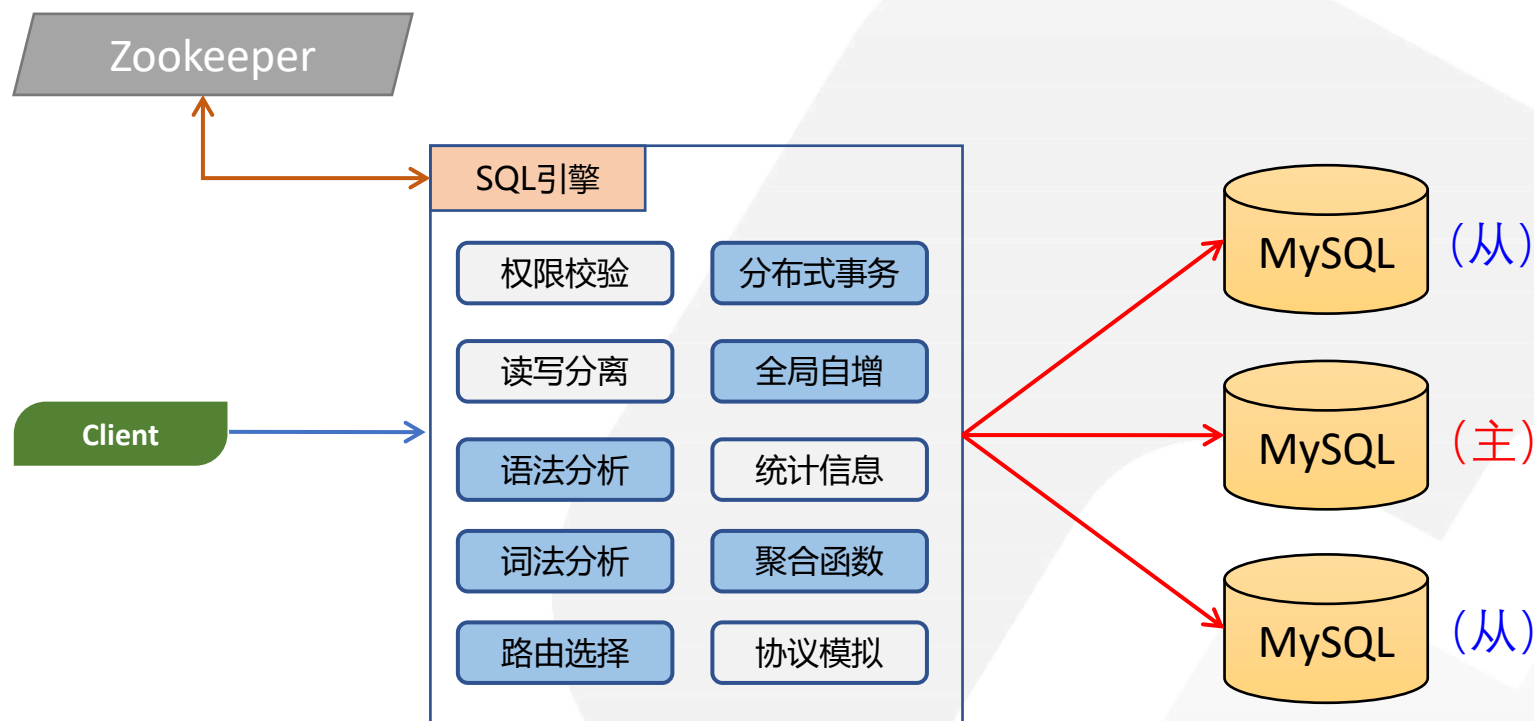


模块划分：SQL引擎（组）节点

- SQL引擎节点

- 在TDSQL整体位于接入层的位置，属于cpu密集型服务
- 在机型选择上对机器的cpu要求最高，其次是内存

注：SQL引擎没有主备之分，要求多节点部署以实现容灾





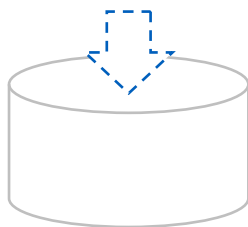
数据一致性保障

主备数据复制方式

- 原生复制方式

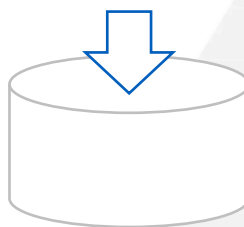
- 异步：主机不等从机应答直接返客户端成功
- 半同步：主机在一定条件下等待从机应答再返回客户端成功

强同步：主机等待至少一台备机应答成功后才返回客户端成功



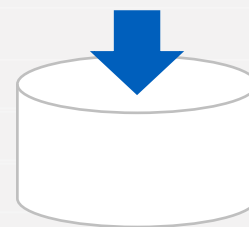
异步复制

Async replication



半同步复制

Semi-Sync replication



强同步复制

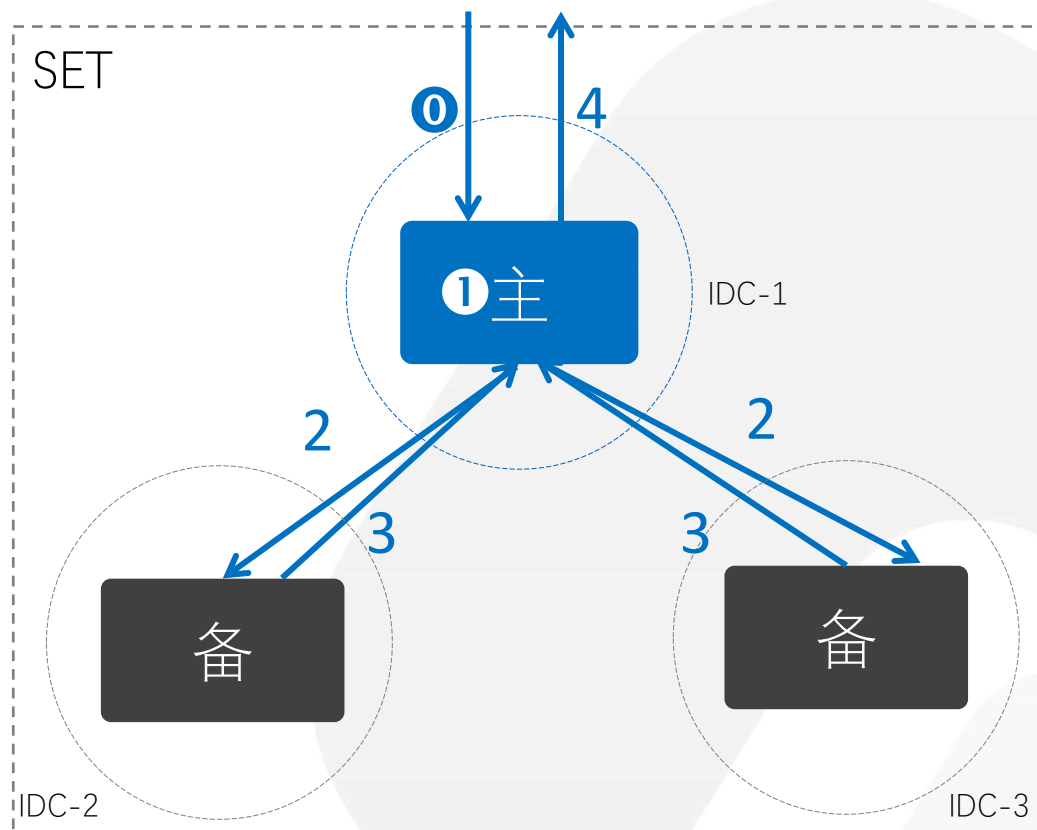
Sync replication

强同步

- 数据一致性

- 强同步机制：任何一笔应答前端成功的请求除了在主机落盘成功外还会在其中一台备机落盘成功；
- 强同步性能：在原生异步复制的基础上做了性能改良，接近异步复制

强同步机制是TDSQL数据不会丢、不会错的最核心的保障



数据复制比较

- 半同步复制
 - 超时后会退化为异步
 - 原生的半同步复制性能不好，尤其在跨IDC的网络环境下

异步：虽然性能最好，但是数据没有多副本的保障

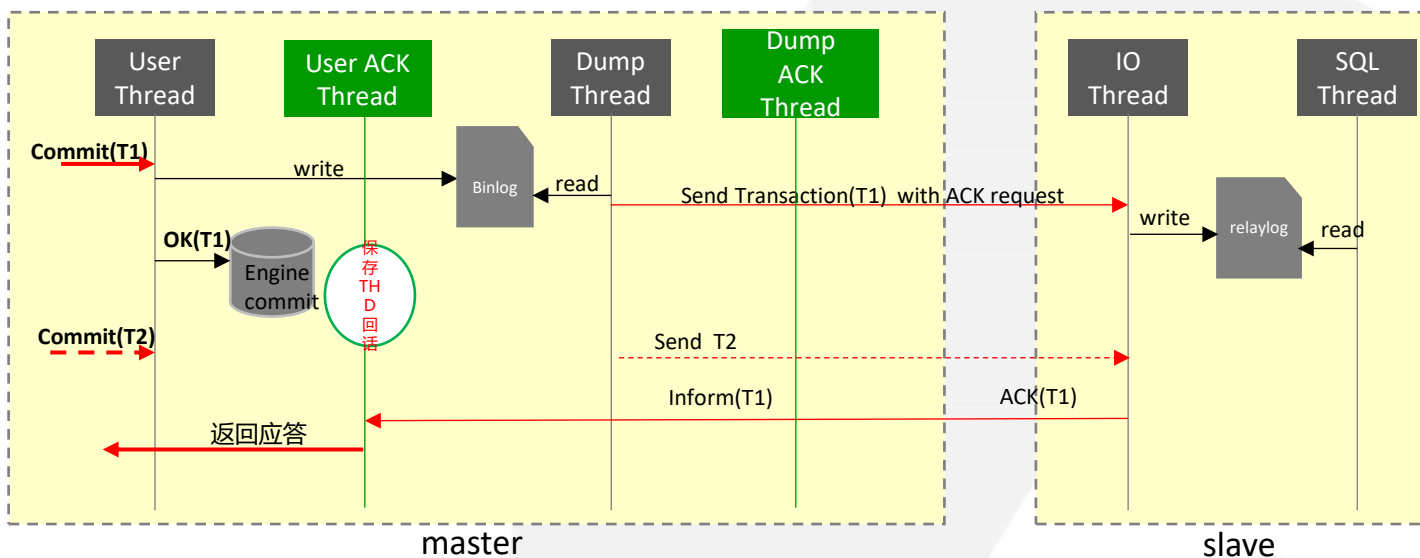
主备复制方案（同城跨IDC）	TPS	时耗(ms)
异步	20,000	<10
半同步	2,200	4~600
MariaDB Galera Cluster	6,000	4~10000

