DTCC

第十四届中国数据库技术大会

DATABASE TECHNOLOGY CONFERENCE CHINA

数智赋能 共筑未来

IT168.com ChinaUnix ITPUB

北京国际会议中心 | 2023/8/16-18

Klustron Global MVCC 原理和实现技术

泽拓科技 创始人 赵伟

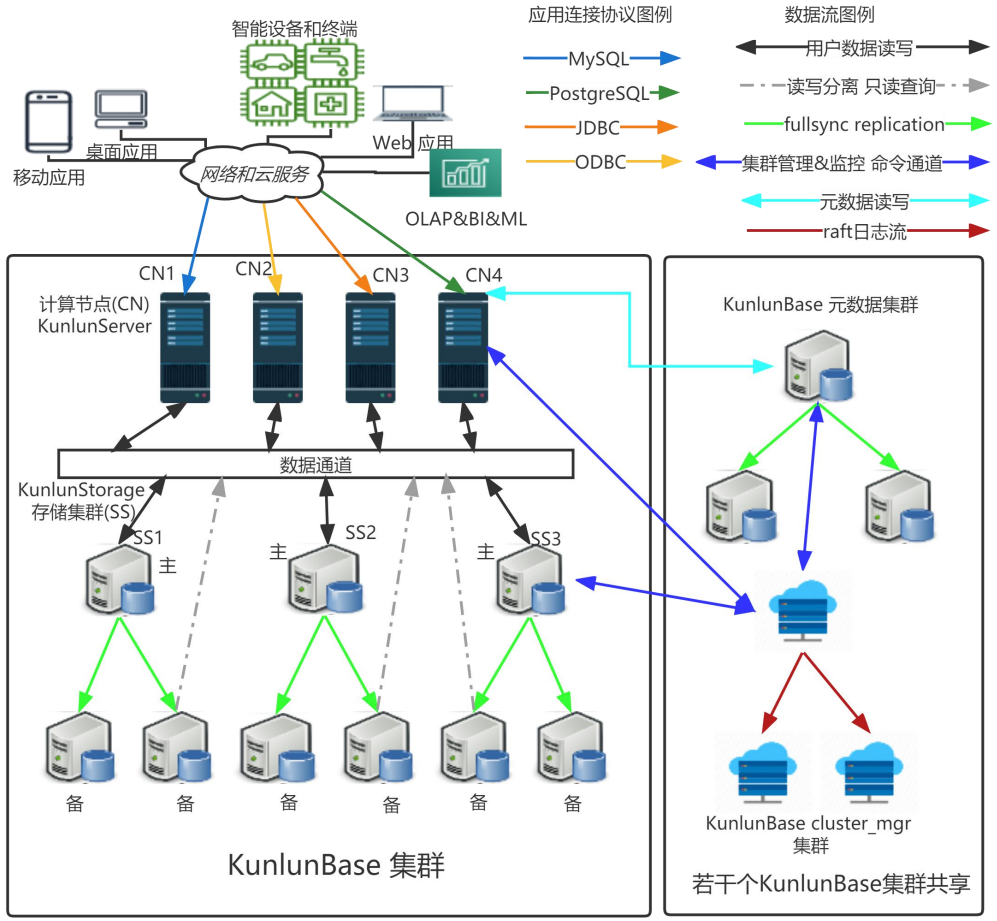
目录 Agenda

- Klustron 架构和核心功能简介
- Klustron Global MVCC解决什么问题
- Klustron Global MVCC的技术方案
- Klustron Global MVCC对InnoDB的改造
- Klustron Global MVCC的性能开销对比分析与实测结果

