2019
05
08-10
北京新云南皇冠假日酒店

数据风云十年变迁

DTCC 第十届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA 2019

IT168 ChinaUnix ITPUB

银联自研分布式数据库：UPDRDB

银联科技事业部·操作系统及数据库团队
主管 周家晶

1 内容概要

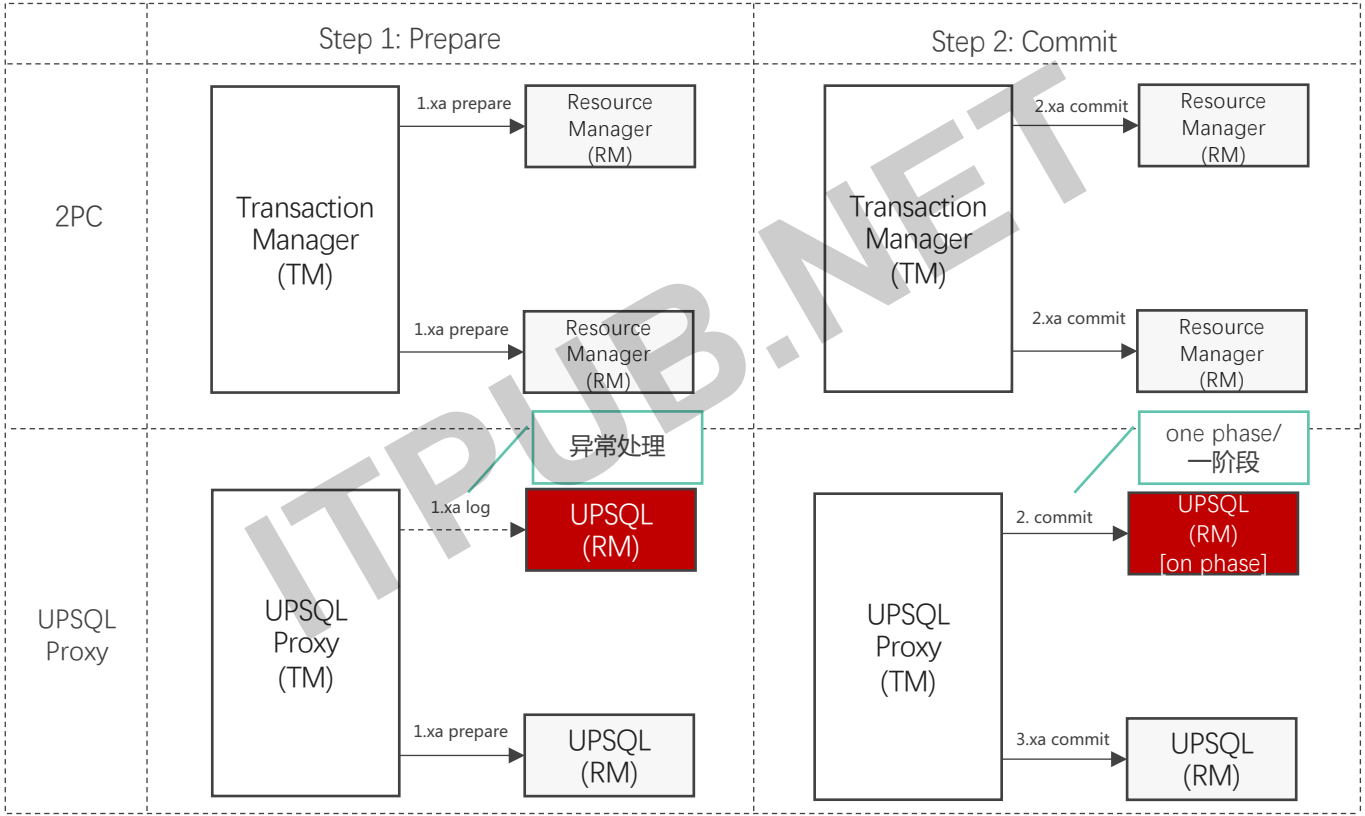
■UPSQL Proxy

- □2015年2月，一期研发：高可用、分库
  - 解决有无问题
  - 提供了高可用、读写分离、分库
- □2016年8月，二期研发：连接池、分布式事务
  - 分库性能优化
  - 分布式事务支持
  - 业务适配
- □内部已部署2000+实例
- □通过4年多的研发产品已趋于稳定，计划不再进行功能迭代

■UPDRDB

- □利用UPSQL Proxy优势(性能、事务管理器)，面向未来发展提出了的自主NewSQL方案
- □特点：
  - 保持OLTP的高性能、同时兼顾OLAP功能需求
  - 研发基于UPSQL Proxy的分布存储引擎XProxy
  - Proxy架构向分布式数据库架构演化
- □2018年开始研发
- □逐步替代UPSQL Proxy

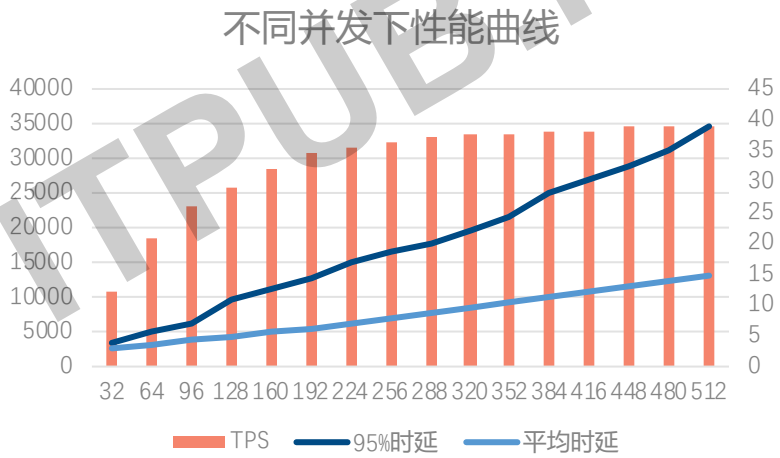
2. UPSQL Proxy / 二期功能 / 分布式事务



2. UPSQL Proxy / 二期功能 / 连接池

■连接池作用

- □避免频繁的建立连接、控制目的库负载
- □提高系统响应速度
- □分库性能扩展性的基石

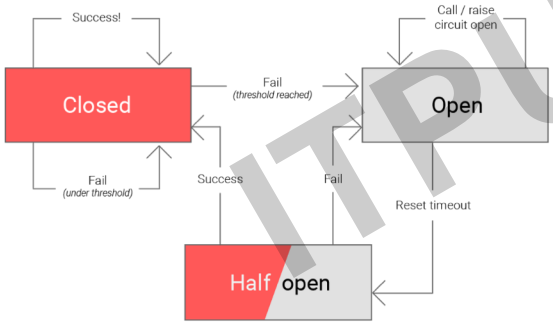


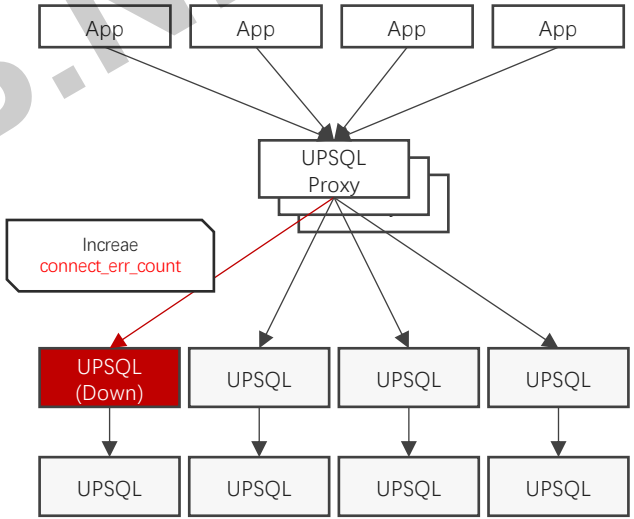
连接池配置为(16-128)

