解密数据库分布式事务处理核心技术

- 泽拓科技 Zettadb.com 赵伟

纲要

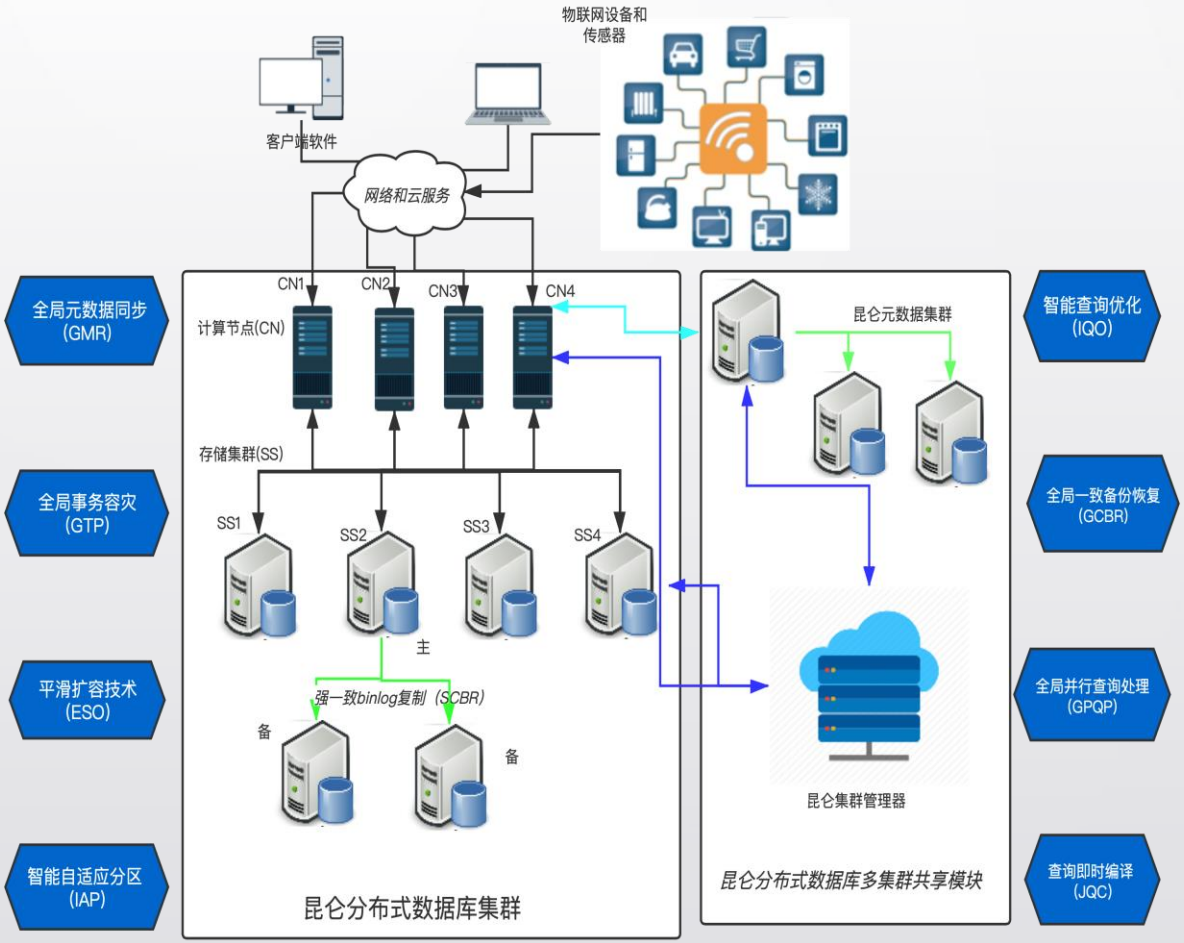
- 昆仑分布式数据库架构
- 昆仑分布式数据库的技术特点和优势
- 昆仑分布式数据库容灾能力概述
- 昆仑分布式数据库全局事务处理机制
- 昆仑分布式数据库全局事务恢复机制
- 昆仑分布式数据库DDL事务处理机制
- 昆仑分布式数据库DDL事务恢复机制

线上交流群

公司公众号

昆仑分布式数据库架构

- 昆仑分布式数据库有机融合了PostgreSQL和MySQL的优势，成为具备独特技术优势和灵魂的全新的分布式数据库管理系统。
- 昆仑分布式数据库“站在PostgreSQL和MySQL两个巨人的肩膀上”，与二者是协作关系，有利于二者未来更好地发展