Klustron
架构和核心功能简介

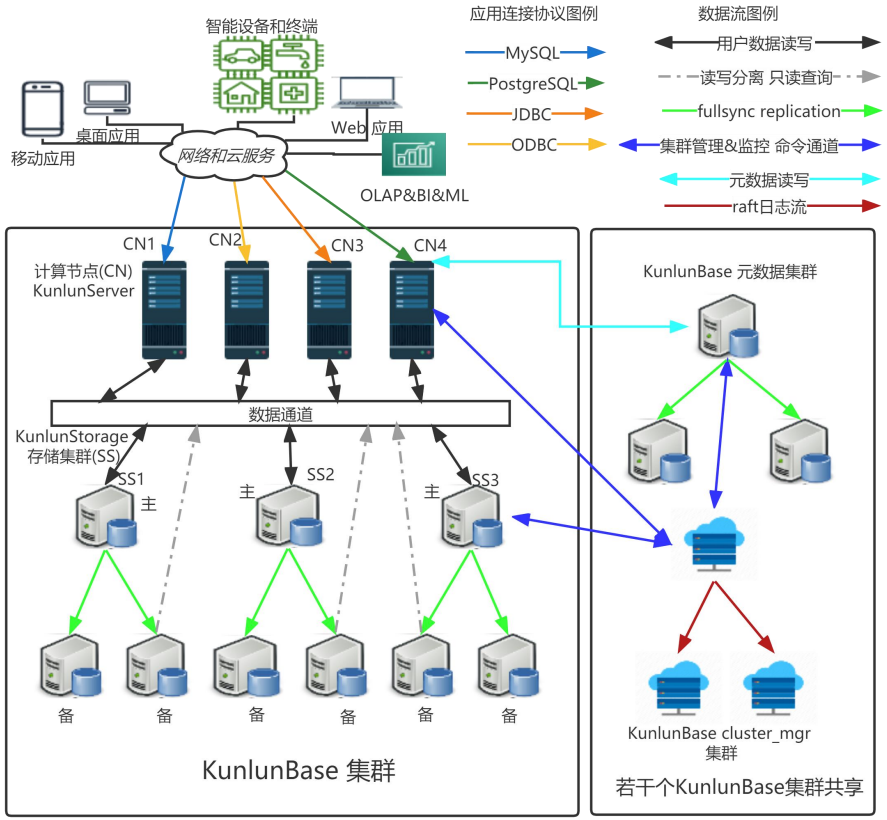
Klustron 架构与核心能力

- 弹性伸缩的计算和存储能力
  - 数据分区(partition): hash, range, list
    - 任意数量和类型的分区列
  - 数据分布(distribution)
    - auto, random, mirror, table grouping
  - 自动、柔性、不停服、无业务侵入、终端用户无感知
- 金融级高可靠性
  - 自动处理软硬件和网络故障和机房整体故障
    - 数据不丢不乱，服务持续在线
    - 确保RTO < 30秒 & RPO=0
  - 自动发现主节点故障并选主和主备切换