2. UPSQL Proxy / 二期功能 / 数据库熔断

■数据库熔断需求

□单分片处于hang故障时，避免hang住大量前端连接

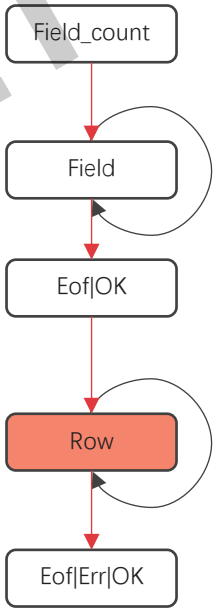




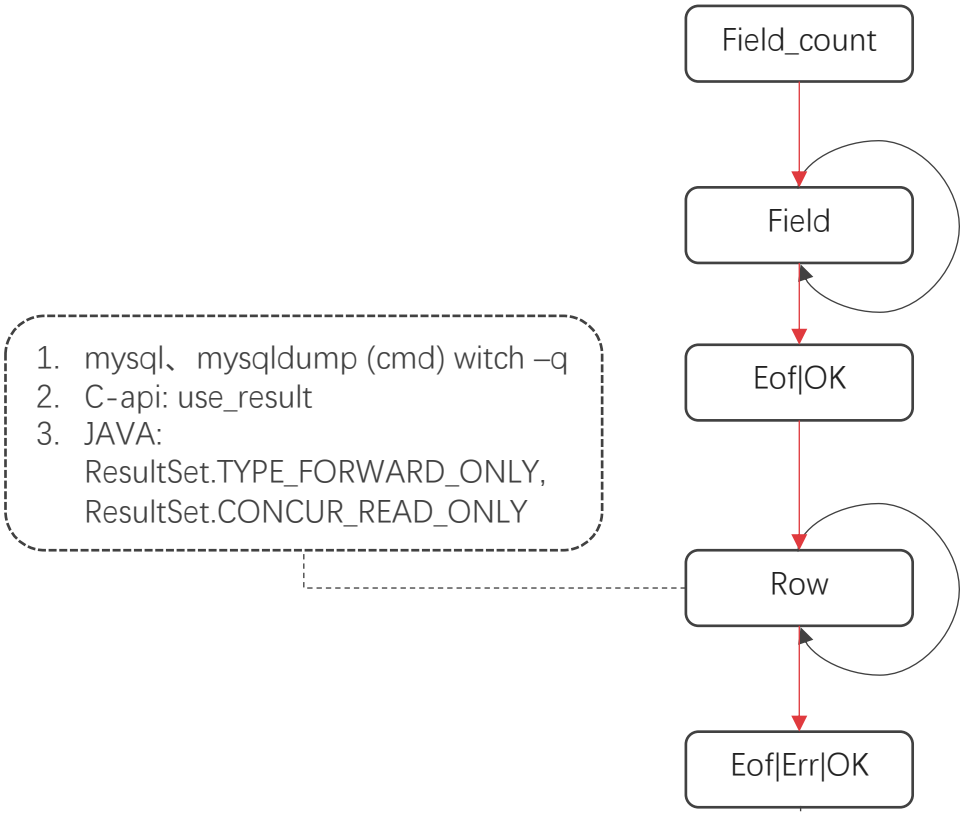
2. UPSQL Proxy / 二期功能 / 流式处理

■流式处理策略

- □MySQL应答包，包含多个报文，主要包含2个部分：
  - 表头信息：由多个报文组成
  - 行数：每至少一个报文
- □流式合并策略：
  - 每收到一批行数据后，立刻发送给请求方
  - 适合没有多分片聚合场景
  - 减少内存依赖
- □需求来源：
  - 日常数据导出
  - 对接Spark SQL



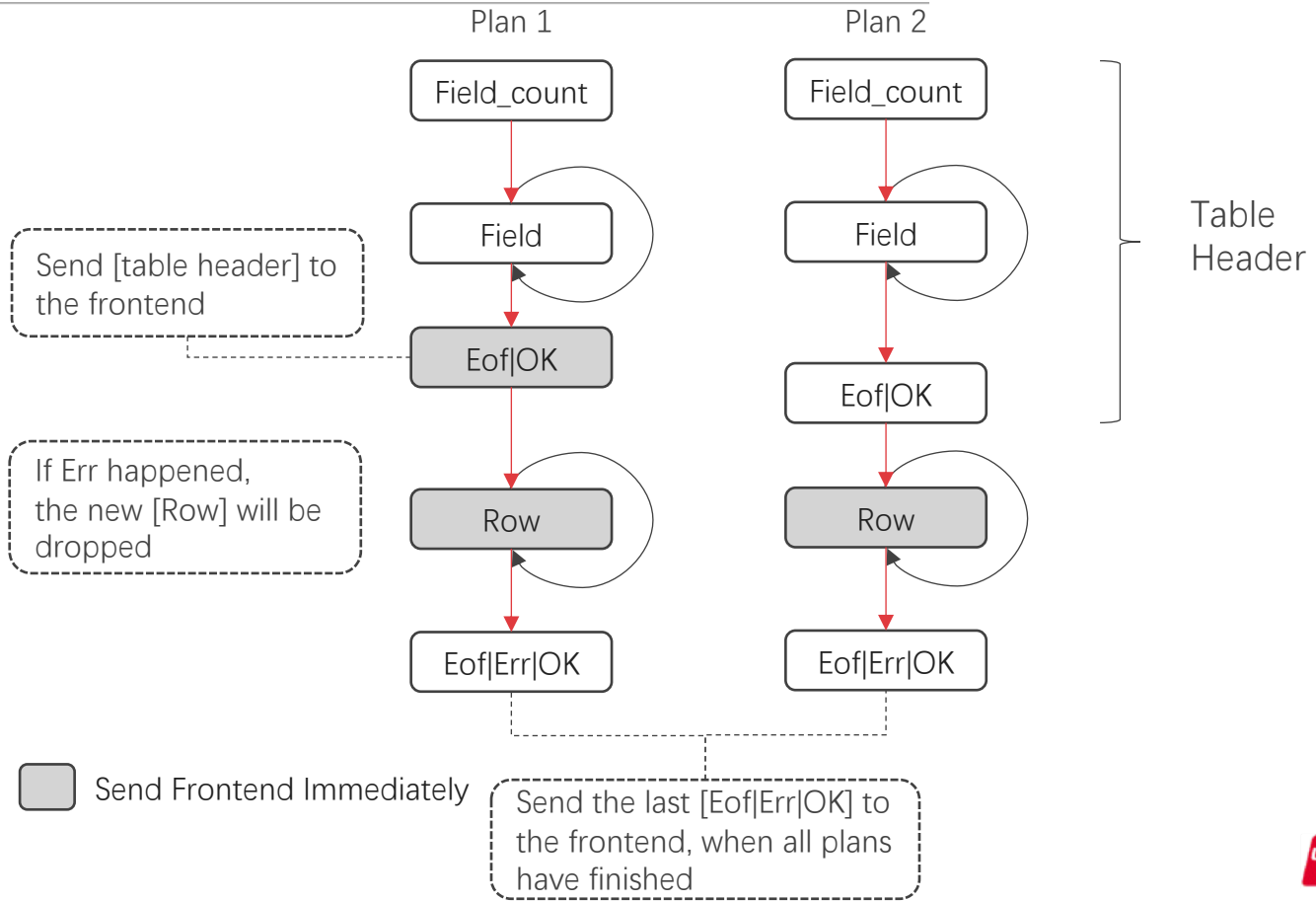
2. UPSQL Proxy / 二期功能 / 流式处理



1. mysql、mysqldump (cmd) witch –q
2. C-api: use_result
3. JAVA: ResultSet.TYPE_FORWARD_ONLY, ResultSet.CONCUR_READ_ONLY

1. mysql、mysqldump (cmd) without –q
2. C-api: store_result
3. JAVA: <normal>

2. UPSQL Proxy / 二期功能 / 流式处理



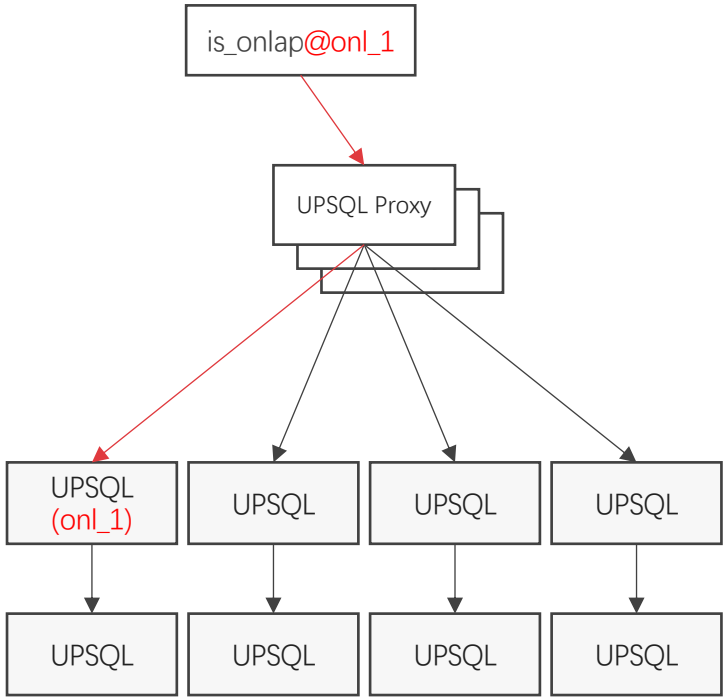
2. UPSQL Proxy / 功能不足 / 不支持单点DDL

■主要影响

- □需要用户对各个数据节点进行DDL操作
- □对分库配置等有学习和运维成本

■缓解措施

- □避免直连数据库DDL操作
  - @datanode方案
  - 连接属性方案
- □支持分布式truncate



2. UPSQL Proxy / 功能不足 / 其他

不支持复杂查询

■主要影响

- 不支持垮库Join
- 有限支持聚合函数
- 不支持子查询
- 等

■改进措施

- 有限支持聚合函数
  - agg、sum等
  - 并有额外格式要求
- 有限支持排序
  - 不支持中文字符
  - 只支持固定字符序

在线扩缩容能力缺失

■主要影响

- 资源分配灵活性不足
- 给应用团队带来了额外工作

■改进措施

- 提供了一种基于moray的扩容方案
  - 需要短暂中断交易
  - 资源浪费
  - 全量数据搬迁，扩缩容效率低

3. 对标分析 / 产品列表

- **腾讯：**
  - TDSQL (DCDB)
- **阿里：**
  - DRDS
  - PetaData (HybridDB for MySQL)
- **MariaDB：**
  - Spider
- **PingCAP：**
  - TiDB
- **巨杉**
  - SequoiaDB
- **亚马逊、阿里**
  - Aurora、PolarDB