昆仑分布式数据库架构

- 计算节点（Computing Nodes，CN）
  - 接受和验证用户连接请求
  - 接收和处理来自用户连接中的DDL/DML SQL语句
  - GTP：分布式事务处理
  - IQO：分布式查询处理
  - GMR：DDL事务并发执行和并发复制
  - 数量按需弹性增减，互相独立，本地元数据相同
    - 不存储用户数据，只存储元数据
    - 占用微量存储空间（若干MB）
    - 本地存储会话/连接临时数据（临时表和物化视图）
  - 用户数据存储在存储集群中
    - GPQP：异步读写存储集群访问用户数据
  - 本地状态可用集群的元数据重建
    - 无需为计算节点本地数据做容灾
    - 不会给DBA和运维带来额外管理工作

昆仑分布式数据库架构

- 存储集群（Storage Shards，SS）
  - 存储应用（用户）数据，只使用mysql单表，每个单表或者表分区对应一个mysql单表
  - 执行计算节点发起的XA事务分支
  - 计算节点自动探测&切换最新主节点（auto failover）
  - 目前只支持innodb引擎
    - 不使用非事务引擎(myisam, etc)
    - 未来可能支持myrocks
  - 可选地使用 Kunlun-storage
    - 含有关键的bug修复和支持功能，以及性能巨大提升
    - 会随上游版本升级
  - 可以使用AWS Aurora，Aliyun PolarDB等自带高可用机制的mysql分支

昆仑分布式数据库架构

- 存储集群（Storage Shards，SS）高可用（HA）机制
  - MGR单主模式做集群高可用
    - 自动主选举
    - 健壮的一致性保障
  - Row Based Replication（*）
    - 强一致
    - 集群内自动探测主节点故障 & 选举新主节点
  - 非binlog复制的高可用（no_rep)
    - 类似 PolarDB, Aurora的HA
    - 基于共享存储的HA

昆仑分布式数据库架构

- 元数据集群
  - 存储着若干个昆仑分布式数据库集群的元数据
    - commit log
    - ddl log
    - 所有节点连接信息以及存储集群拓扑结构信息
  - 若干个昆仑分布式数据库集群共用
  - 与存储集群相同的mysql HA 集群
- Cluster_mgr
  - 维持每一个存储集群及其节点的replication状态
    - 节点随时可能宕机/退出/加入
  - 启动整个存储集群
    - 选择正确的主节点
  - 服务注册到一个元数据集群的所有昆仑数据库集群
  - 分布式事务特定故障处理
  - 统计信息和集群状态同步

昆仑分布式数据库的容灾能力

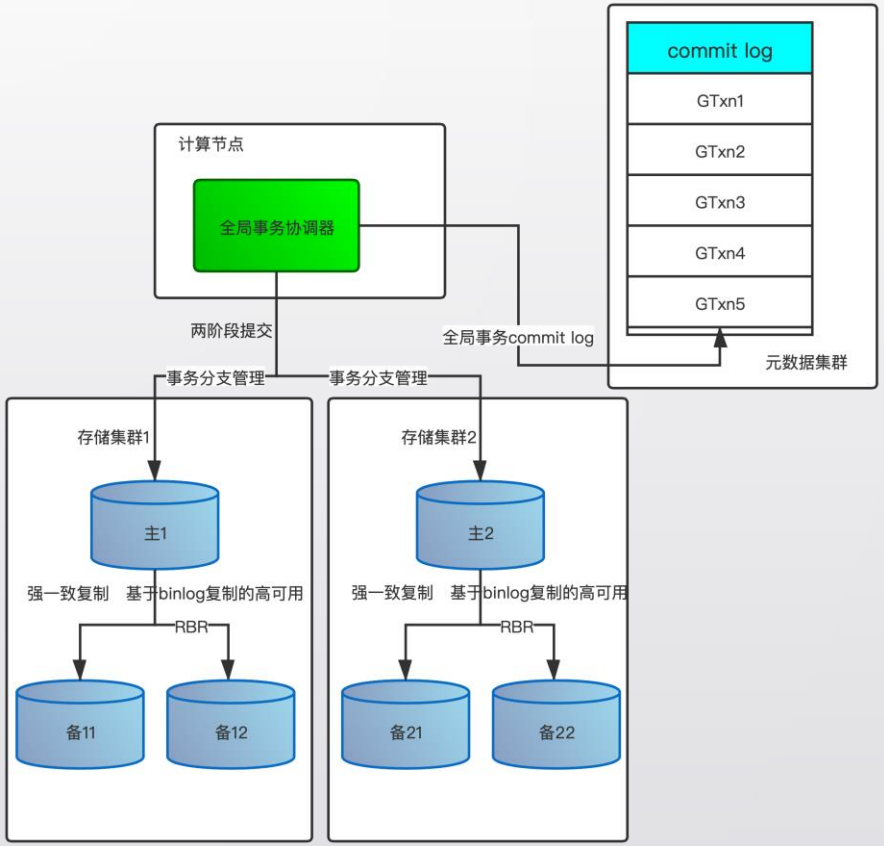
- 用户友好（单机DB用法；无缝兼容MySQL* 和PostgreSQL应用；DBA：管理MySQL集群；）
- 高可靠性，强一致性和完备的容灾能力（自动容灾，自动恢复，使用无忧）
- 高可扩展性(*)：按需的弹性的无感知的水平扩展能力
- 高性能：全系统并行查询处理能力（计算节点层，计算节点与存储节点之间，存储节点层*）
  - 充分挖掘多核CPU的潜力
- 完备的查询处理功能
  - prepared stmt，跨shard多表连接和子查询，OLAP分析，CTE，视图，物化视图，存储过程
- 全方位数据安全保障（数据&binlog加密，SSL连接，prepared防SQL注入，多层级多粒度访问控制）
- 揉合MySQL的优势 InnoDB + Binlog Replication 和 PostgreSQL优秀的查询处理能力