Klustron 架构与核心能力

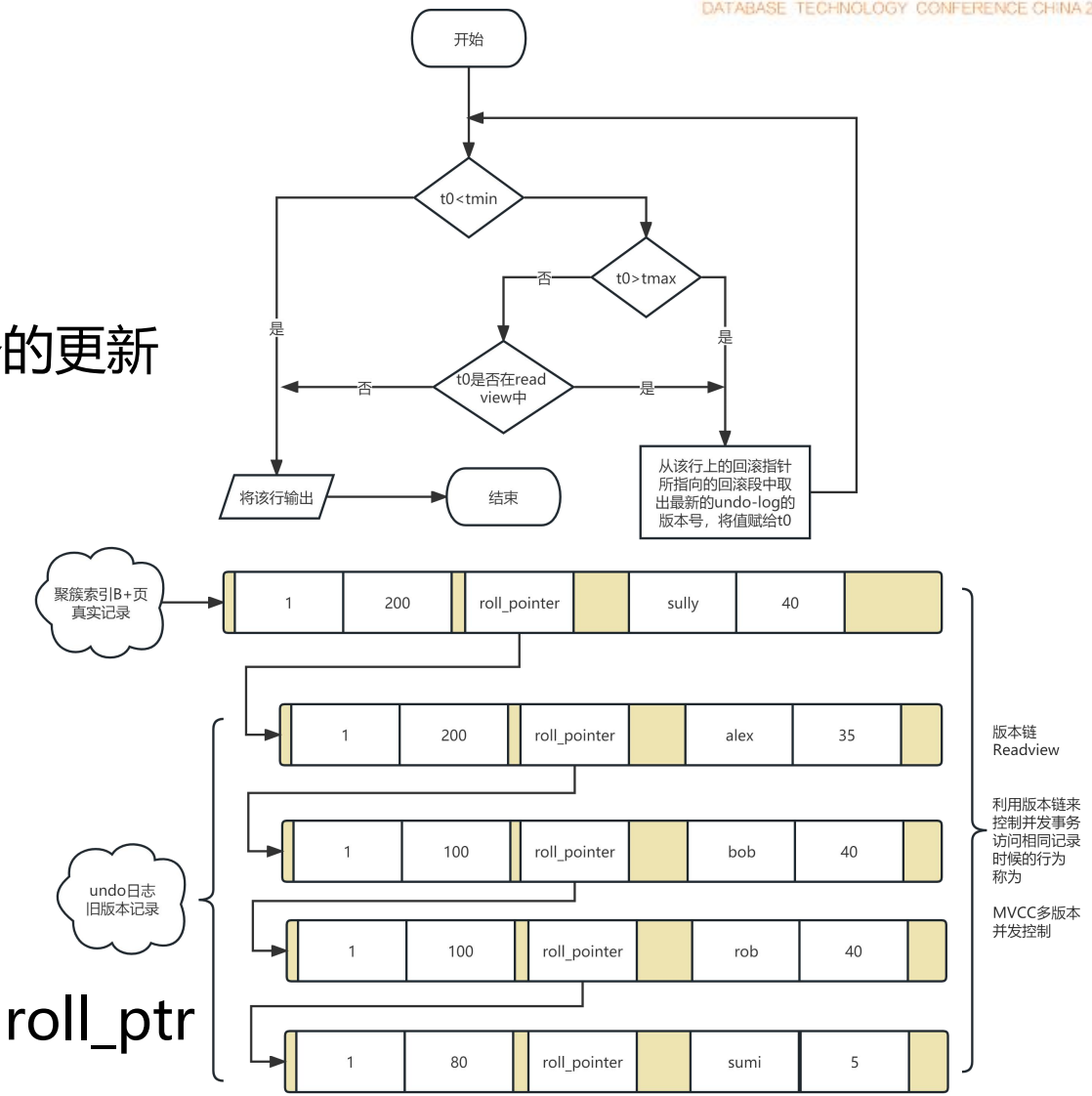
- HTAP: OLTP & OLAP 互不干扰
  - OLTP为主：对应用软件等价于使用MySQL或PostgreSQL
  - OLAP为辅：多层级并行查询实现高性能
    - 多语言存储过程的弹性计算：ML，隐私计算
- 生态兼容性
  - 支持PostgreSQL和MySQL 两种连接协议和SQL 语法
    - 支持MySQL 常用DDL语法
  - 支持JDBC，ODBC，常见编程语言的PostgreSQL和MySQL客户端connector
- 全方位多层级安全性
  - 加密存储和传输
  - 多层级访问控制机制