1.数据存储：MySQL
2.高可用：MySQL复制

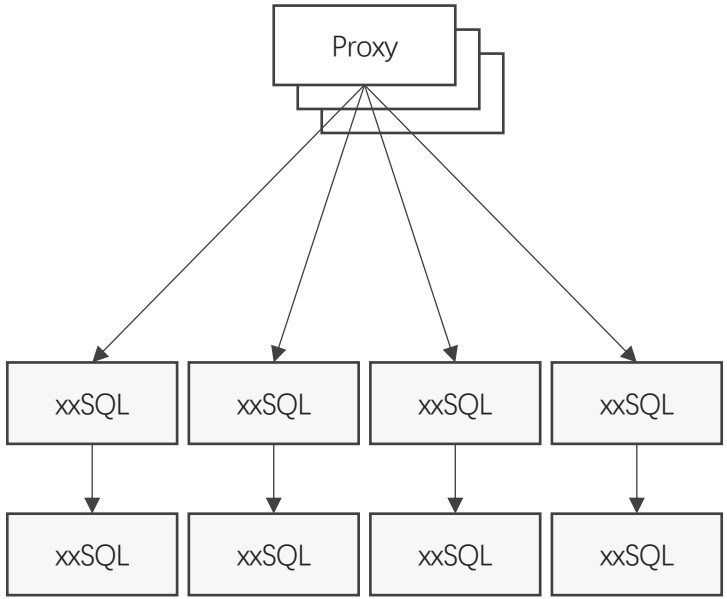
1.数据存储：KV
2.高可用：Multi-Raft

1.数据存储：KV
2.高可用：类似MySQL复制
3.分布式存储引擎

1.单点写、多点读
2.存储与计算分离

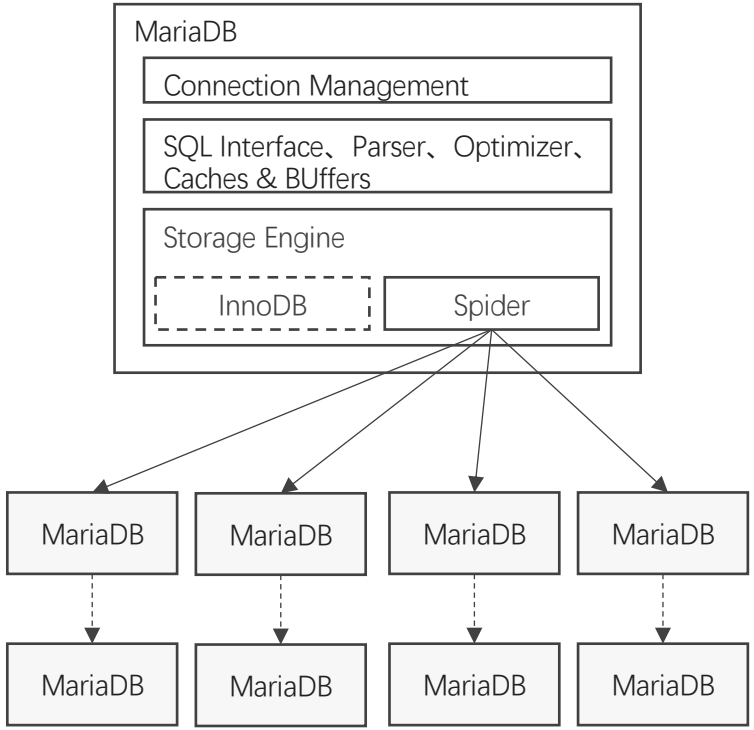
3. 对标分析 / TDSQL & DRDS

- **技术路线**
  - 丰富完善Proxy层的SQL解析能力
- **主要技术状态**
  - 分布式事务支持
  - 复杂语句
    - 聚合类有限支持
    - 不支持部分数据类型



3. 对标分析 / MariaDB Spider

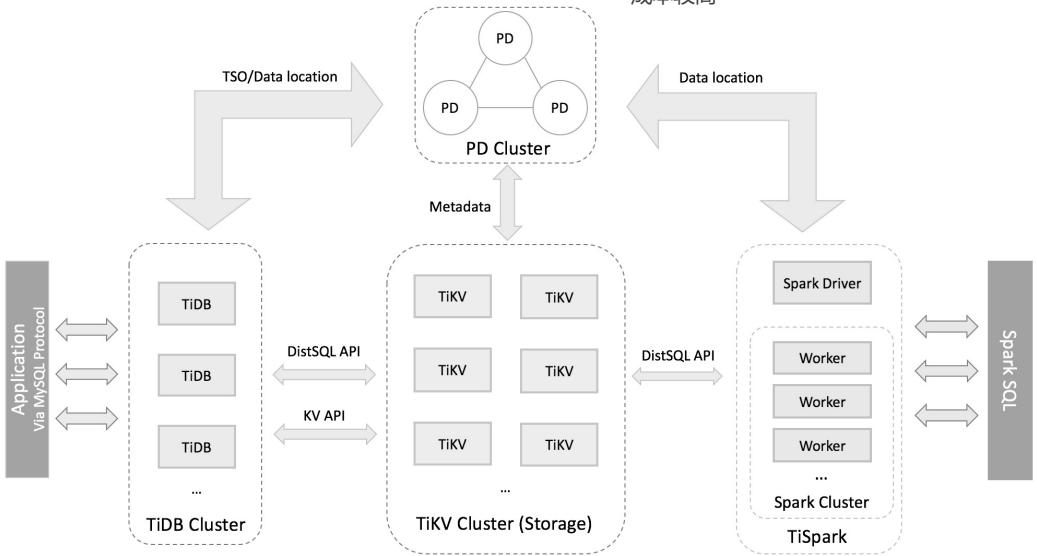
- **技术路线**
  - 实现分布式的存储引擎
- **主要技术状态**
  - 支持分布式事务
  - 复杂语句
    - 支持复杂语句
    - 并增加特性
  - DDL支持
    - 不支持单点DDL
  - 性能：
    - 与数据节点之间为同步调用
    - 集群性能低于单机MariaDB



3. 对标分析 / TiDB

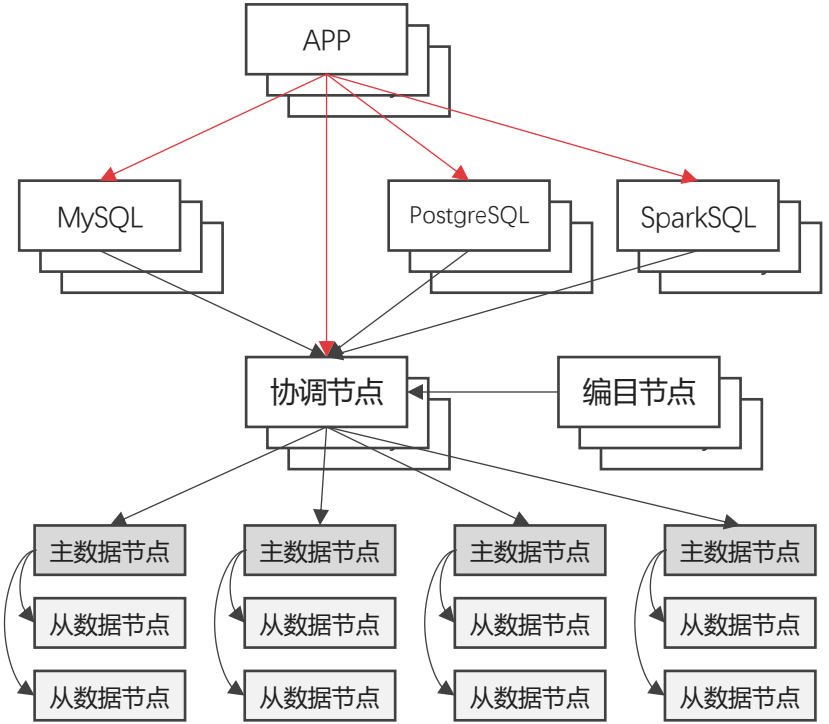
- **技术路线**
  - 借鉴Google Spanner和F1
  - 使用KV进行数据存储
  - 使用Multi-Raft实现高可用与数据迁移