昆仑分布式数据库的技术特点和优势

- 存储集群主备强一致 （High Availability, Strong Consistency)
  - 组复制（Group Replication）/半同步复制（Semisync Replication）
- GTP：全局事务处理 (Crash safety & Fault tolerance)
  - 分布式事务两阶段提交
  - 节点/网络故障时可以保障分布式事务ACID
- GTP：计算节点自动切换存储集群主节点（auto failover）
  - 元数据集群，存储集群
- GTP：cluster_mgr:自动维护各shard的存储集群复制状态
- GMR：DDL事务与集群全局元数据一致性
  - DDL事务涉及 计算节点，元数据集群，存储集群
- GMR：计算节点DDL语句复制及其一致性保障
  - 复制过程随时可能中断
  - “断点续传” 无遗漏和重复执行

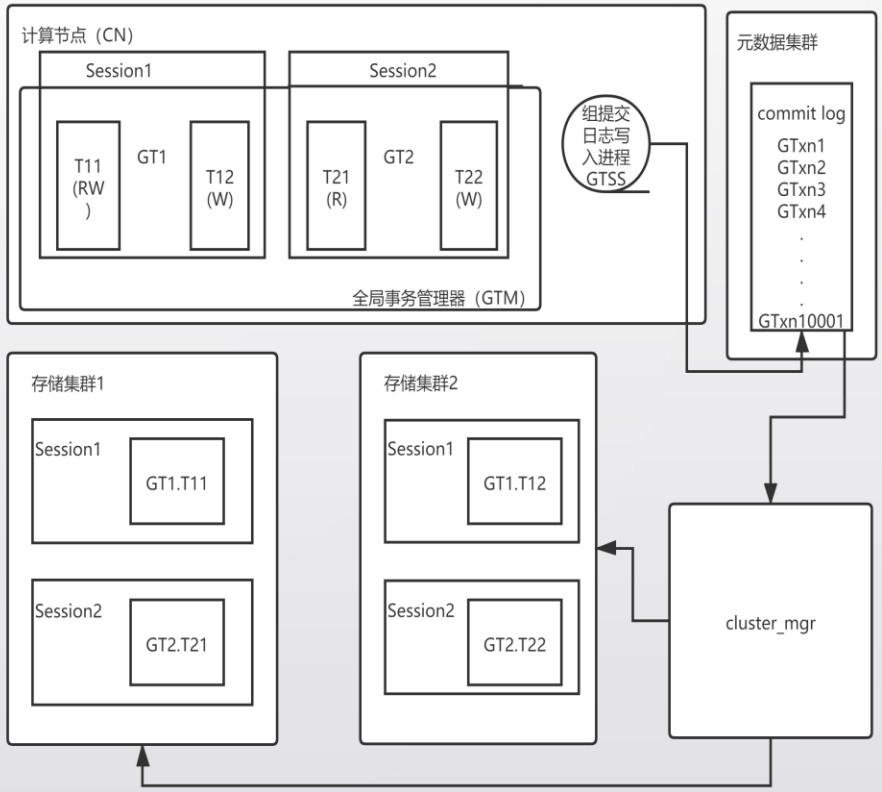
昆仑分布式数据库全局事务处理 --- 模块和组件

- 全局事务管理器 (Global Transaction Coordinator)
  - 位于计算节点
  - 跟踪每个事务读写的存储集群
  - 控制分布式事务提交过程
    - 2PC：写入多个存储集群
    - 1PC：写入1个存储集群和所有只读集群
- 存储集群 (Resource Manager, RM)
  - 执行分布式事务分支
  - 保障本地事务分支的容灾和高可用
- 元数据集群 commit log
  - 全局事务id写入commit log 《=》必将完成提交
  - 2PC流程中断不会破坏全局事务ACID
  - group commit，高性能，低开销
- cluster_mgr
  - 容灾后处理