备注：以上测试数据仅有对比意义

强同步改造

- 传统半同步复制
 - mysql工作线程在等待备机应答时阻塞
 - mysql工作线程利用率低

优化：新开两组线程，实现工作线程的异步化



数据复制比较

- 强同步复制
 - TPS和时耗与异步复制接近

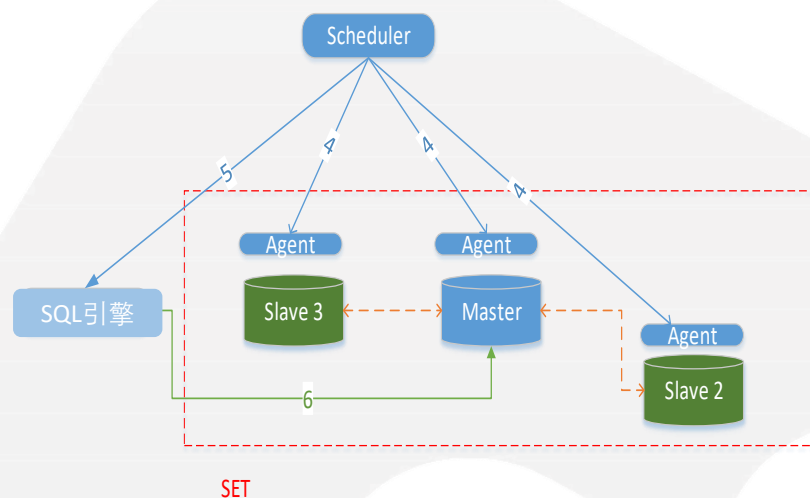
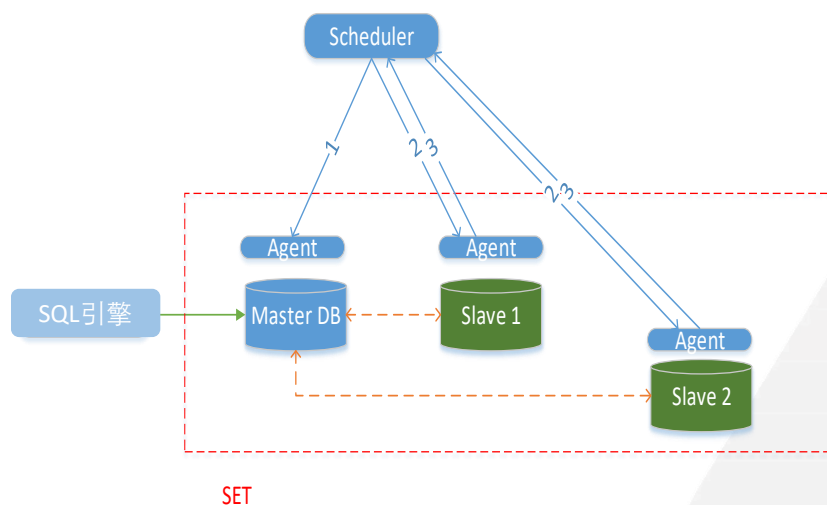
主备复制方案 (跨IDC)	TPS	时耗(ms)
异步	20,000	<10
半同步	2,200	4~600ms
强同步	20000	<10
MariaDB Galera Cluster	6,000	4~10000ms

核心功能：容灾切换

- 原则

- 主机可读可写，备机只读，任何时候只有一个主机提供服务
- 宁愿拒绝服务，不提供错误的服务

过程：主节点发生宕机，重新选举出新的主节点并提供服务，同时保证数据的一致性。



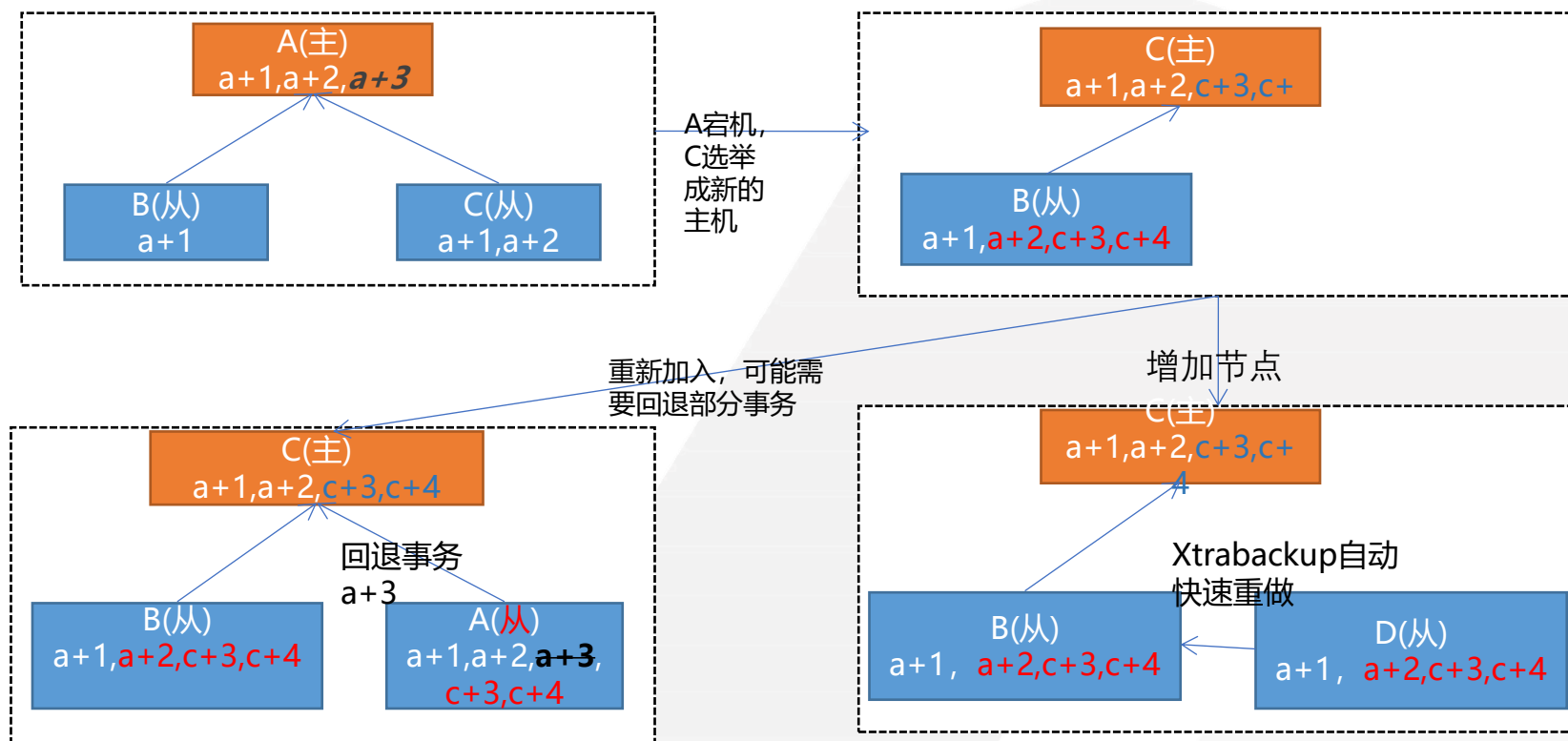
- 1、主DB降级为从机
- 2、参与选举的从机上报最新的binlog文件偏移
- 3、scheduler收到binlog点之后，选择出binlog最大的节点