- **主要技术状态**
  - 支持分布式事务
  - 乐观锁机制：不适合热点数据
  - 较好的MySQL兼容性
  - 时延较高
  - 成本较高



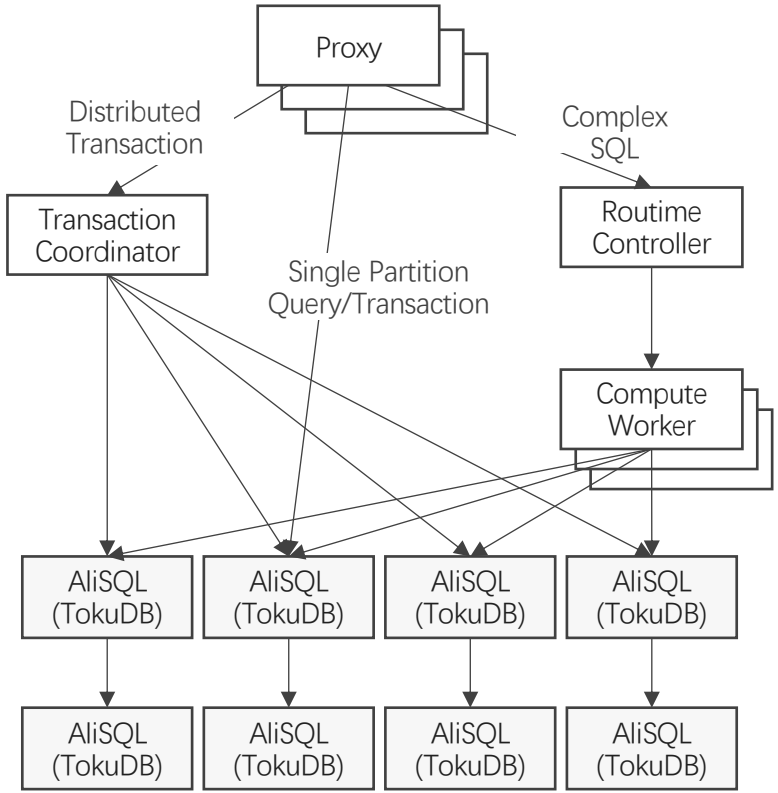
3. 对标分析 / SequiaDB

- **技术路线**
  - 使用KV进行数据存储
  - 实现分布式存储引擎
- **主要技术状态**
  - 支持分布式事务
  - 支持复杂语句
  - ~~悲观锁，不适合高并发场景~~(3.2版本已改进)
  - 较好的MySQL语法兼容
  - 支持多种xSQL



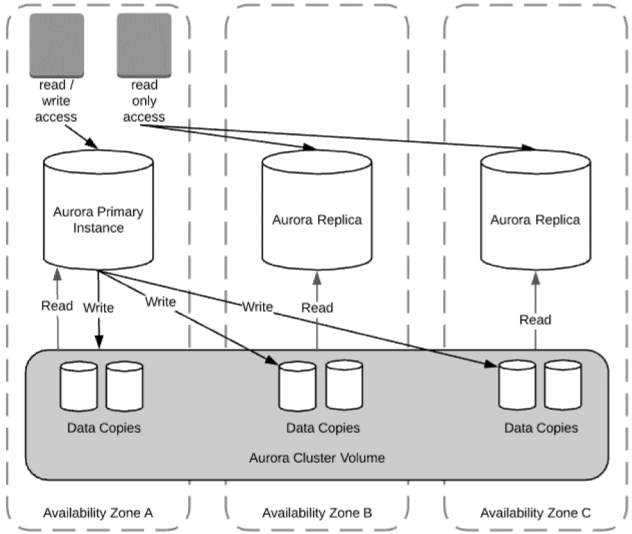
3. 对标分析 / PetaData

- **技术路线**
  - 通过Proxy层进行复杂语句路由
  - OLTP与OLAP混合
- **主要技术状态**
  - 分布式事务
    - 简单语句支持分布式事务
    - 复杂语句目前不支持事务
  - 复杂语句
    - 支持MySQL单机语句
    - 增加了OLAP特性语法支持
      - TPC-H
      - TPC-DS



3. 对标分析 / Aurora、PolarDB

- **技术路线**
  - 单点写、多点读
    - 多点写研发中
  - 存储计算分离
  - 分布式文件系统
  - 用户态读写网络、文件
- **主要技术状态**
  - 读性能线性扩展
  - 写性能可以达到MySQL单机性能的6倍以上