昆仑分布式数据库全局事务处理 --- 全局事务运行时状态

- 计算节点 GTM
  - 每个会话（Session）
  - 本事务读写过的后端连接集合
  - 事务分支的读写状态
- 存储集群 (Resource Manager, RM)
  - 每个来自计算节点的连接
  - 本地事务分支的状态
- cluster_mgr
  - 每个存储主节点恢复后的XA事务分支
    - XA Recover



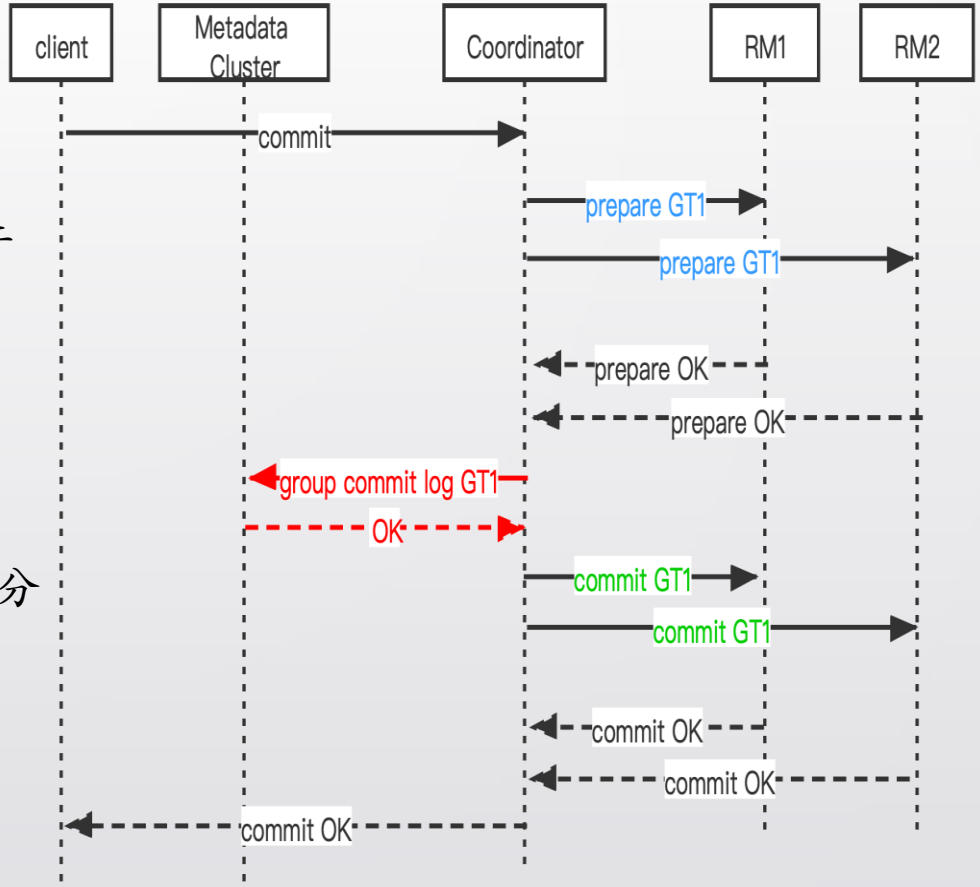
昆仑分布式数据库全局事务处理

- commit log表
  - autocommit事务写入innodb表
  - 计算节点group commit 成组写入
- 全局事务ID产生方式
  - 每个计算节点独立产生
  - 全局唯一，非全局有序
  - 格式：计算节点id-提交时间戳-本地事务id

```
mysql> select*from commit_log_clust5 limit 10 offset 200;
+--------------+---------------------+--------------+---------------------+
| comp_node_id | txn_id              | next_txn_cmd | prepare_ts          |
+--------------+---------------------+--------------+---------------------+
|            1 | 68989686442992007 89 | commit       | 2020-11-25 16:01:50 |
|            1 | 68989686442992007 90 | commit       | 2020-11-25 16:01:50 |
|            1 | 68989686528891354 11 | commit       | 2020-11-25 16:01:52 |
+--------------+---------------------+--------------+---------------------+
3 rows in set (0.02 sec)
```