- 4、重建主备关系
- 5、修改路由
- 6、请求发给新的主机

数据强一致性

• 特殊场景

- 极端场景下数据一致性的保障;
- 快速重建节点, 即新加备节点快速恢复数据。





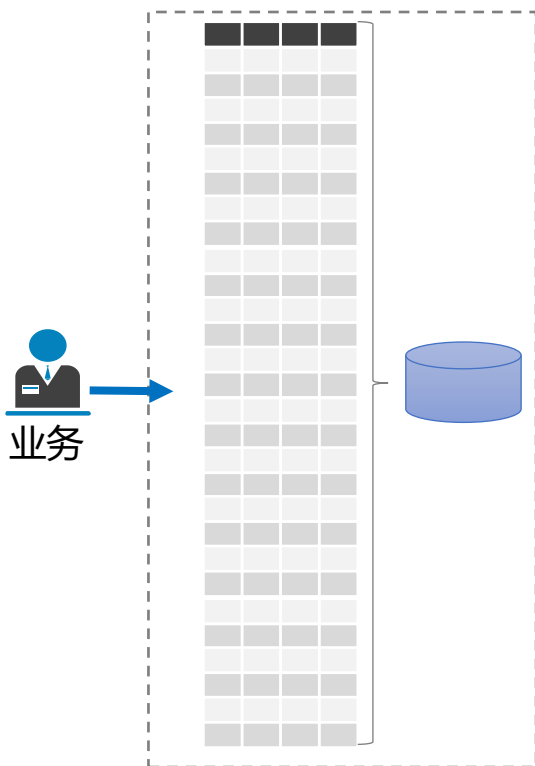
分布式TDSQL实践

分布式实践：分表

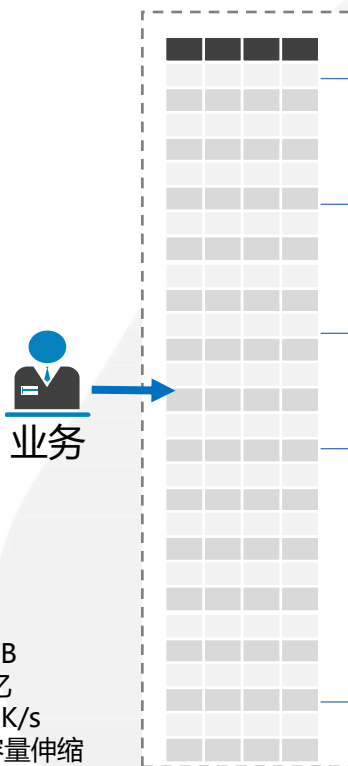
- 分布式

- 在单实例模式下，一张库表分布在一个mysql实例上；
- 在分布式模式下，一张表根据分片的数量分布在不同的mysql节点上；

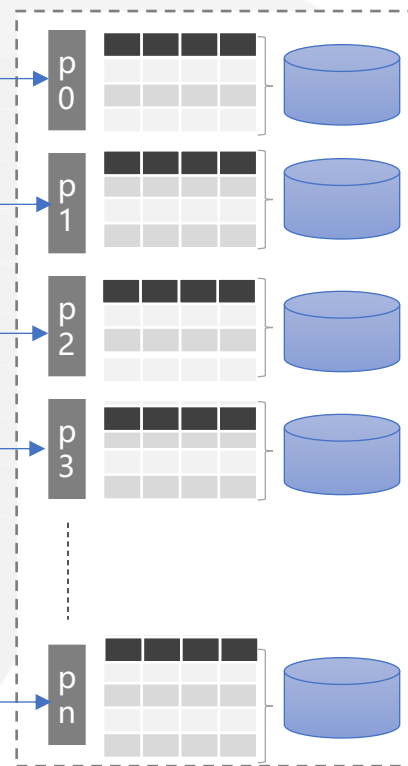
逻辑表/物理表



逻辑表



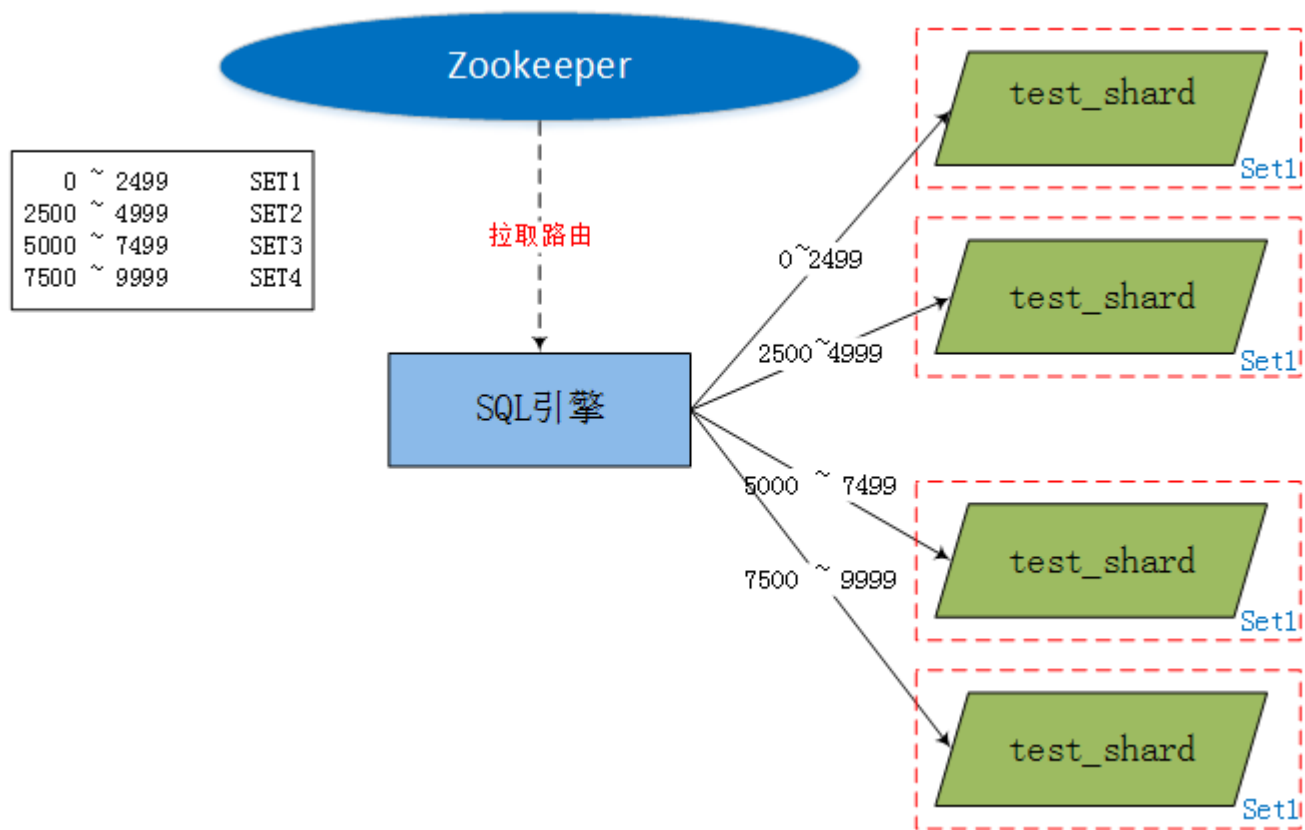
物理表



分布式实践：分表

- 按照shardkey拆分

- 不同的分片负责不同范围的号段，网关根据sql中的shardkey值发往对应的分片。
- 最初只有一个分片，通过不断拆分为多个分片。**

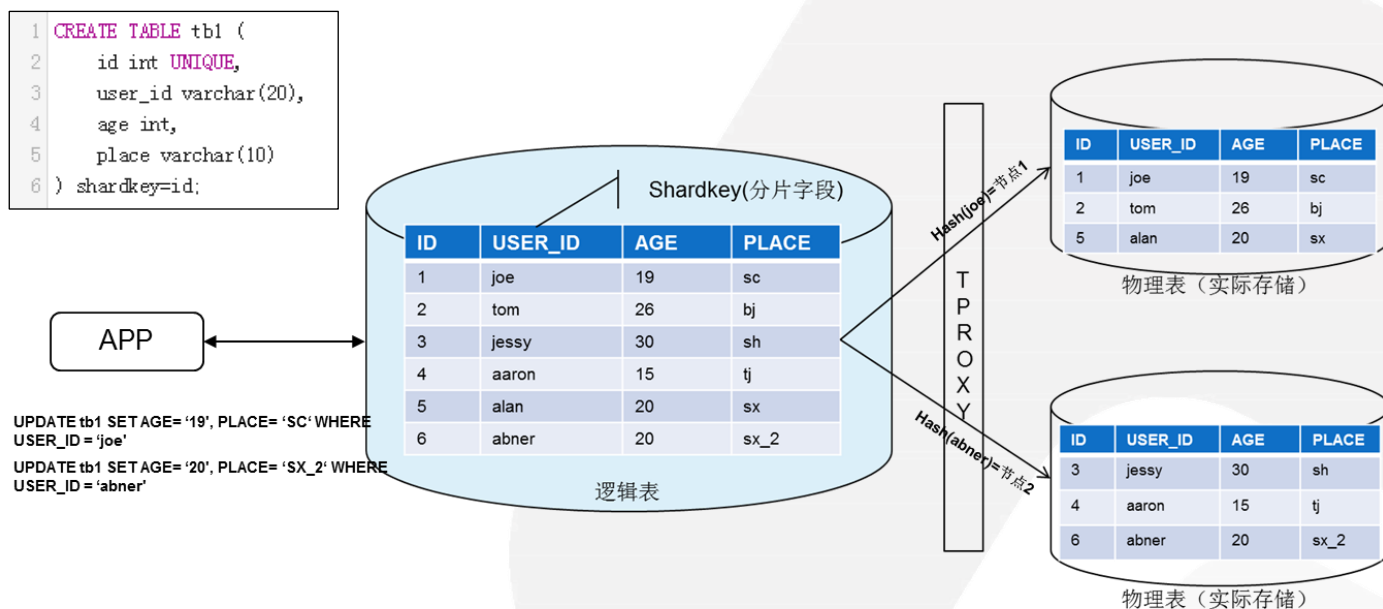


分布式实践：分表

• 典型SQL举例

- 创建表时需要指定路由字段shardkey;
- 业务SQL的增、删、改、查包含shardkey时, Proxy通过对shardkey进行hash;
- 数据根据分片算法, 将SQL发往对应的分片。