Klustron Global MVCC 解决什么问题？

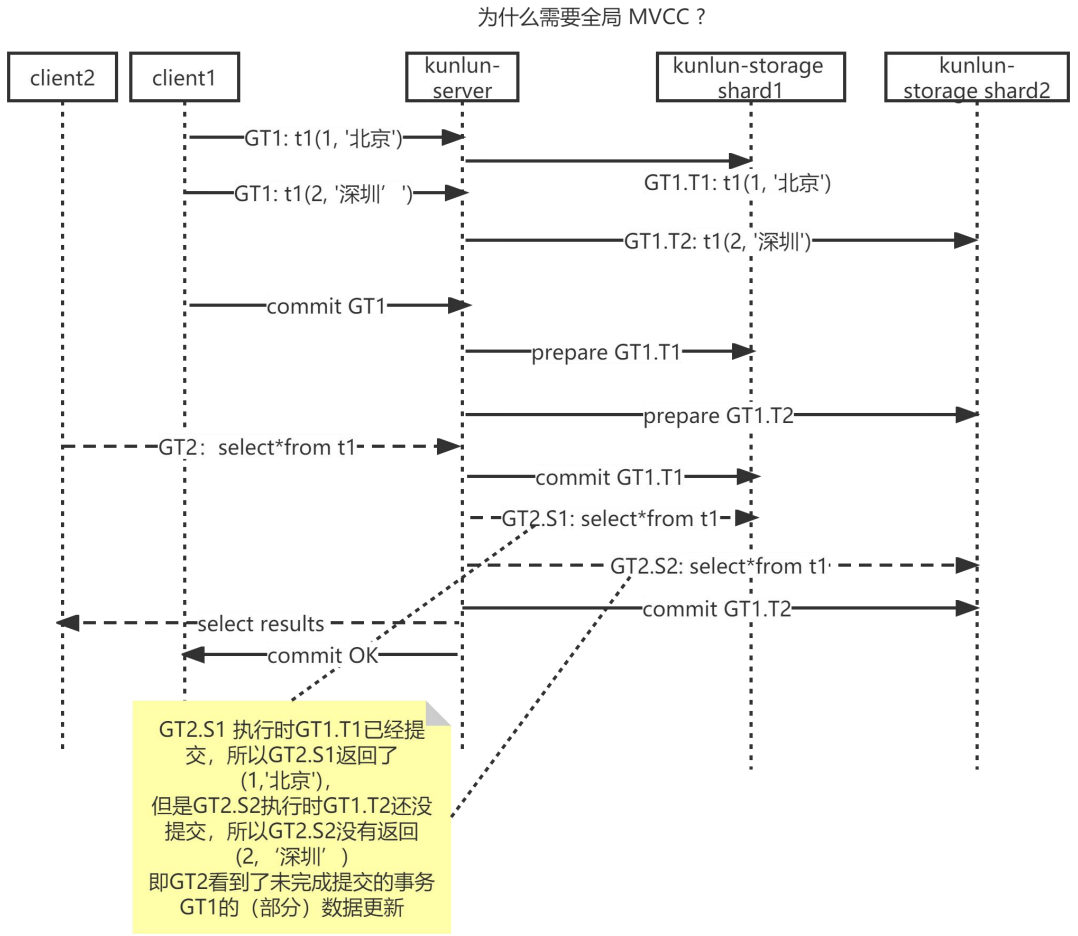
InnoDB MVCC

- MVCC基础
  - 优势：读不被写阻塞
  - 原理：只能读取到获取快照时刻已提交事务的更新
  - 快照时机：RC VS. RR
- InnoDB MVCC
  - row header: trx_id, roll_ptr
  - undo log: trx_id, roll_ptr
  - ReadView
    - [min, max], Active Trx_id List
    - changes_visible()
  - secondary index entries: no trx_id or roll_ptr



没有Global MVCC 的读一致性问题

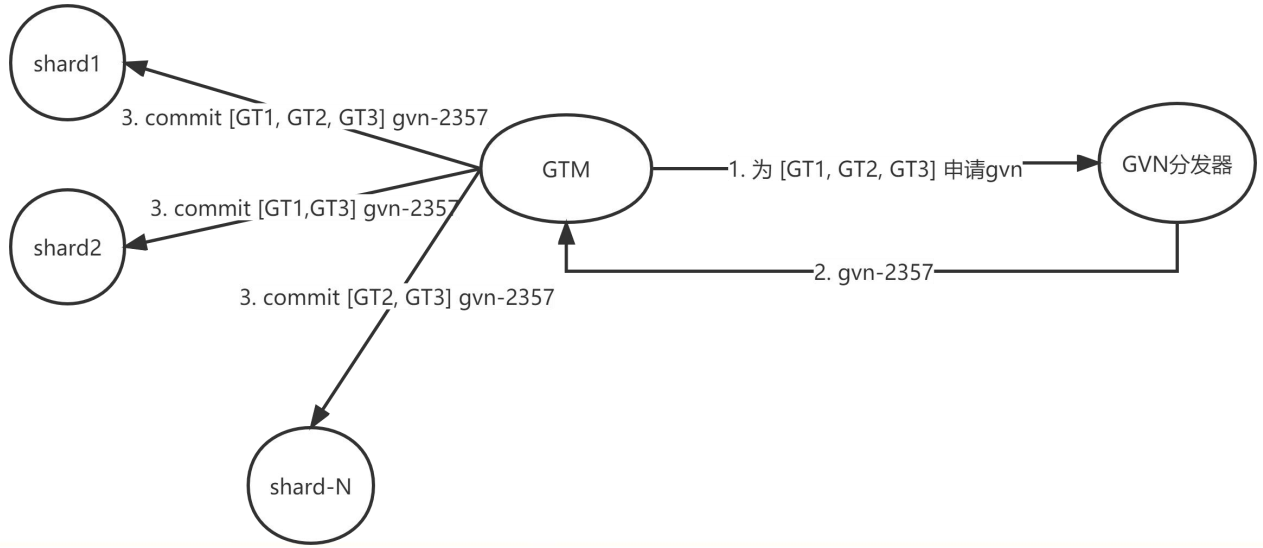
- 正在提交的分布式事务GT
  - 写入多个 shard (shard1 & shard2)
  - 两阶段提交
- 正在运行的SELECT
  - 读取到GT在shard1的更新
  - 未读取到GT在shard2的更新