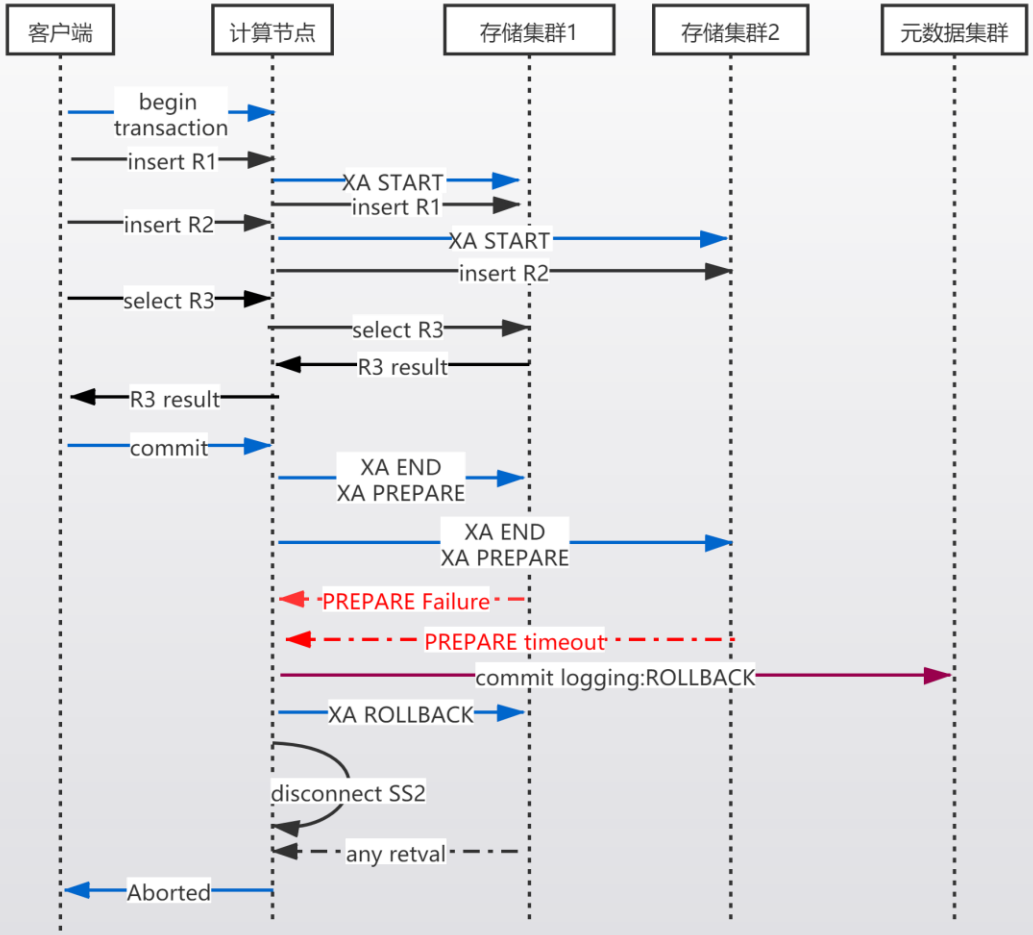
昆仑分布式数据库全局事务处理

- 可能的错误
  - 计算节点宕机
  - 存储集群主节点宕机
  - 计算节点与存储集群网络连接断开
  - 错误或者重负载引起的超时
- 异常处理:分布式集群范围，以写commit log完成为界
  - 之前：coordinator或者cluster_mgr abort所有事务分支
  - 之后：cluster_mgr commit所有分支，coordinator返回警告
- 异常处理：单存储集群内
  - mysql innodb&binlog 恢复机制