注意：如果SQL中不带shardkey则该SQL会发往所有的分片

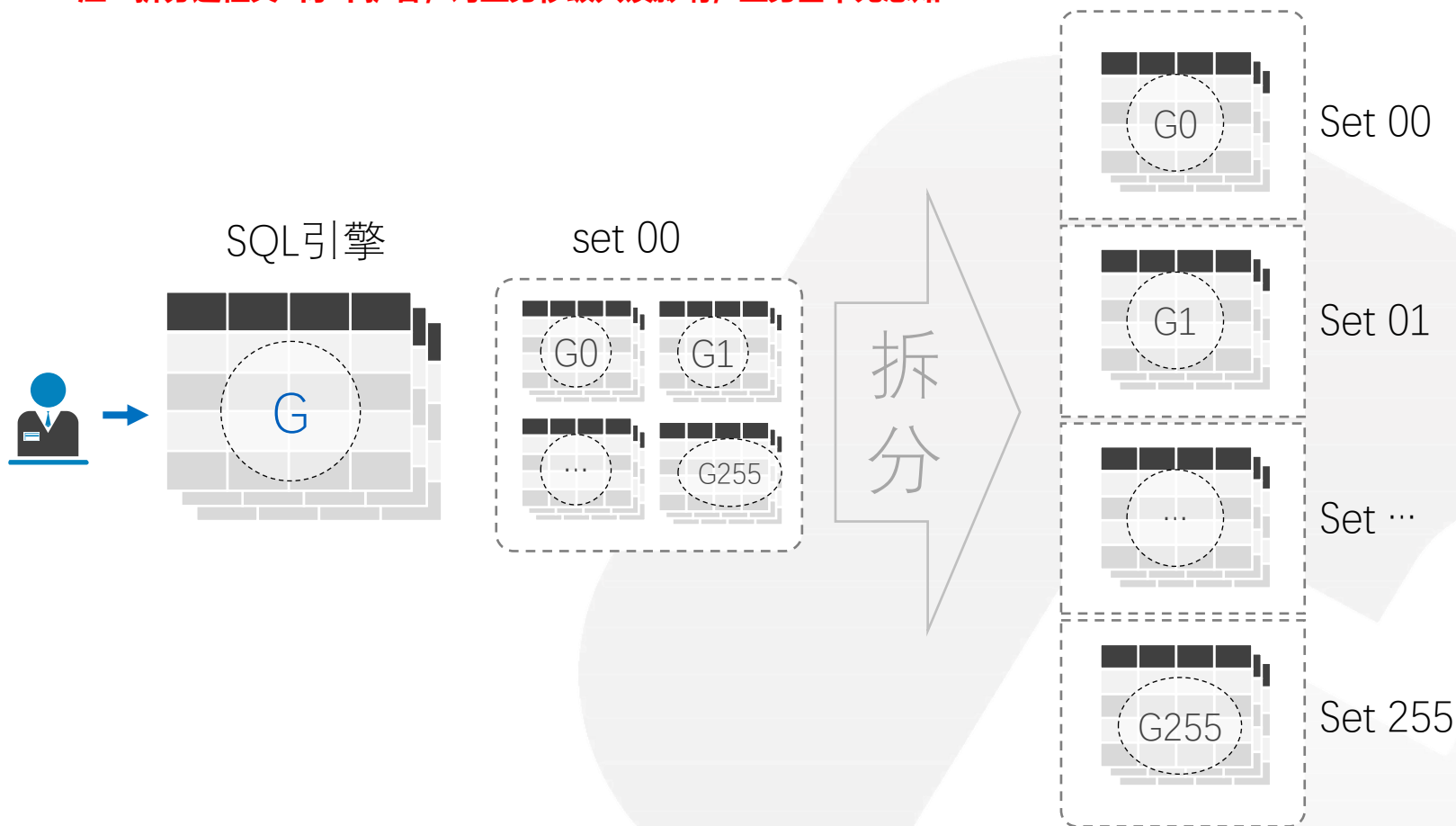


分布式实践：水平拆分

- 拆分过程

- 最起初一张表在一个SET上，随着节点容量的瓶颈逐步拆分到其他SET

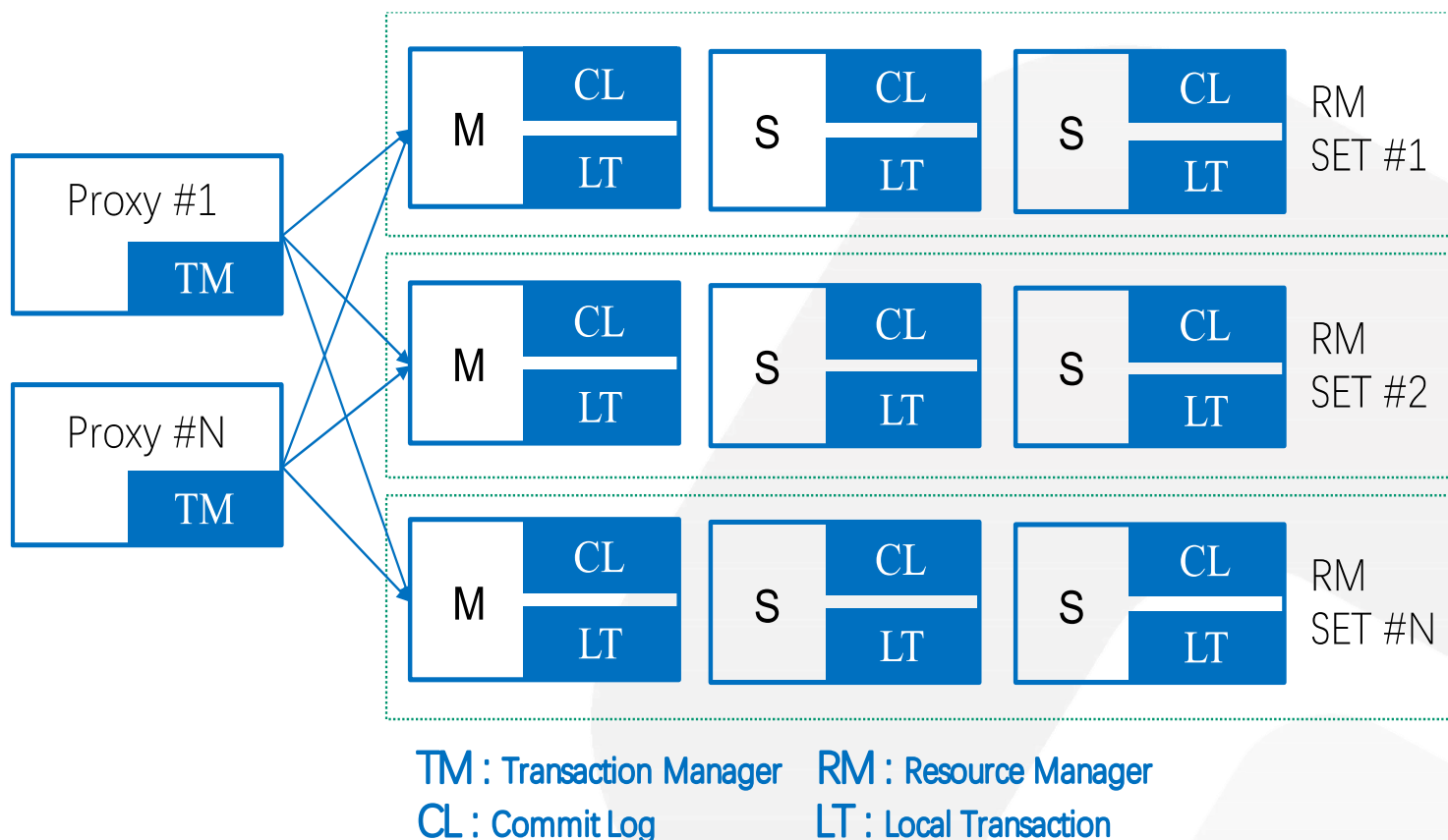
注：拆分过程又叫水平扩容，对业务秒级只读影响，业务基本无感知



分布式事务

- 数据一致性

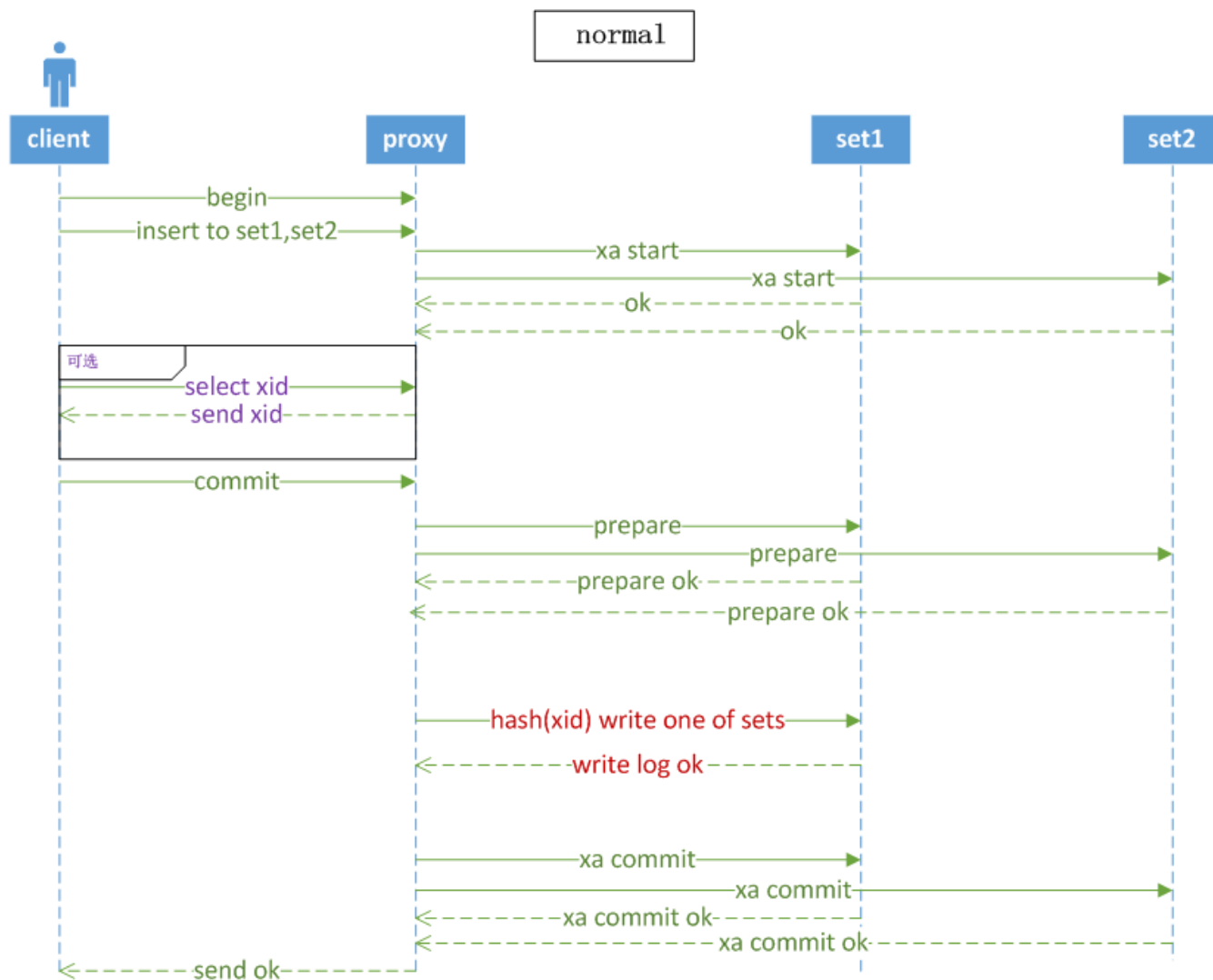
- 单节点的事务由InnoDB引擎保障，多节点涉及不可靠的网络链路，相对单节点较为复杂。
- 要求：完全去中心化、性能线性增长，健壮异常处理，全局死锁检测机制，TPCC标准验证