Klustron Global MVCC 技术方案

为分布式事务获取和设置全局版本号

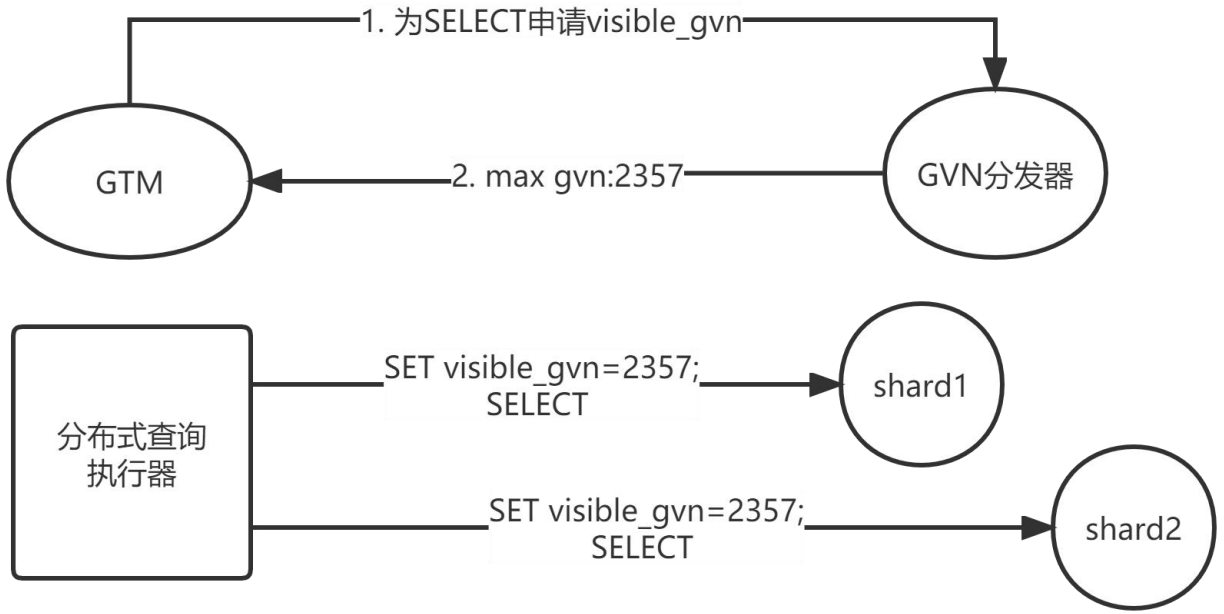
- 批量写commit log时获得全局版本号GVNO
  - 批量获取GVNO：高性能
  - 没有额外时间开销
- 发送给存储节点
  - XA COMMIT ... [ONE PHASE] [GVNO]
- GVNO 分发器
  - 运行在元数据集群的sequence
  - 高可用复制
  - 单调递增