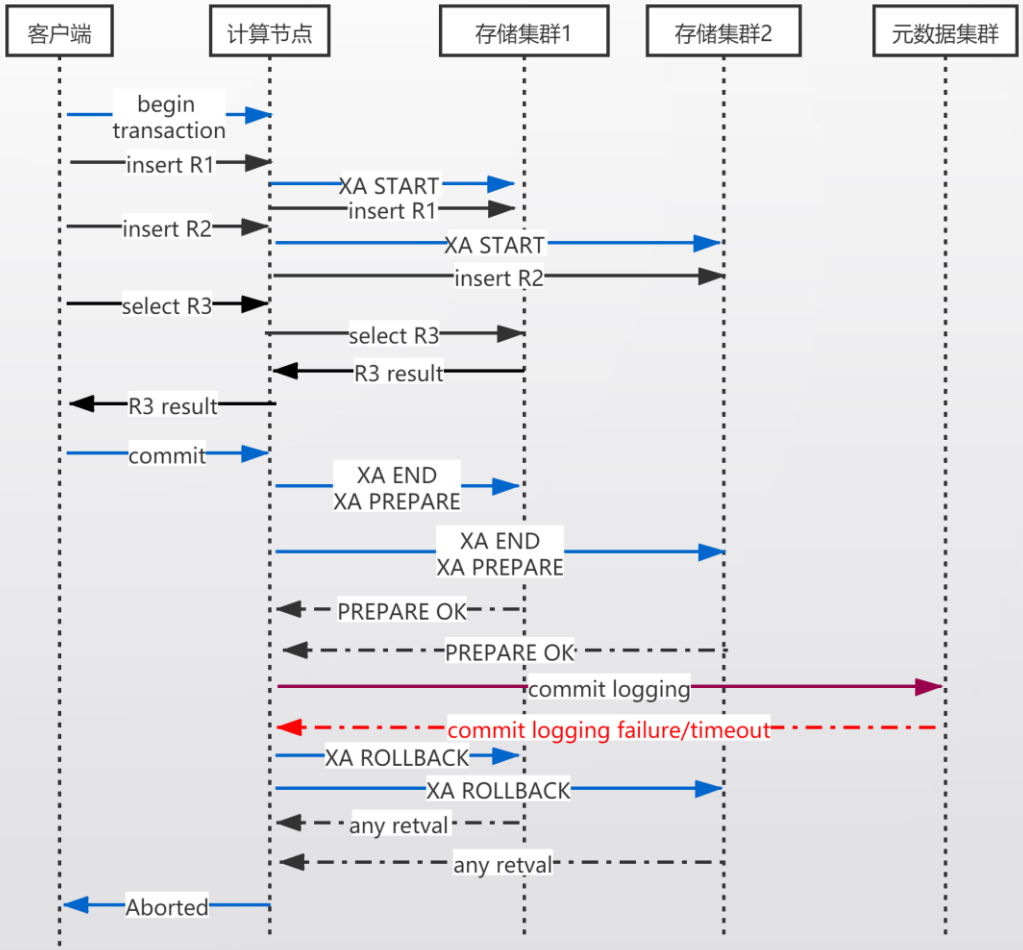
昆仑分布式数据库全局事务处理--- 错误处理

- Prepare阶段有RM失败或者超时
  - 异步发送ROLLBACK 到commit log
  - coordinator abort所有事务分支



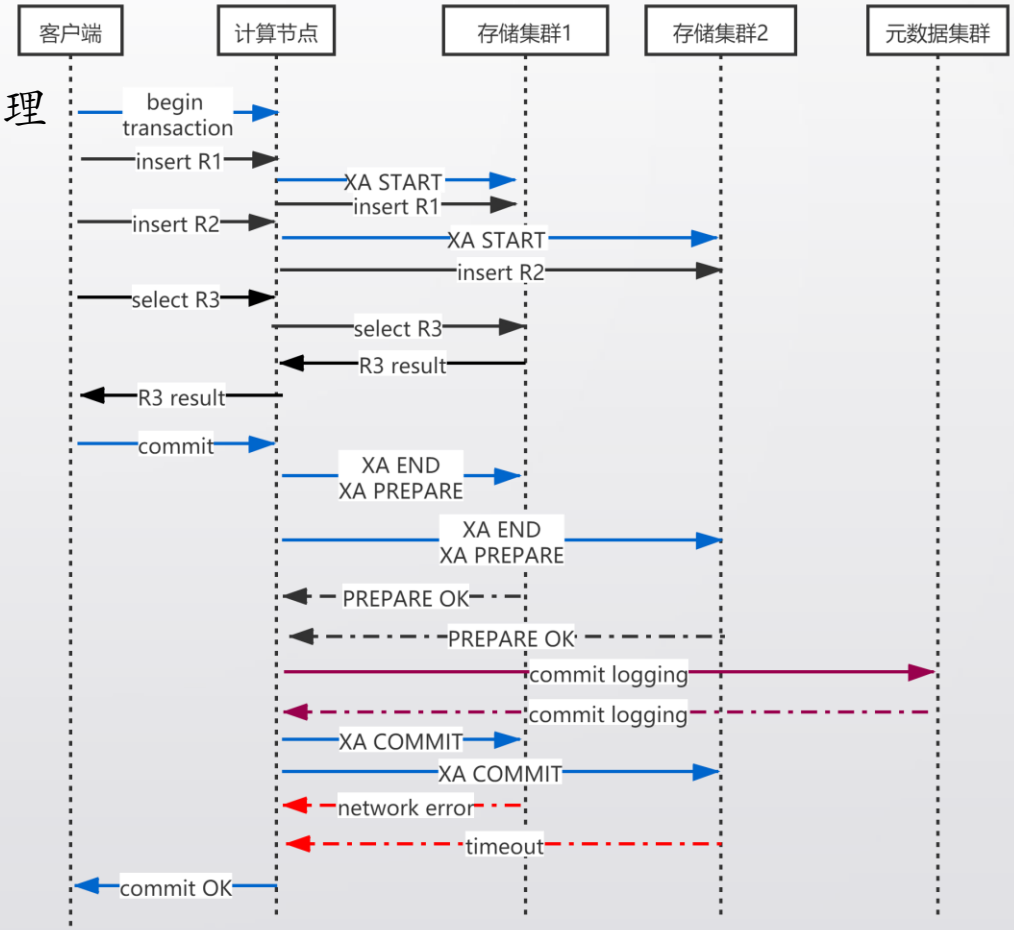
昆仑分布式数据库全局事务处理--- 错误处理

- Commit log写入失败或者超时
  - coordinator abort所有事务分支
  - 返回错误给应用端，事务回滚



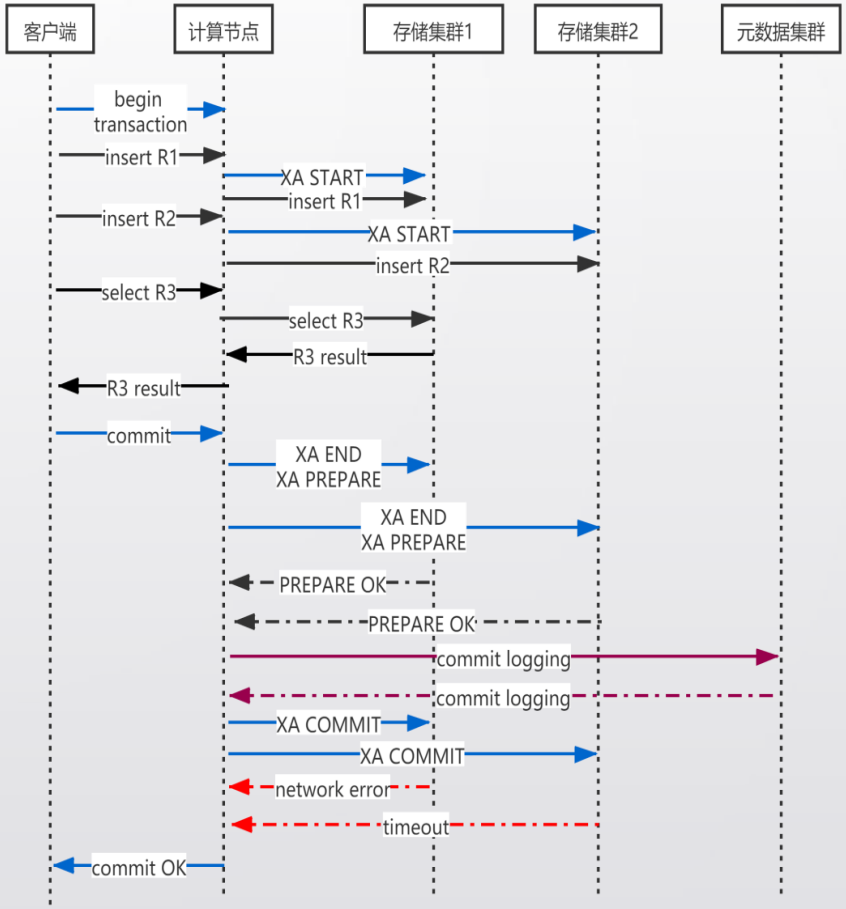