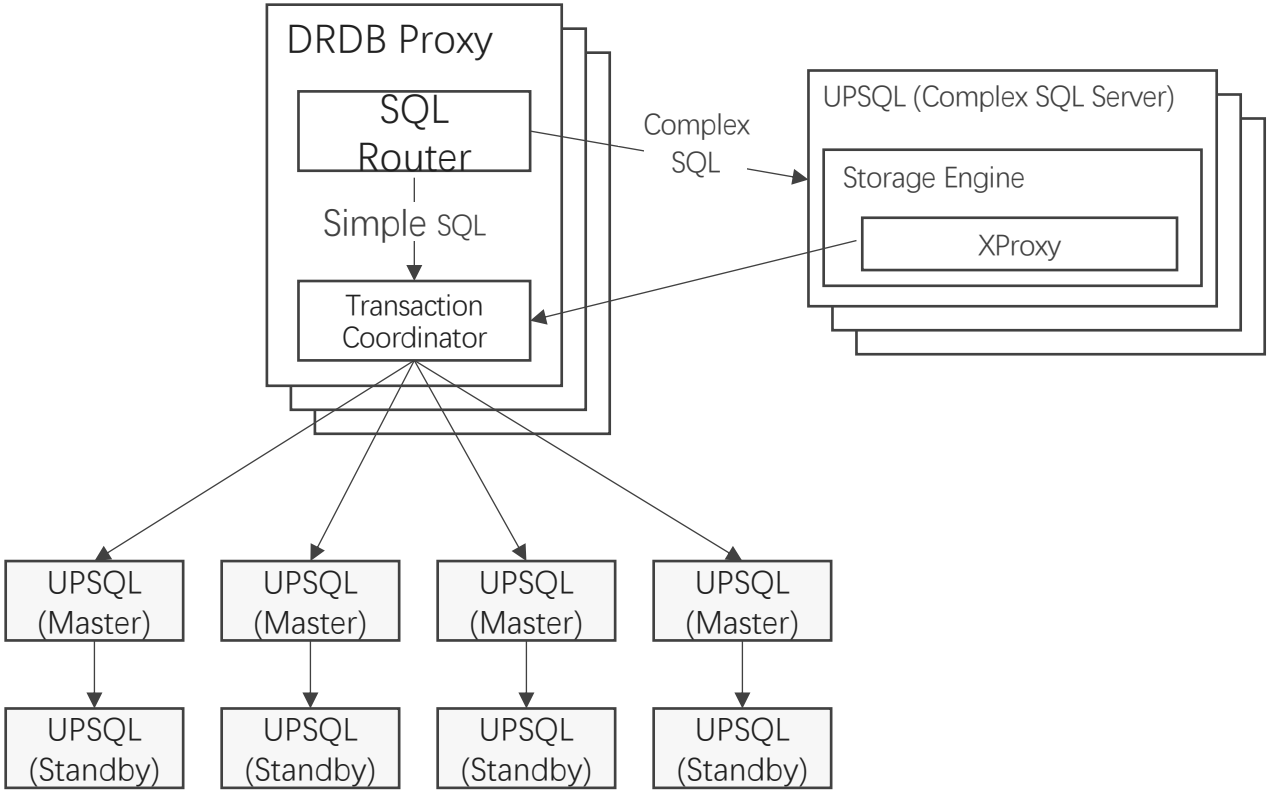


3. 对标分析 / 综述

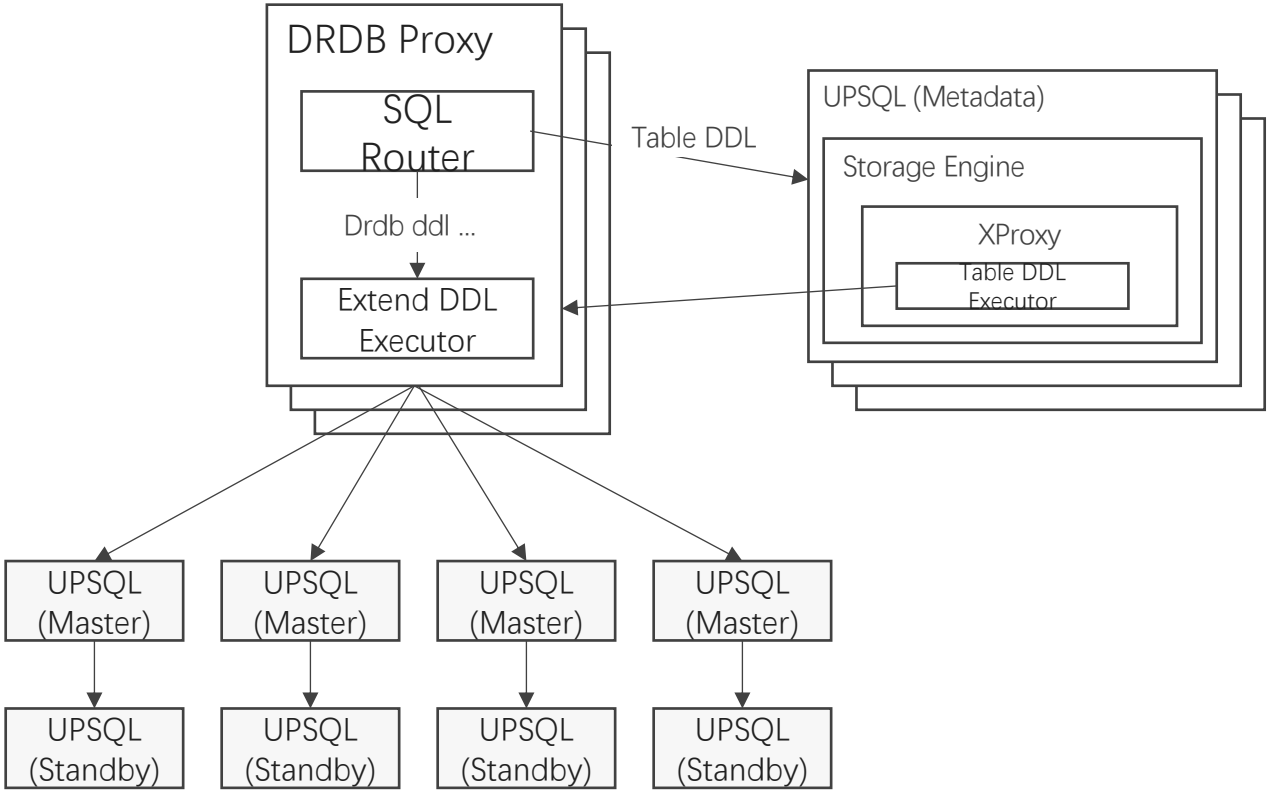
- **技术路线分类**
  - SQL解析层的自主化程度
    - 完全自研：TiDB、PetaData、TDSQL、DRDS等
    - 利用MySQL Server： MariaDB Spider、 SequoiaDB
  - 分布式存储引擎的自主化程度
    - 完全自主化： SequoiaDB
    - 接近完全自主化：TiDB （ TiKV底层使用了RocksDB ）
    - 使用MySQL Server: PetaData、TDSQL、DRDS等
- **区别**
  - SQL解析的自主化程度越高，其OLAP可能性就越高，如PetaData在OLAP领域功能更为强大
  - 分布式存储引擎的自主化程度越高，其架构的灵活性就越高，如SequoiaDB可以对接多种Server

- **UPDRDB的选择**
  - 面对应用和运维需求，有2条可选技术路线
    - 1.丰富SQL解析层，深化分布式执行计划
    - 2.旁路复杂语句和DDL，实现一个旁路处理子系统
  - 经过评估更深入的自主化SQL与分布式执行计划在工程上风险巨大，方案2风险小、收益大，在功能架构上做了解耦，能显著提高整体的稳定性
- **UPDRDB的设计思路**
  - 学习PetaData与SequoiaDB的优点，遵循业界经验，扬长避短
    - PetaData：通过Proxy进行语句路由，复杂语句路由给OLAP服务器，但复杂语句不支持事务
    - SequoiaDB：协调节点提供更高性能，xSQL提供不同的SQL特性与计算模型
  - 总体方案：
    - 由UPSQL Proxy进行语句分类路由，并由UPSQL Proxy实现事务管理，保证简单语句和复杂语句的事务性
    - 在UPSQL上实现XProxy引擎， XProxy访问UPSQL Proxy实现分布式存储