为分布式事务SELECT获取和设置全局快照

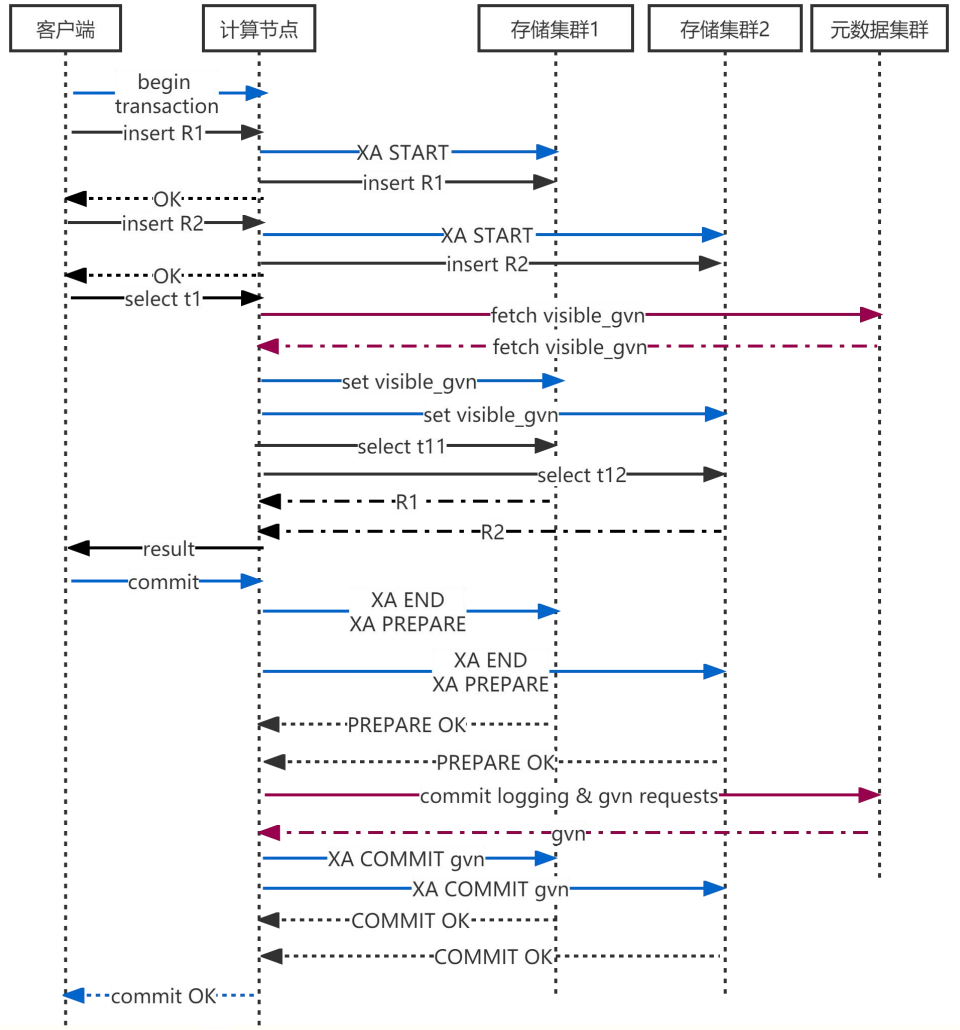
Klustron

- 全局快照：可见的全局版本号GVNO
  - 从GVNO分发器获得当前最大的全局事务版本号GVNO
  - 可以看到所有不大于此GVNO的所有全局事务的更新
- 获取版本号的时机
  - RC VS. RR
  - RC：事务的每个SELECT
  - RR: 事务第一个SELECT



Klustron Global MVCC in Action

- 对事务提交延迟的影响忽略不计
- 对SELECT查询执行有略微影响
- 全局事务的id全局唯一
  - 非全局有序
- 对用户透明，无需任何显式介入
  - 可以在集群层面enable/disable
  - 存储节点内仅对XA事务有效
- 融入了故障恢复和高可用体系



Global MVCC与其他功能的相互作用

- Global MVCC与崩溃恢复
  - 恢复后的事务具备GVNO：innodb recovery
- Global MVCC与高可用复制
  - 在gtid事件中记录gvno
  - 备机重放时gvno对XA COMMIT 有效
  - 主备切换后新的主节点可以正确支持Global MVCC
- Global MVCC与读写分离
  - 主备延迟：备节点执行一个事务的时机无法预期
  - Global MVCC对备机读不生效：备机读对数据一致性本来就没有严格要求
- Global MVCC与并行查询
  - 所有连接使用相同的gvno
  - XA COMMIT ONE PHASE

Global MVCC 状态观测

INFORMATION_SCHEMA.INNODB_GLOBAL_MVCC_TR

```
mysql> select * from INNODB_GLOBAL_MVCC_TRX;
+---------+----------------------+-------------------+------------+------------+------------+------------+--------------+
| trx_id  | xa_trx_id            | global_version_no | num_waiters | create_ts  | gvno_ts    | is_aborted | is_recovered |
+---------+----------------------+-------------------+------------+------------+------------+------------+--------------+
| 4368384 | 1-1691721891-5223245 |           4175269 |           0 | 1691721891 | 1691721891 |          0 |            0 |
| 4367360 | 1-1691721891-5222489 |           4175027 |           0 | 1691721891 | 1691721891 |          0 |            0 |
| 4364288 | 1-1691721890-5220122 |           4174335 |           0 | 1691721890 | 1691721890 |          0 |            0 |
| 4366336 | 1-1691721891-5221696 |           4174803 |           0 | 1691721891 | 1691721891 |          0 |            0 |
| 4368385 | 1-1691721891-5223265 |           4175262 |           0 | 1691721891 | 1691721891 |          0 |            0 |
| 4367361 | 1-1691721891-5222457 |           4175009 |           0 | 1691721891 | 1691721891 |          0 |            0 |
| 4364289 | 1-1691721890-5220109 |           4174335 |           0 | 1691721890 | 1691721890 |          0 |            0 |
| 4366337 | 1-1691721891-5221707 |           4174812 |           0 | 1691721891 | 1691721891 |          0 |            0 |
| 4368386 | 1-1691721891-5223196 |           4175262 |           0 | 1691721891 | 1691721891 |          0 |            0 |
| 4367362 | 1-1691721891-5222468 |           4175026 |           0 | 1691721891 | 1691721891 |          0 |            0 |
| 4364290 | 1-1691721890-5220121 |           4174335 |           0 | 1691721890 | 1691721890 |          0 |            0 |
| 4366338 | 1-1691721891-5221701 |           4174800 |           0 | 1691721891 | 1691721891 |          0 |            0 |
| 4368387 | 1-1691721891-5223229 |           4175263 |           0 | 1691721891 | 1691721891 |          0 |            0 |
| 4364291 | 1-1691721890-5220134 |           4174330 |           0 | 1691721890 | 1691721890 |          0 |            0 |
```