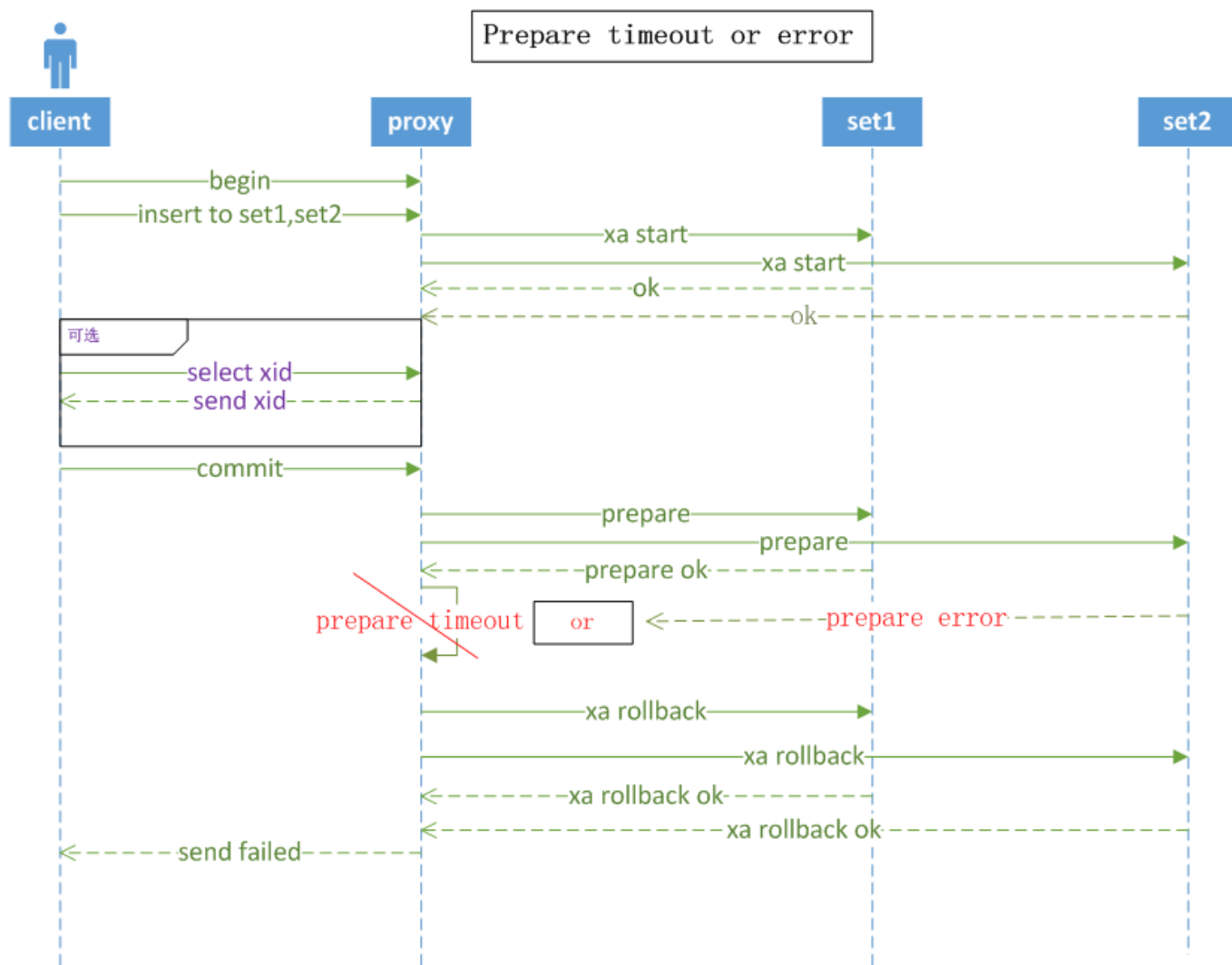
分布式事务

- 数据一致性

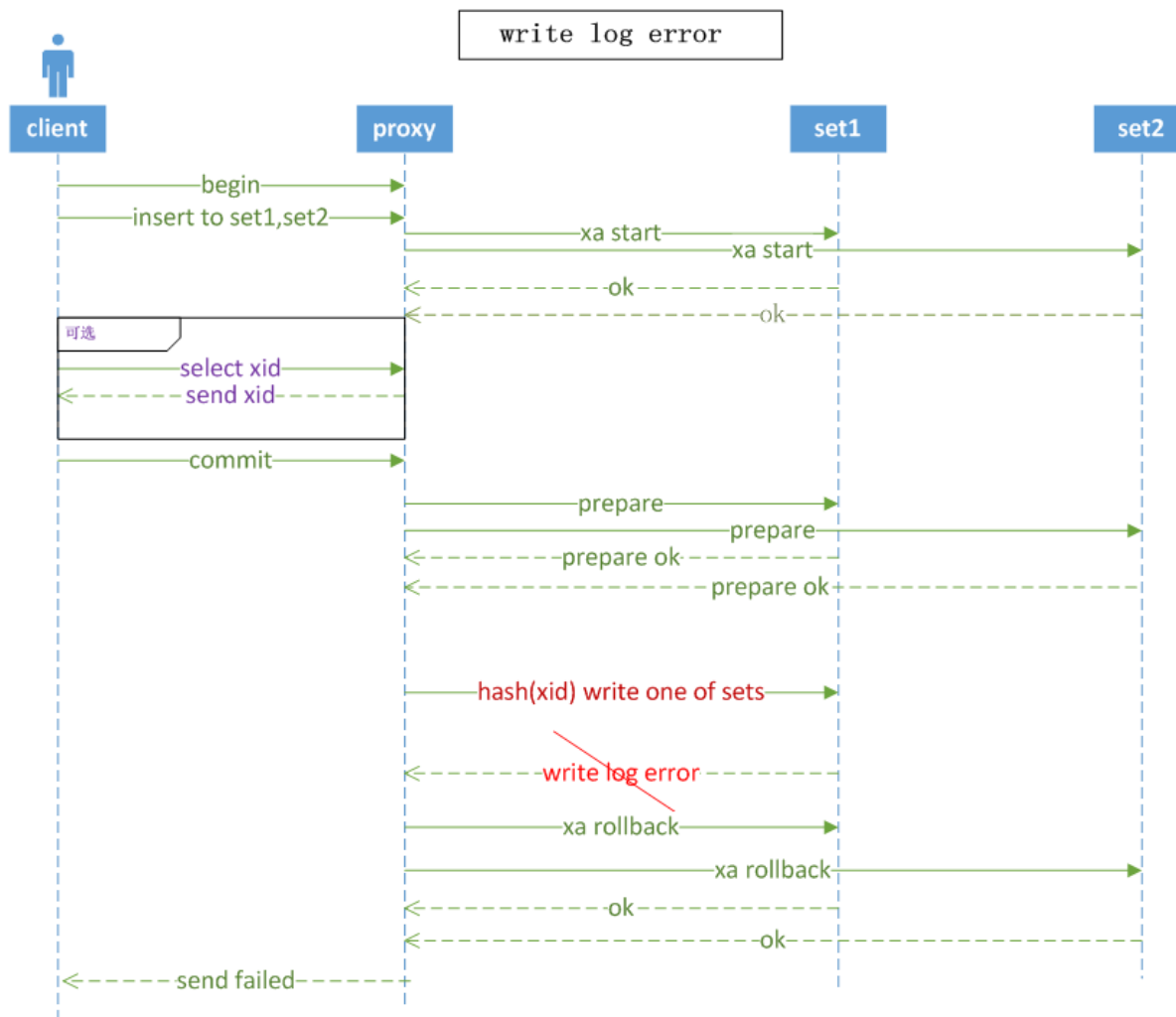
- Prepare 超时或者失败;
- Commit log写失败
- Commit log写超时
- Commit超时或者失败