4. UPDRDB / 架构 / DML



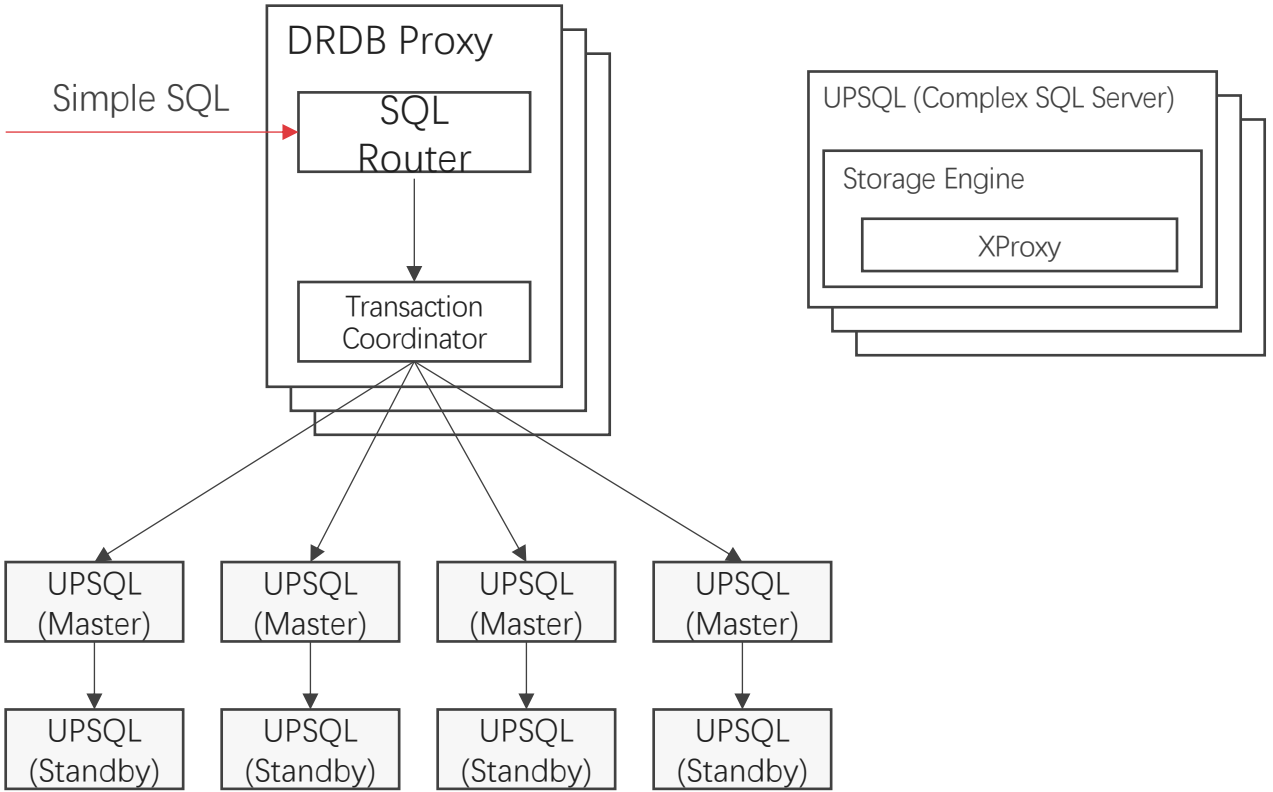
4. UPDRDB / 架构 / DDL



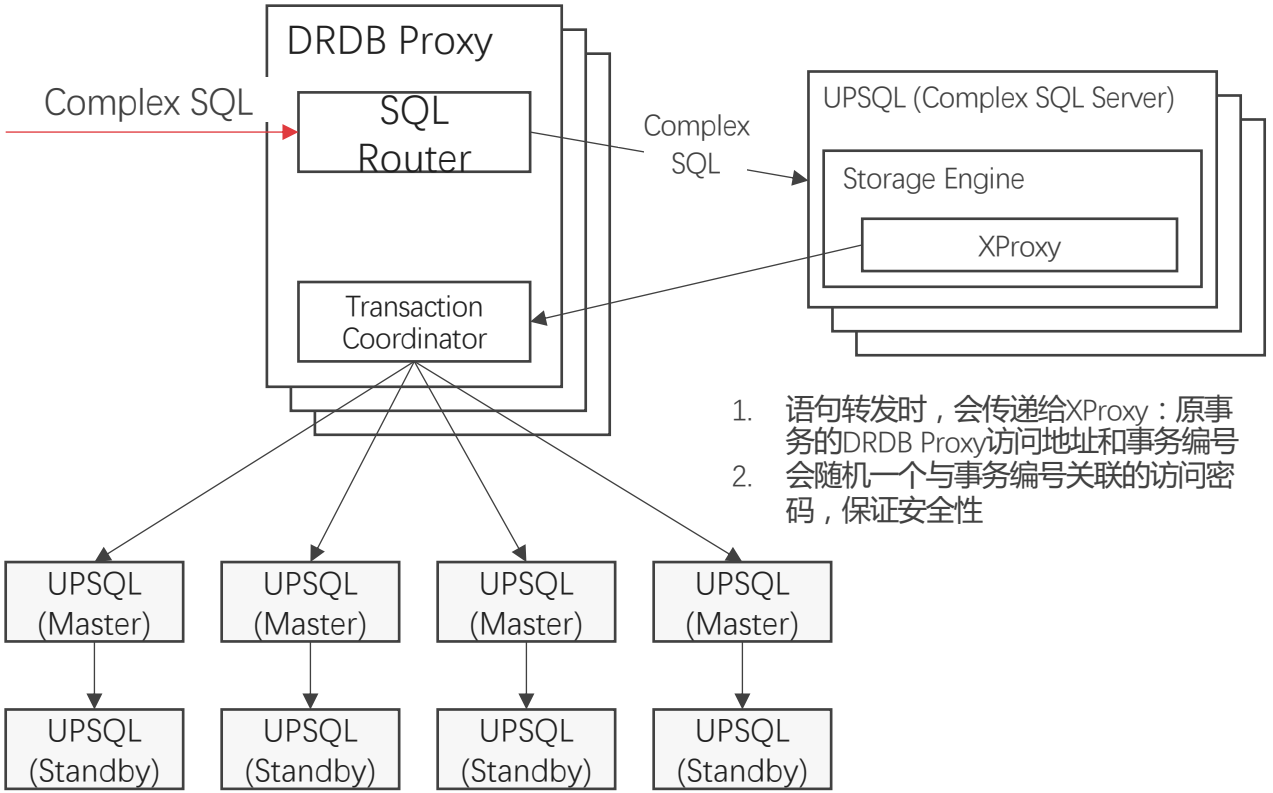
4. UPDRDB / 关键方案 / 语句路由与事务

- **语句路由与事务管理**
  - 语句路由与事务管理，是方案整体方案的前提
  - 核心由DRDB Proxy进行分布式事务管理
- **语句路由规则(概要)**
  - 路由到proxy
    - 简单DML，一般为插入更新、无聚合操作的单表查询
    - 事务管理：begin、commit
    - set语句
    - 部分DDL
  - 路由到Metadata
    - 复杂DML，一般为多表、有聚合函数的单表查询
    - 大部分DDL
    - show语句

4. UPDRDB / 关键方案 / 语句路由与事务



4. UPDRDB / 关键方案 / 语句路由与事务



1. 语句转发时，会传递给XProxy：原事务的DRDB Proxy访问地址和事务编号
2. 会随机一个与事务编号关联的访问密码，保证安全性

4. UPDRDB / 关键方案 / 语句路由与事务

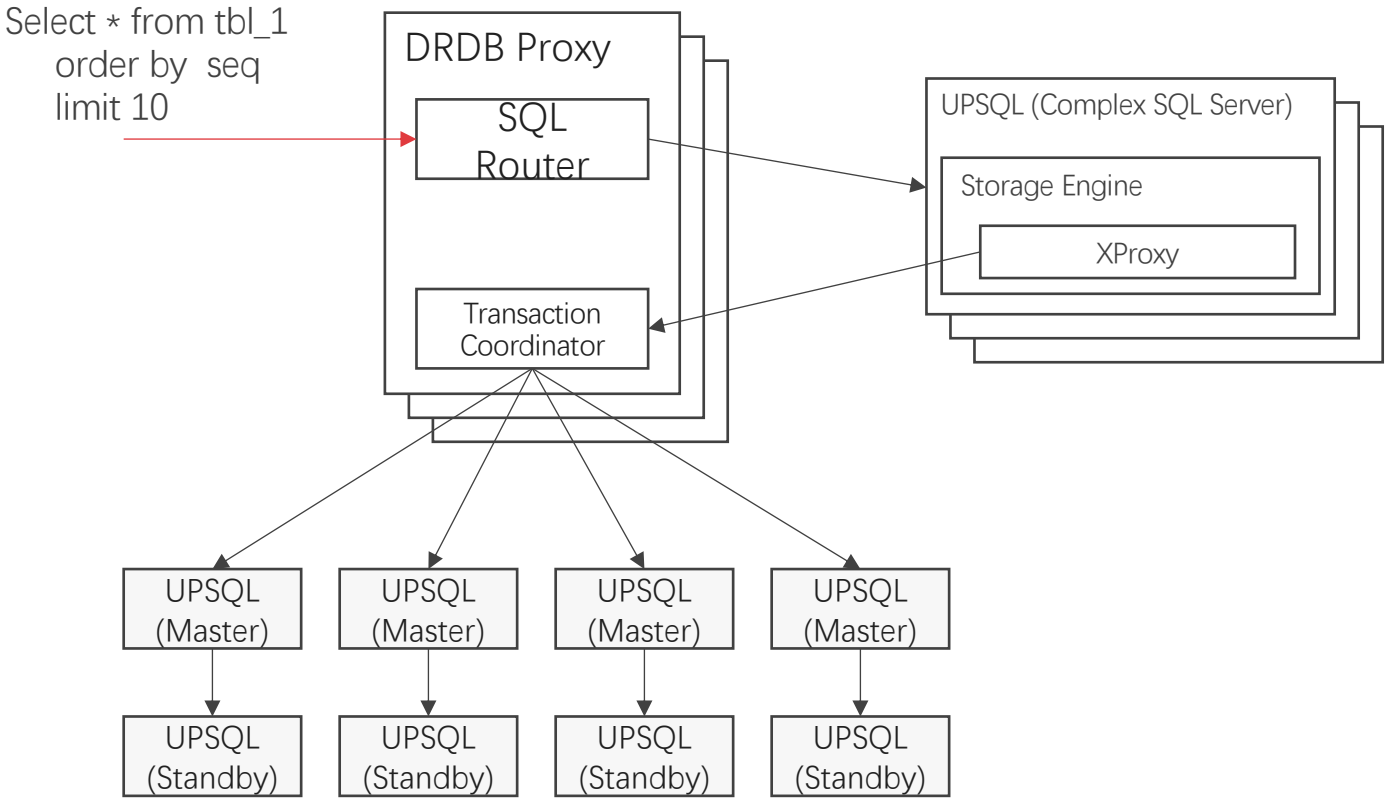
示例：

```
mysql> Begin;                                                    /*-> proxy */
Query OK, 0 rows affected (0.00 sec)

mysql> Insert into tbl_1 (id, name, age) values (1, “Jack”,  20);    /* -> proxy */
Query OK, 1 rows affected (0.01 sec)

mysql> Insert into tbl_2 values
       (2, “Mike”,  21), (3, “Lucy”,  20), (4, “Tony”,  24) ;          /*-> metadata */
Query OK, 3 rows affected (0.01 sec)

mysql> Select * from tbl_1
                  inner join tbl_2 on tbl_1.age = tbl_2.age;          /*-> metadata */
+-----+------+-----+----+-------+------+
|  id | name | age | id |  name |  age |
+-----+------+-----+----+-------+------+
|  1  | Jack |  20 |  2 |  Lucy |  20  |
+-----+------+-----+----+-------+------+
1 row in set (0.15 sec)

mysql> Commit;                                                   /*-> proxy */
Query OK, 0 rows affected (0.00 sec)
```

4. UPDRDB / 关键方案 / 分布式存储引擎 / DML

- **技术路线**
  - 分布式存储引擎由XProxy与DRDB Proxy组合实现
- **技术要点**
  - 实现了for update 和 lock in share mode锁
  - 全局表状态信息获取(show table status)
    - DRDB Proxy提供了并发执行show table status
    - XProxy进行信息聚合
  - Replace处理
  - 自增列适配
    - XProxy转发的写操作不带自增列
  - 排序与数据获取
    - 由DRDB Proxy提供并发查询能力
    - XProxy每次获取一定批次的数据(例如：每个分片10行数据)
    - XProxy进行排序运算和进度控制(例如：只推进部分分片的查询)