INFORMATION_SCHEMA.INNODB_GLOBAL_WAITERS

```
+----------------+--------------+----------------------+---------------+-----------------+-------------------+---------------
------+------------------+
| creator_trx_id | visible_gvno | waited_xa_trx_id     | waited_trx_id | time_waiting_ms | min_gmvcc_trx_id | global_mvcc_en
abled | is_purgesys_view |
+----------------+--------------+----------------------+---------------+-----------------+-------------------+---------------
------+------------------+
|              0 |      6869895 | 1-1691723822-9014359 |       6709606 |               0 |           6667542 |
    1 |                0 |
+----------------+--------------+----------------------+---------------+-----------------+-------------------+---------------
------+------------------+
```

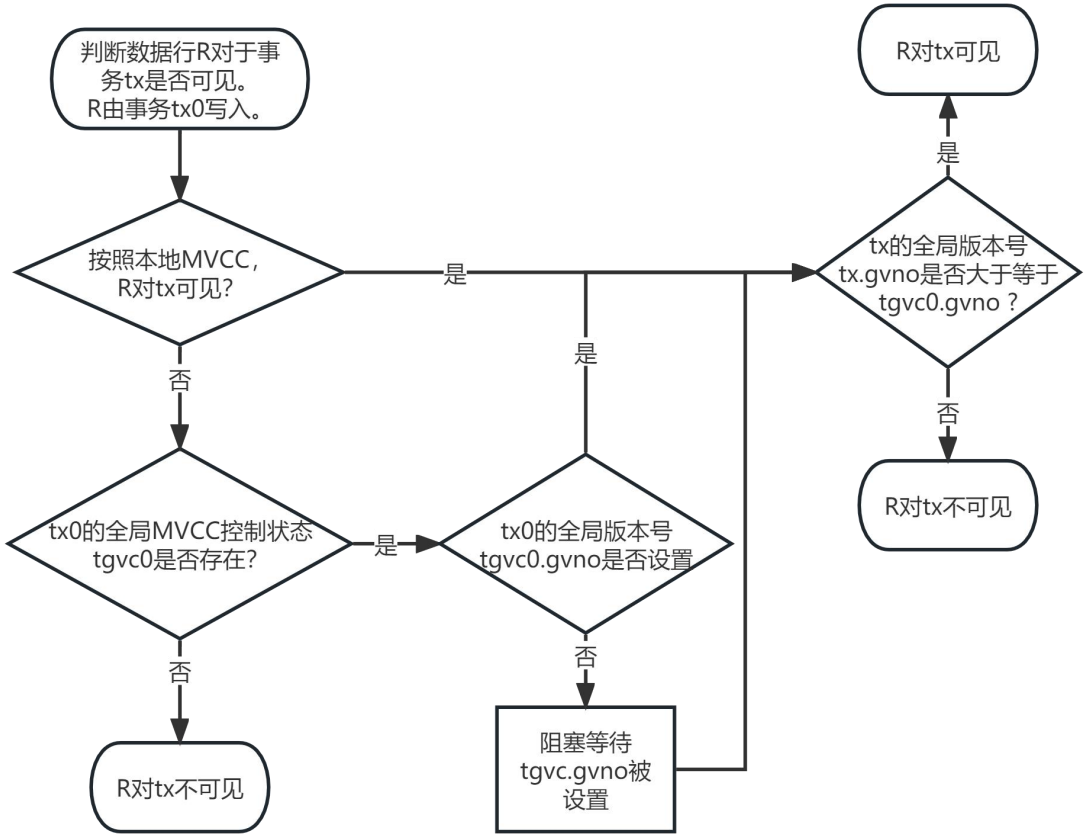
Klustron Global MVCC 对InnoDB的改造

扩充InnoDB MVCC 数据结构

- ReadView
  - local_xmin
  - visible_gvno
- TrxGlobalVersionControl : 事务的global mvcc 控制对象
  - trx_id，trx_gvno
  - 等待和唤醒机制
- GlobalMVCC
  - tgvc_cache: trx_id ->TrxGlobalVersionControl
  - local_xmin
- global_xmin

InnoDB ReadView::changes_visible

- 全局可见性判断算法：仅针对XA事务的更新
  - 先做局部可见性判断
  - 局部可见未必全局可见
    - 小于local_xmin 的一定可见
      - **只有读取最近若干秒内提交的事务的更新才需要做global mvcc可见性判断**
  - 局部不可见未必全局不可见
    - 获取快照时尚未启动的一定不可见
  - 全局版本号存放在哪里？
    - 何时设置？
    - 没有怎么办？等！
  - 全局不可见怎么办？
    - 用 undo log生成更老的行版本



全局事务版本号管理

- TrxGlobalVersionControl
  - 设置GVNO：XA COMMIT [ONE PHASE] GVNO
  - 阻塞等待GVNO