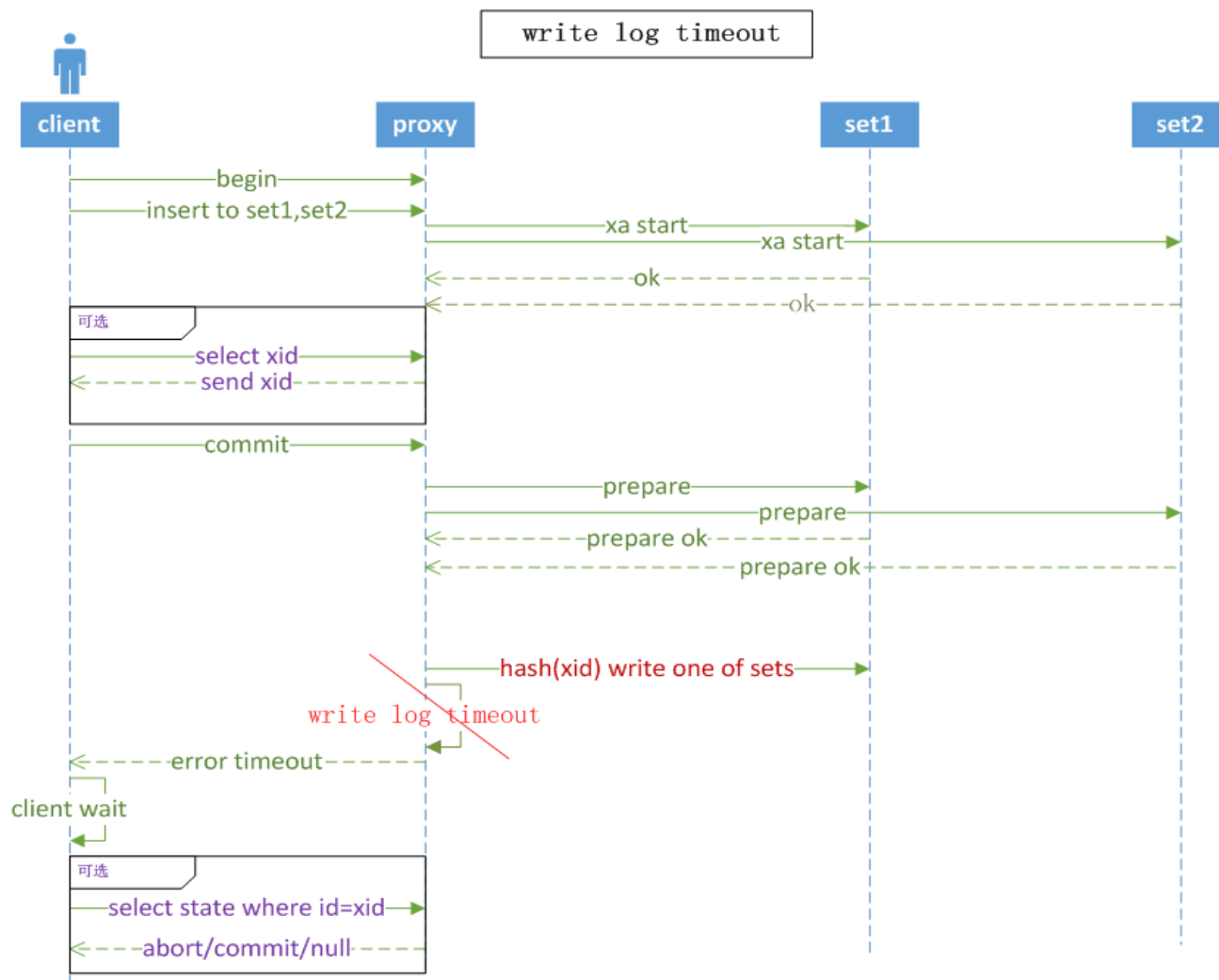
分布式事务 - Prepare 超时或者失败



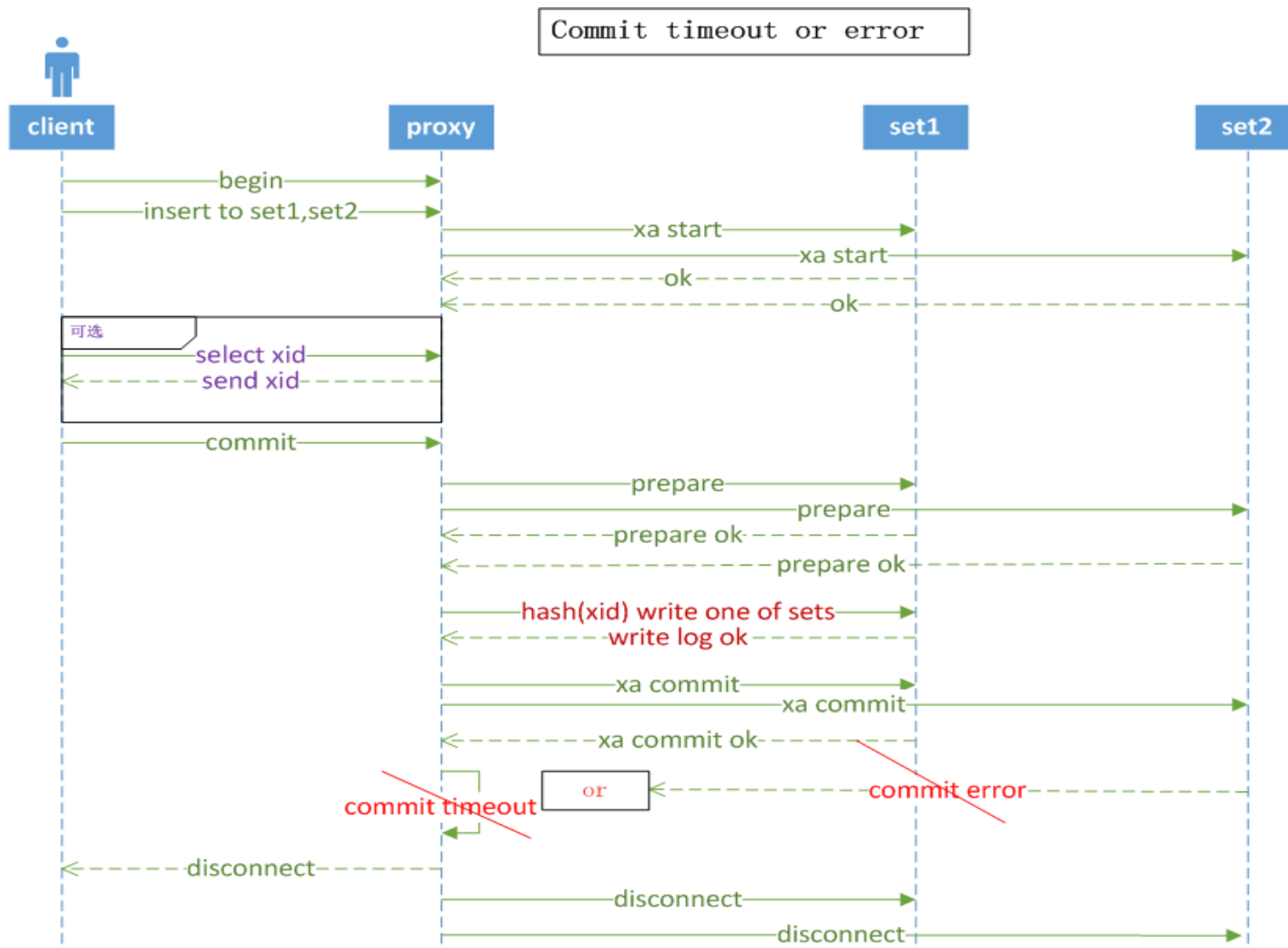
分布式事务 - Commit log写失败



分布式事务 - Commit log写超时



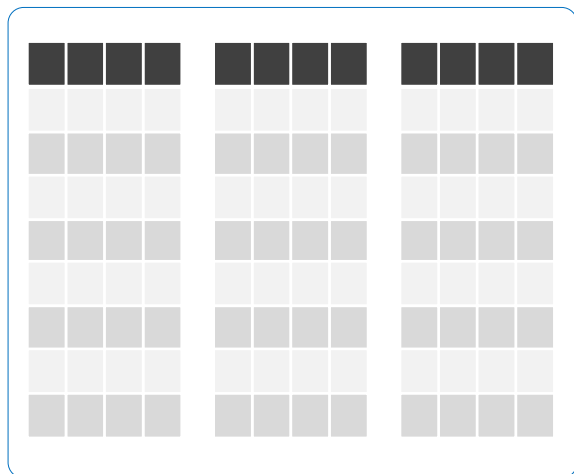
分布式事务 - Commit超时或者失败



分布式总结

- 两种模式
 - 都具备金融级别的容灾，数据一致性保障；
 - Noshard对SQL没有限制，Shard对SQL写法有限制。

NoShard



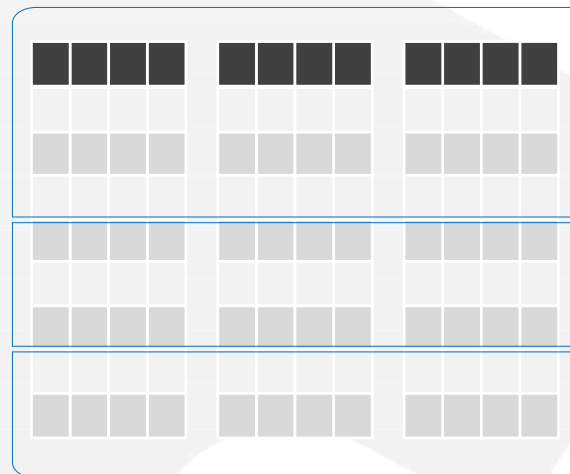
完全
兼容

高一
致性

自动
容灾

水平