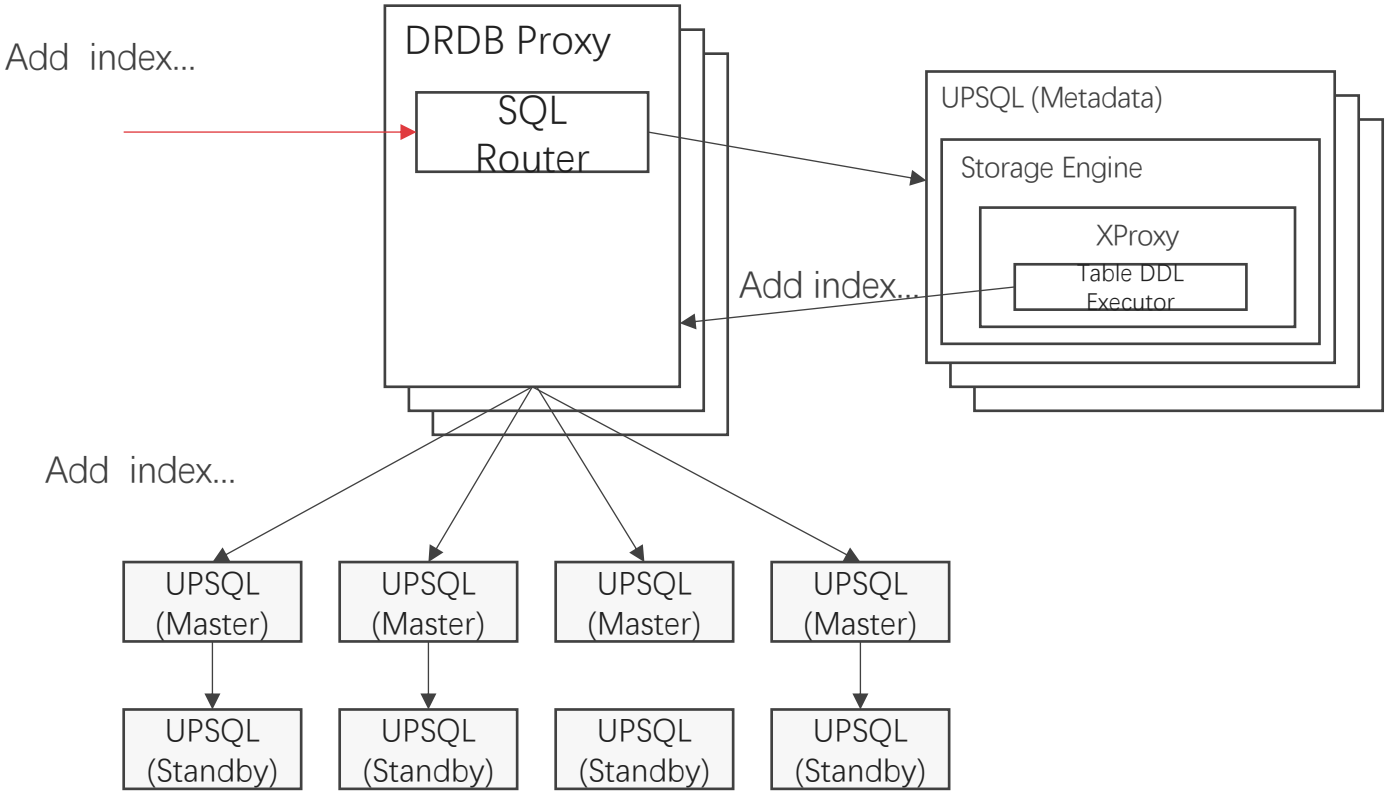
4. UPDRDB / 关键方案 / 分布式存储引擎 / DML



4. UPDRDB / 关键方案 / 分布式存储引擎 / DDL

- **技术路线**
  - XProxy通过Proxy访问各个数据节点进行DDL变更
- **技术要点**
  - 实现表级DDL
  - 表与表之间DDL可以并发执行
  - 单表DDL需要串行处理
  - DLL失败时，根据语句类型和执行状态，需要：
    - 必须重复执行，如：增加索引，部分节点成功部分失败
    - 忽略错误，如truncate失败
  - 提供drdb ddl status语句，用于查询DDL执行状态和出错处理方法

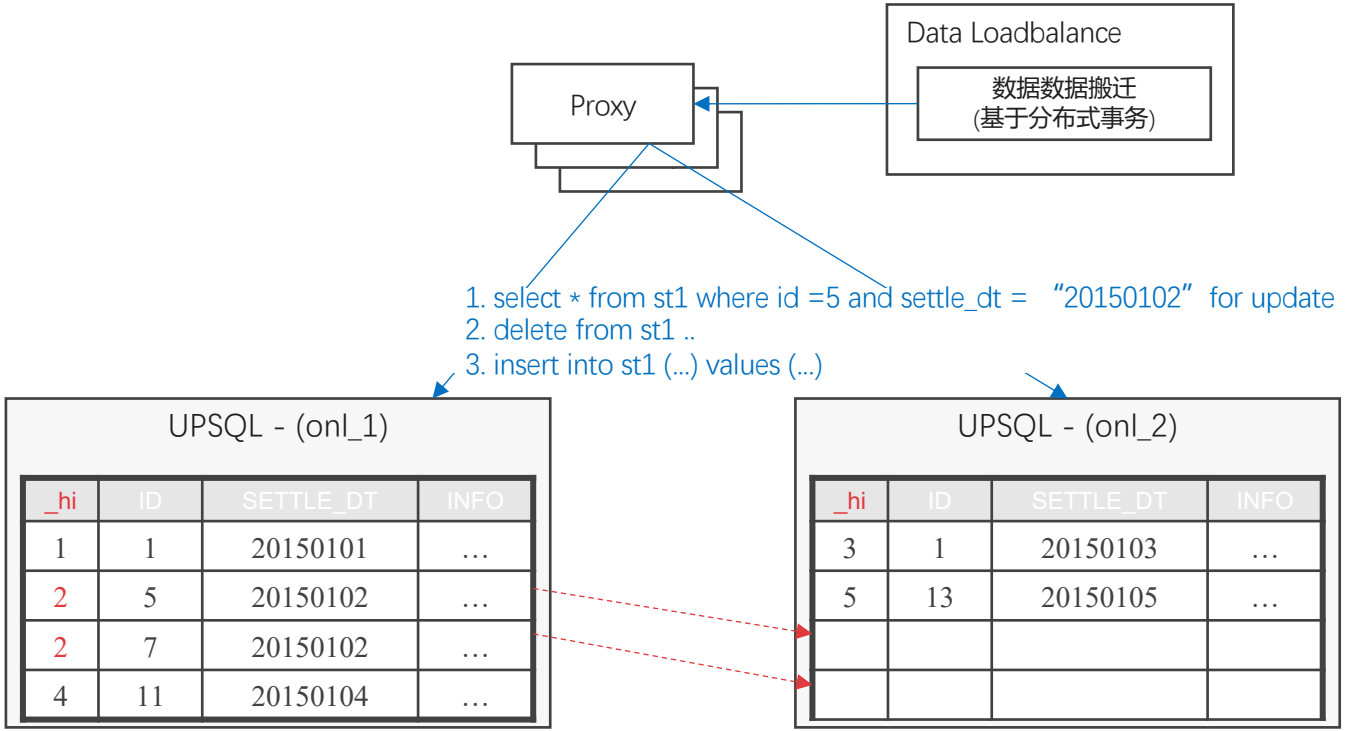
4. UPDRDB / 关键方案 / 分布式存储引擎 / DDL



4. UPDRDB / 关键方案 / 自动扩缩容

- **技术方案**
  - 将数据分成1024个逻辑组
    - 为标识和统计数据分组信息，在表上增加隐藏列“ _hash_index”
    - 隐藏列为数据的逻辑组；分库计算hash值模1024后得到
  - 数据分布规则，实际上就是每个逻辑分组与数据节点的关联关系，如
    - 逻辑分组0，分布在数据节点onl_1上
  - 扩缩容时，产生两套分库规则：原规则和目的(新)规则
    - insert访问新规则，其他操作访问2个规则的并集
  - 后台程序进行进行数据搬迁
- **使用方式**
  - 通过分布式DDL命令触发: drdb data loadbalance