昆仑分布式数据库全局事务处理--- 错误处理

- 第二阶段有节点超时或者网络断连
  - 断开后端连接，让cluster_mgr处理
  - 返回提交成功给客户端



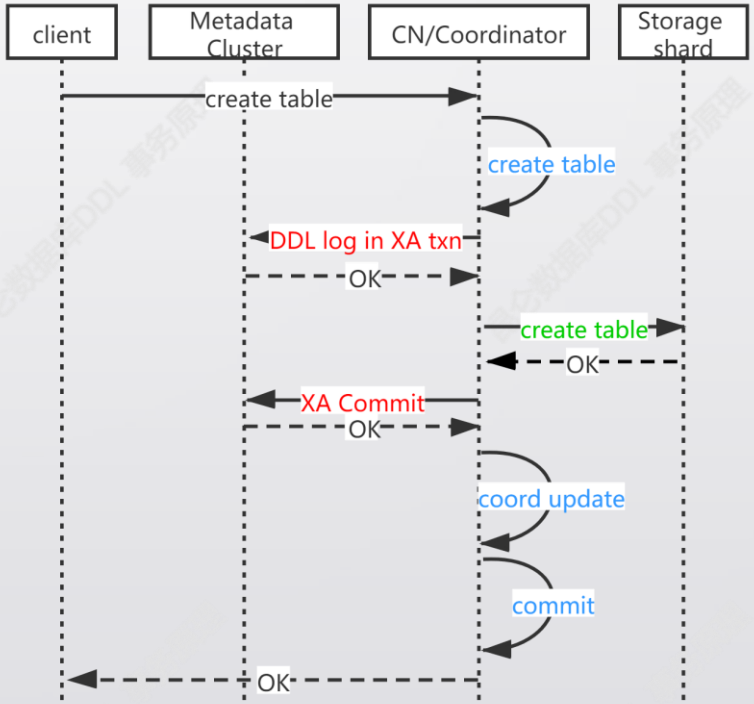
昆仑分布式数据库全局事务处理--- 性能

- 提交每个2PC事务都需要写入元数据集群
  - 合并写入，通常100:1~300:1
  - 变量控制（等待汇集数量和时间）
  - **没有性能瓶颈**
- 事务提交延时：比1阶段增加30毫秒以内
  - 增加了commit阶段的时耗：微量定量IO
  - 写入commit log的时耗：微量定量IO



昆仑分布式数据库DDL事务与集群全局元数据一致性