  - 唤醒等待者
- 不改变行结构，不持久存储
  - 数据格式与社区版MySQL保持一致：生态兼容
  - gvno存入数据行 ？
    - 必须在事务启动时获得全局事务ID
    - 整体算法和架构与PG-XL相同，代价也相同

全局事务版本号管理

- crash safety: 重启后仍然遵循全局MVCC 可见性规则
  - undo log
  - recovery: 重建tgvc hash table
- 推进全局低水位版本号 global_xmin
  - cluster_mgr 从所有计算节点定期
    - 收集其当前本地活跃事务的最小GVNO 作为global_xmin
    - 把global_xmin 送给所有存储节点
  - purge undo log

Klustron Global MVCC代价分析与性能

Global MVCC 代价分析

- 获取GVNO：在批量写commit log 期间完成
  - select nextval（‘global_mvcc_seq’）
  - 未增加新的已有的时间开销
- 分配GVNO：随XA COMMIT 语句发放一个整数，存入tgvc_cache中的tgvc
  - 忽略不计
- 获取全局快照：网络收发开销
  - 每个SELECT语句（RC）或者每个事务（RR）获取一次
  - 从元数据集群sequence取得当前值 select currval（‘global_mvcc_seq’）
- 分配全局快照：随SELECT语句下发一个整数
  - 忽略不计

Global MVCC整体性能损耗约5%~10%

Global MVCC 代价分析

- 存储节点
  - tgvc管理：忽略不计
  - Global MVCC 可见性判断逻辑：整数比较
    - 少量READ等待设置全局版本号：等待时间通常 < 20ms
  - 覆盖索引查找：页头部max_trx_id:上一次更新本页的事务
    - 以前：readview 可见本页所有行(rv.m_up_limit_id > max_trx_id)，则直接返回索引行
    - 现在：上述成立并且如果max_trx_id > local_xmin, 必须回表以便查找
      - 回表查找的比例略有增加
- purge：保留undo log直到global mvcc不再需要
  - 由global_xmin 的上升来推动

Global MVCC整体性能损耗约5%~10%

对比：PG-XL Global MVCC代价分析

- GTM节点维护全局活跃事务ID集合(ATL)
  - 每个节点上报每个事务的启停，加入和移出ATL
    - 全局事务启动时分配全局事务ID
  - 获取快照：复制ATL
    - 1000个并发连接：每个全局快照的最大size：8 * 1000=8KB
      - 为每个SELECT传输8KB N次！
      - 假设每个SELECT查询100ms，每个SELECT 读取shard个数平均分布，RC隔离级别
        - 大约消耗网络带宽：shard总数/2 * 80MB/s ！
        - 从元数据集群获取快照 10000次：80MB/s ！
    - GTM的ATL 是热点数据结构，性能瓶颈 ！
  - GTM 需要高可用机制
    - 否则GTM宕机则快照信息丢失，集群需要完全重启

THANKS