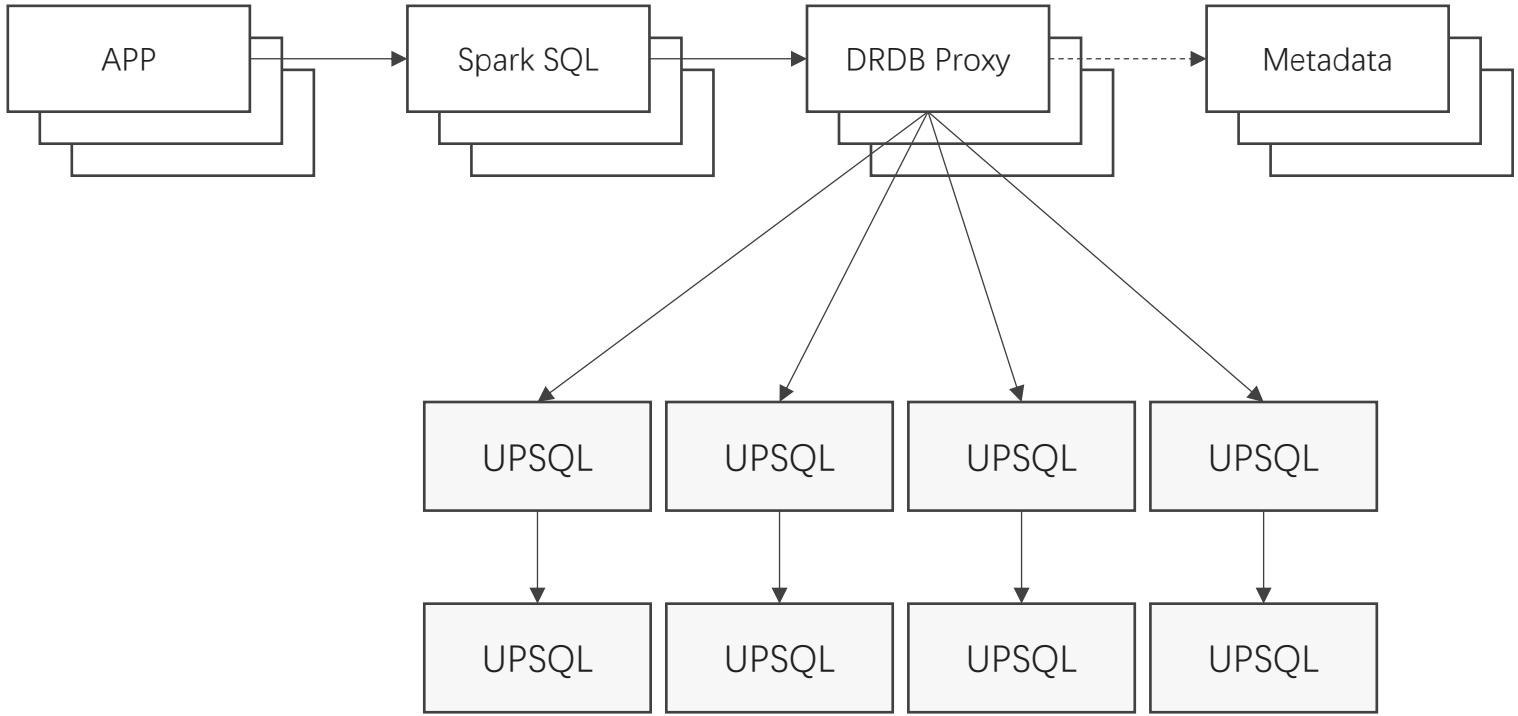
4. UPDRDB / 关键方案 / 自动扩缩容



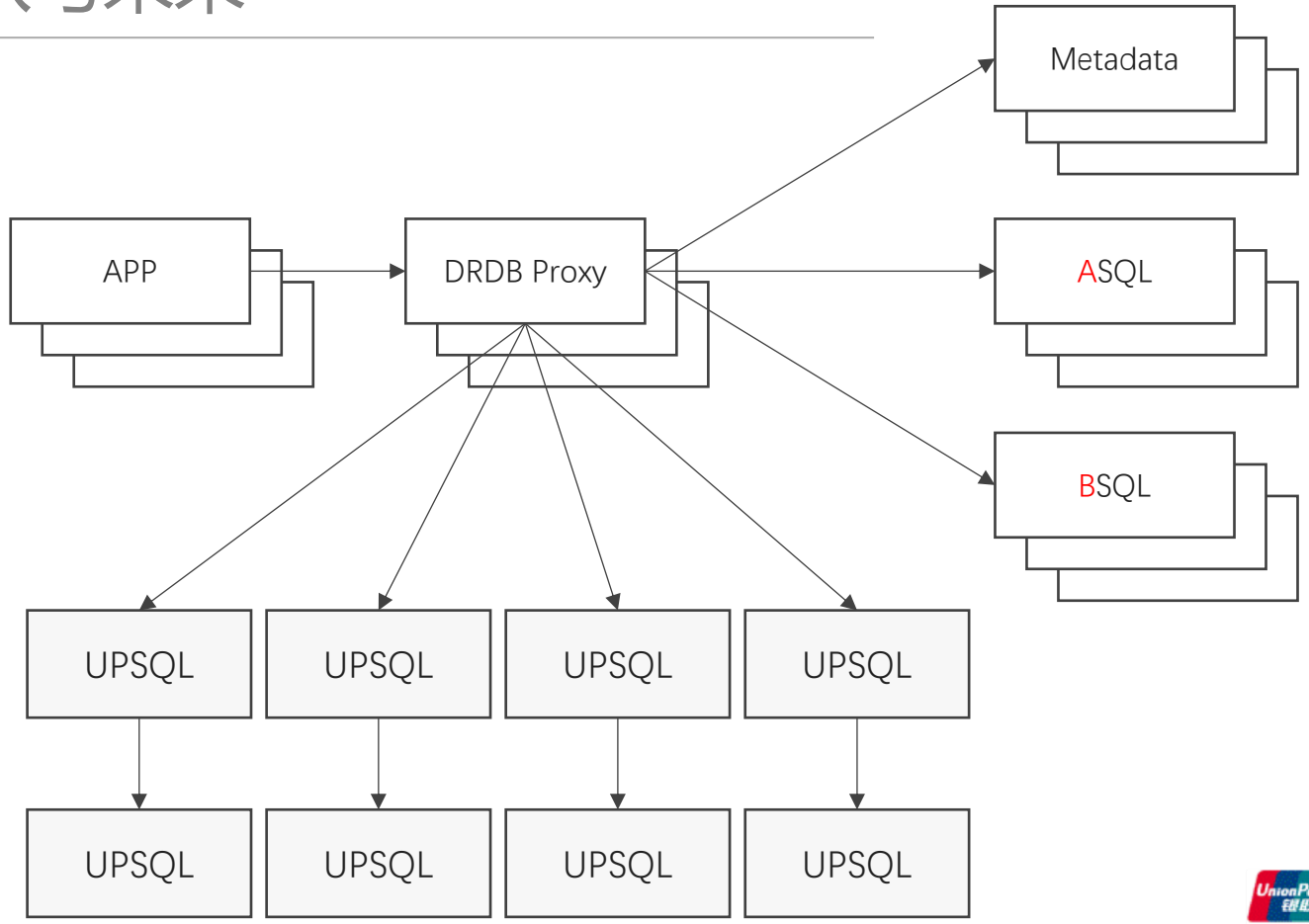
4. UPDRDB / 综述

- **产品定位**
  - 是UPSQL Proxy的架构升级(3.0)，是Proxy架构向分布式数据库的演进
  - 符合当前业界技术发展趋势，符合团队自身技术阶段
  - 以联机场景为核心，兼顾中台业务需求，提供一定T+0大数据的处理能力
- **UPDRDB优缺点**
  - 优点：
    - 简单语句高性能、低时延
    - 支持4种事务隔离级别
    - 对硬件资源要求较低
    - 自主可控
  - 缺点：
    - 依赖MySQL/MariaDB架构（典型的依赖其SQL解析层，难以实现并发执行）
    - 复杂语句效率较低
    - 不支持全局快照
    - 支持工具不丰富(管理子系统缺失等)
- **研发目的**
  - 解决应用的主要痛点：复杂查询、自动扩缩容、分库配置等功能支持不足
  - 解决运维的主要痛点：单点DDL
  - 解决UPSQL Proxy产品研发痛点：产品功能设计耦合
- **UPDRDB应用场景**
  - 覆盖当前UPSQL Proxy的使用场景，功能上更为丰富
  - 以OLTP场景为主：
    - 简单语句为主
    - 可含有少量复杂语句
  - OLAP场景需要适配对接Spark SQL
- **应用改造成本**
  - 需要通过moray做数据搬迁
  - 访问接口协议无变化
    - 功能支持更完善
  - 改造成本低

5. 现状与未来



5. 现状与未来



5. 现状与未来

■对数据库产品持开放态度

- □TiDB、SequiaDB等根据业务需要使用
- □不限制业务方的技术选择自由
- □仅在关键联机业务上做审慎处理

■对数据产品未来的认知

- □从功能角度上有两个趋势：
  - 功能混合
  - 功能专业化
- □这两点并行不悖，推进技术发展

WE ARE JUST ON THE WAY

THANK YOU.


THANKS