伸缩

Shard



完全
兼容

高一
致性

自动
容灾

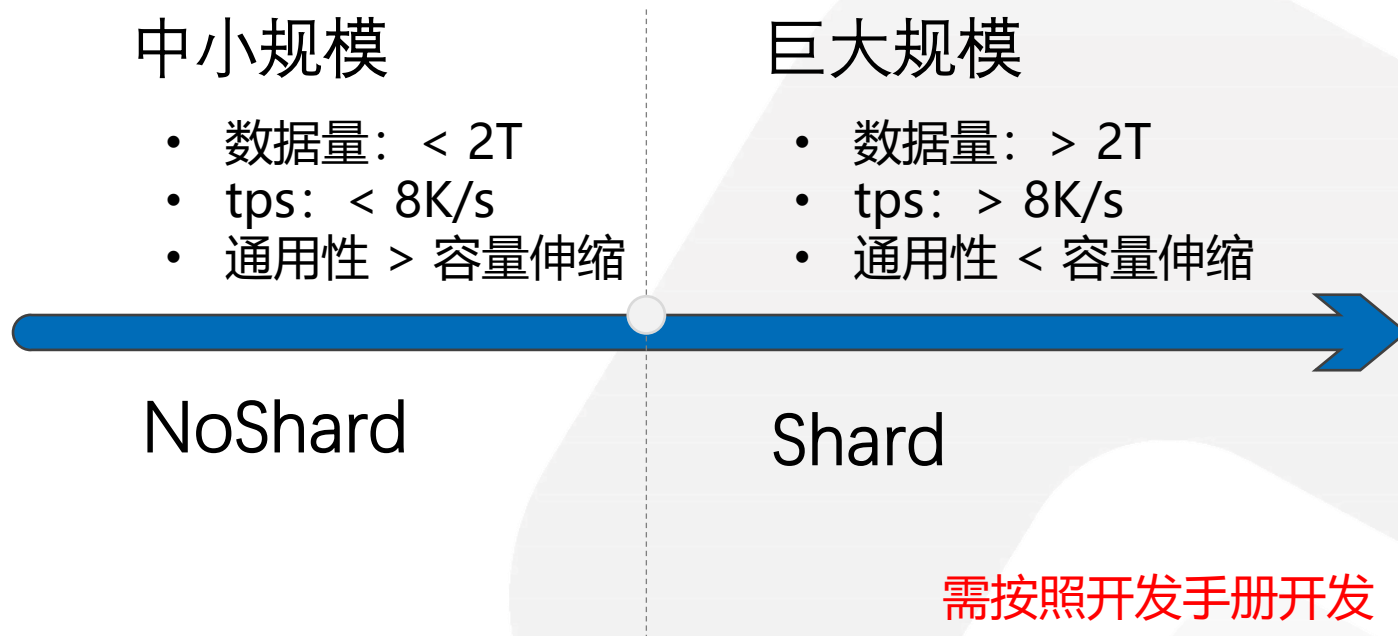
水平
伸缩

分布式总结

- 选型推荐

- NoShard: 非分布式版本, 使用起来完全和MySQL一样;
- Shard: 分布式版本, 适用单节点已经无法满足负载要求, 需要将数据分散到其他节点。

选择: 中小规模适用于no shard, 大规模适用于Shard

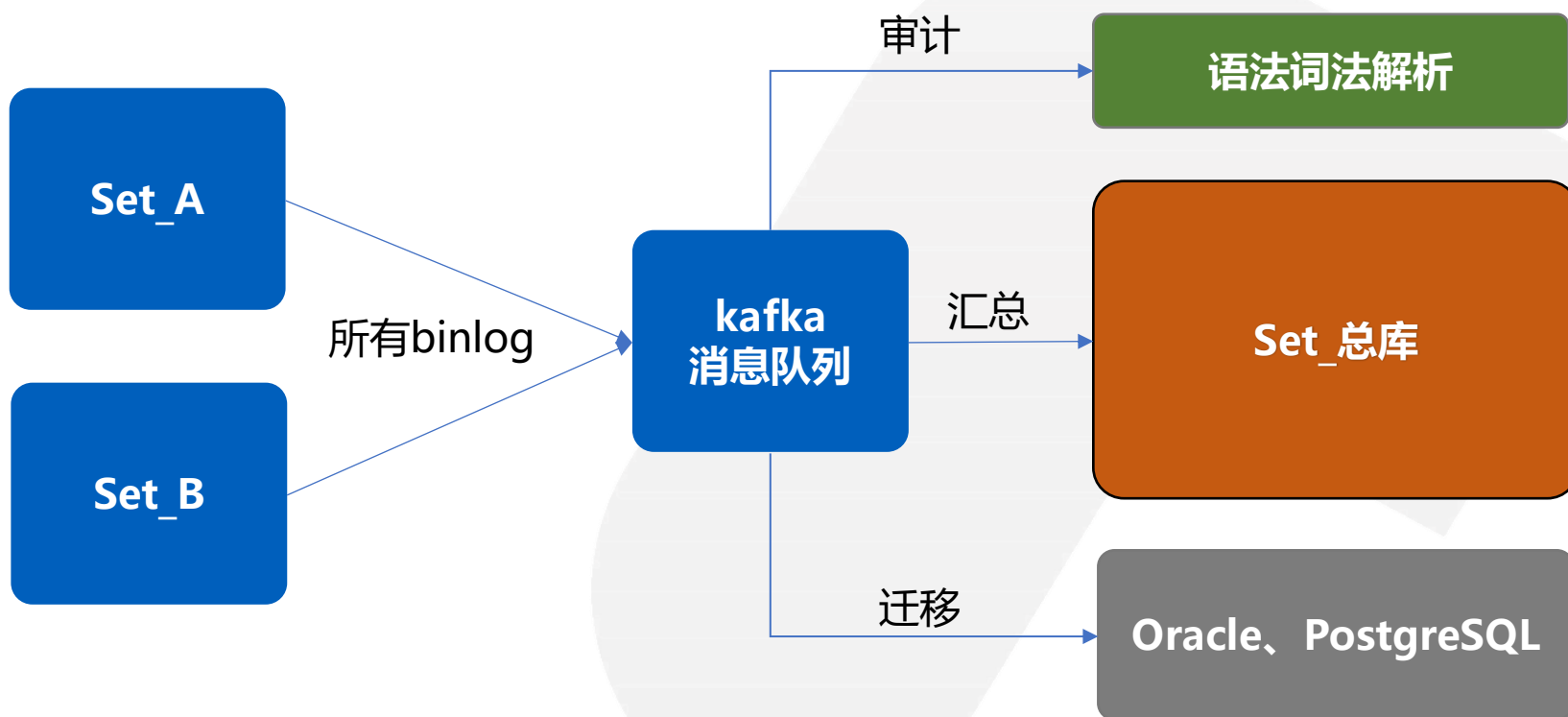


数据同步

- 应用场景

- 汇总：多个数据库实例的数据同步到一个数据库实例，适用于分支>总部子、系统>总系统等系统架构
- 迁移：异构数据库的迁移，将数据从TDSQL同步到MySQL、Oracle、PostgreSQL等数据库。

原理：将数据库日志灌入kafka，消费端读取kafka里的消息消费

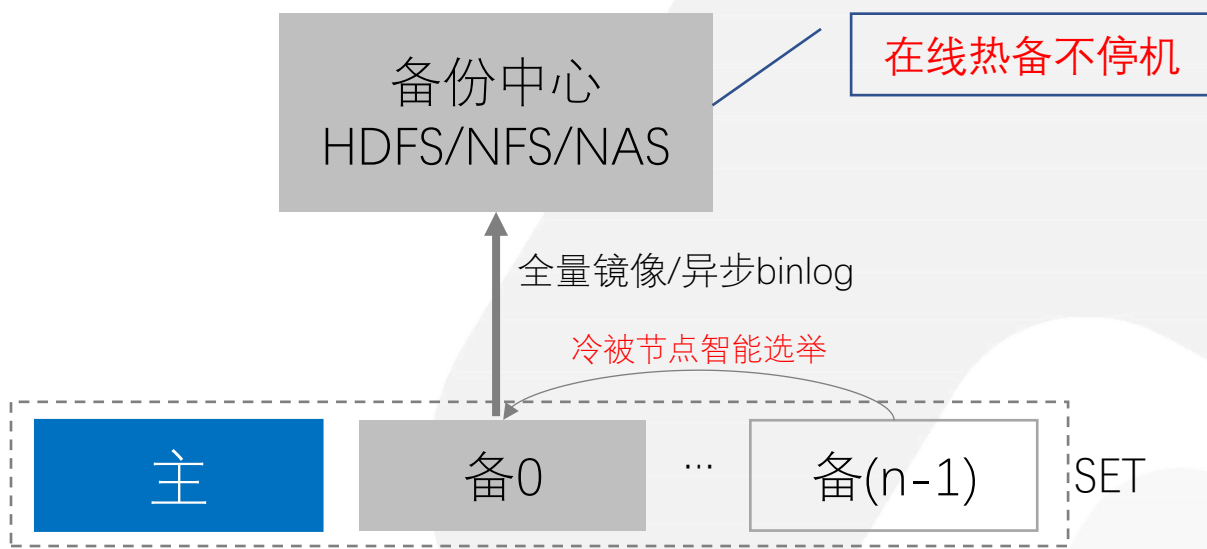


数据备份

- TDSQL备份体系

- 数据备份：采用增量+全量的备份方式。每天零点发起全量备份；增量通过实时备份binlog日志实现；
- 备份结果可知、备份过程可知。

注：全量备份支持物理、逻辑两种方式

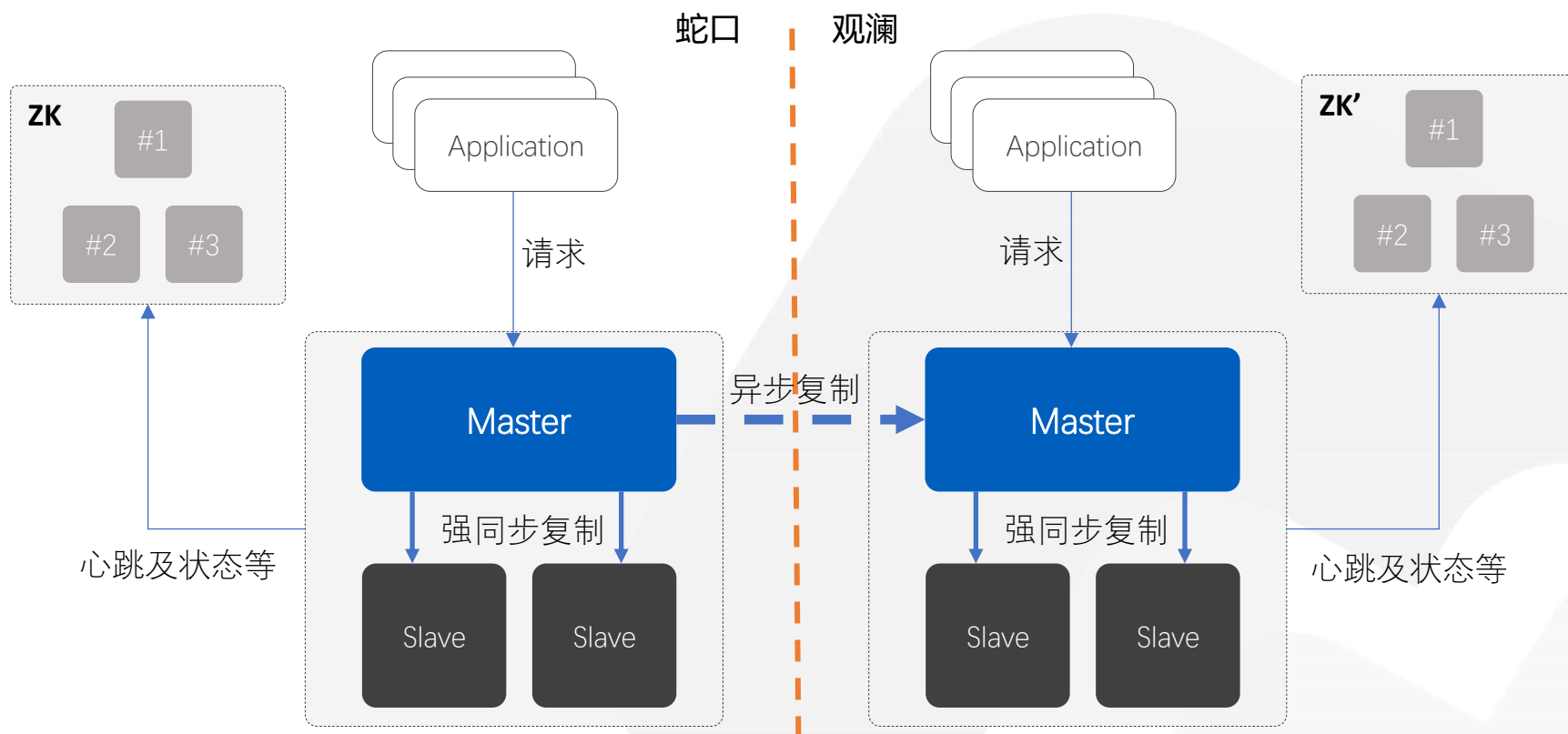




部署实践

同城双中心

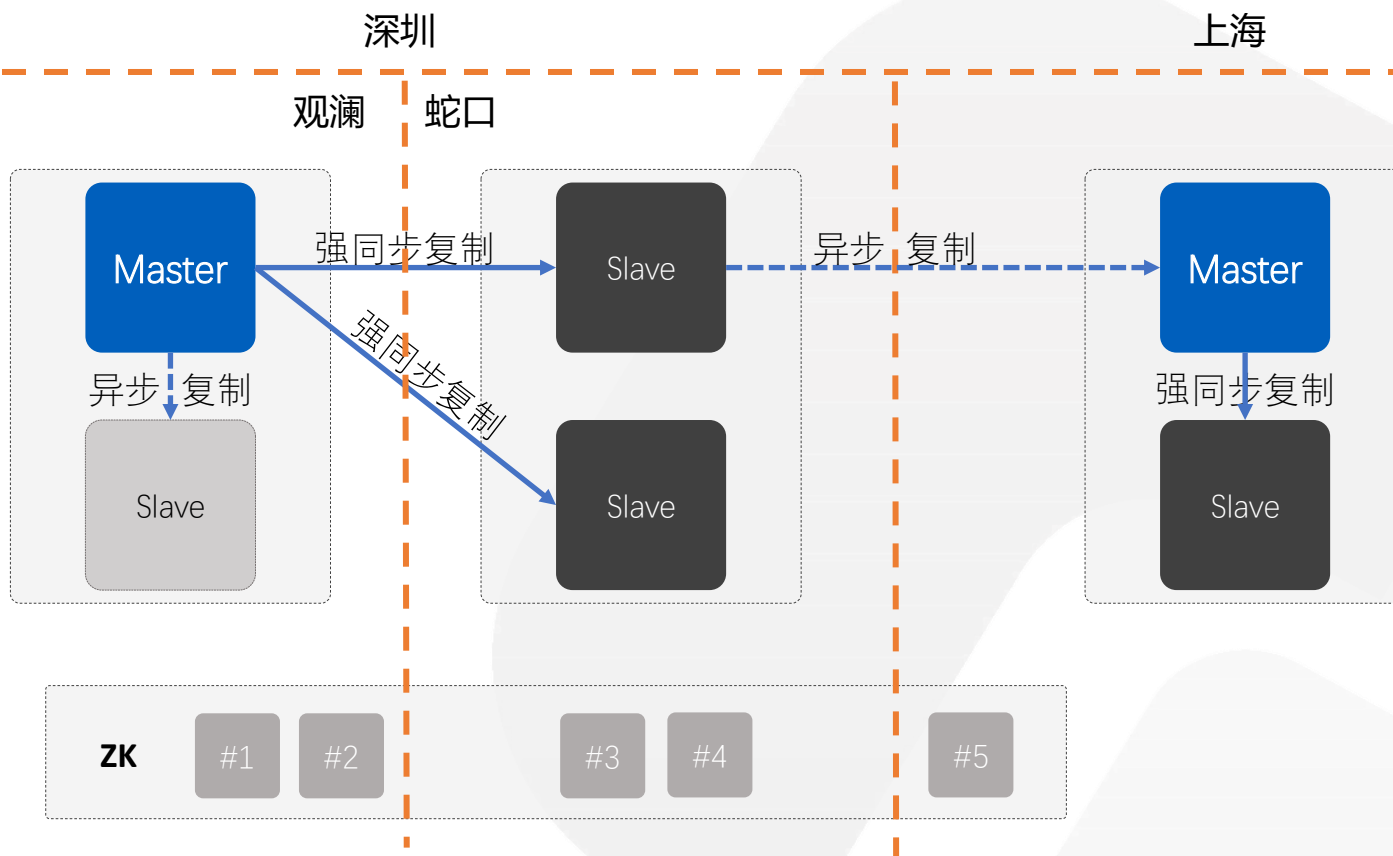
- 部署体系
 - 由于成本限制只具备两个机房，只能做到跨机架级别容灾；



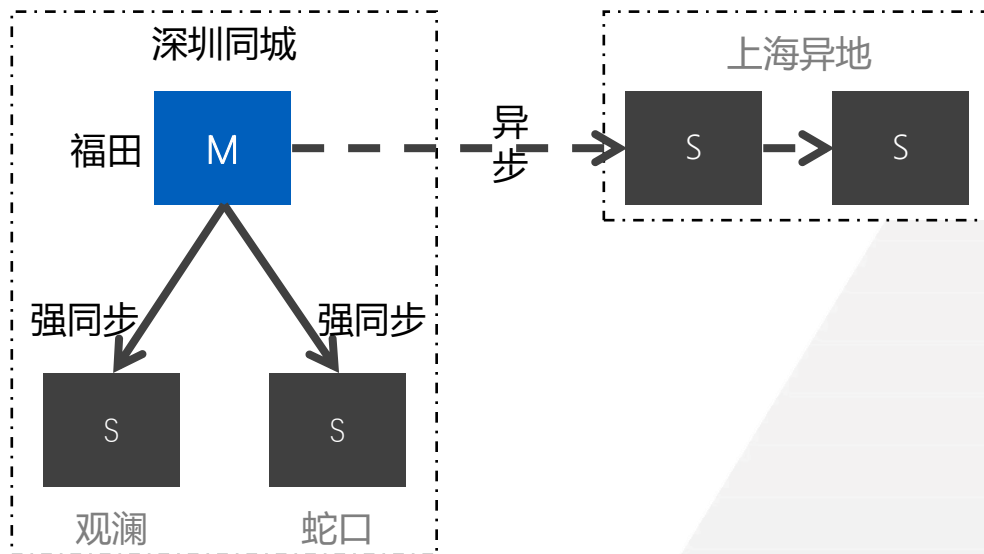
两地三中心

- 部署体系
 - 同城和异地分别有两个、一个IDC，经典的“两地三中心”部署架构，可以做到跨IDC级别容灾；

注：异地之间复制为异步复制



两地四中心



- 同城三中心集群化部署，简化同步策略，运营简单，数据可用性、一致性高
- 单中心故障不影响数据服务
- 深圳生产集群三中心多活
- 整个城市故障可以人工切换

常用术语回顾

- **集群**: 一个独立网络区域的TDSQL服务器集合集群; 一般来说, 先有集群才能在集群上分配实例;
- **SET**: TDSQL最小数据单元, 包含一主N备N+1个节点, 在TDSQL内部各模块之间通过set_xxxxx这样的名称标识一个SET;
- **实例**: 用户实际使用的一个最小单位的数据库服务集合。分为分布式实例和非分布式实例, 非分布式实例为单个SET, 分布式实例一般为多个SET的集合;
- **主机**: MySQL数据节点中, 直接承担读写的节点, 简称主机 (Master) ;
- **从机**: 通过主从复制协议和主机保持数据同步关系, 在主机不可用时替换主机, 简称从机 (Slave) ; 从机通常只可读取数据, 不可写入数据;
- **分片**: 分布式TDSQL中一个SET又成为一个分片;
- **Shardkey**: 路由键, 也叫分片关键字, TDSQL根据一个表中的shardkey对该表进行拆分。
- **Group**: 一个分布式实例又叫一个group, 在TDSQL内部常用group_xxxx 来标识一个group。
- **路由**: 保存在zk上的一组元数据信息, 用来指导网关将sql发往指定分片。

赤兔管理台





Thanks!