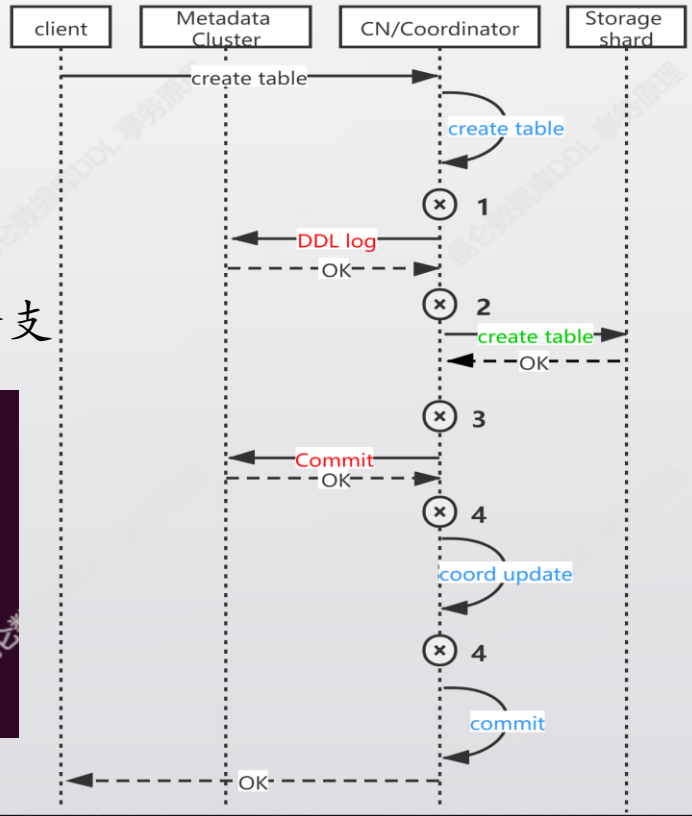
- DDL事务协调器：位于计算节点
- DDL事务的参与者
  - 计算节点：更新本地元数据和执行坐标
    - pg_ddl_log_progress.max_op_id_done_local
  - 存储集群：数据对象的创建/更新
    - 多数DDL不涉及存储集群
  - 元数据集群：存储DDL log
- 事务一致性要求：以下三者保持一致
  - 计算节点元数据
  - 元数据集群的DDL log
  - 存储集群的数据对象
- 复制一致性模型
  - 最终一致
  - 不一致期间容错
    - DDL：禁止同DB
    - DML：发现和返回错误



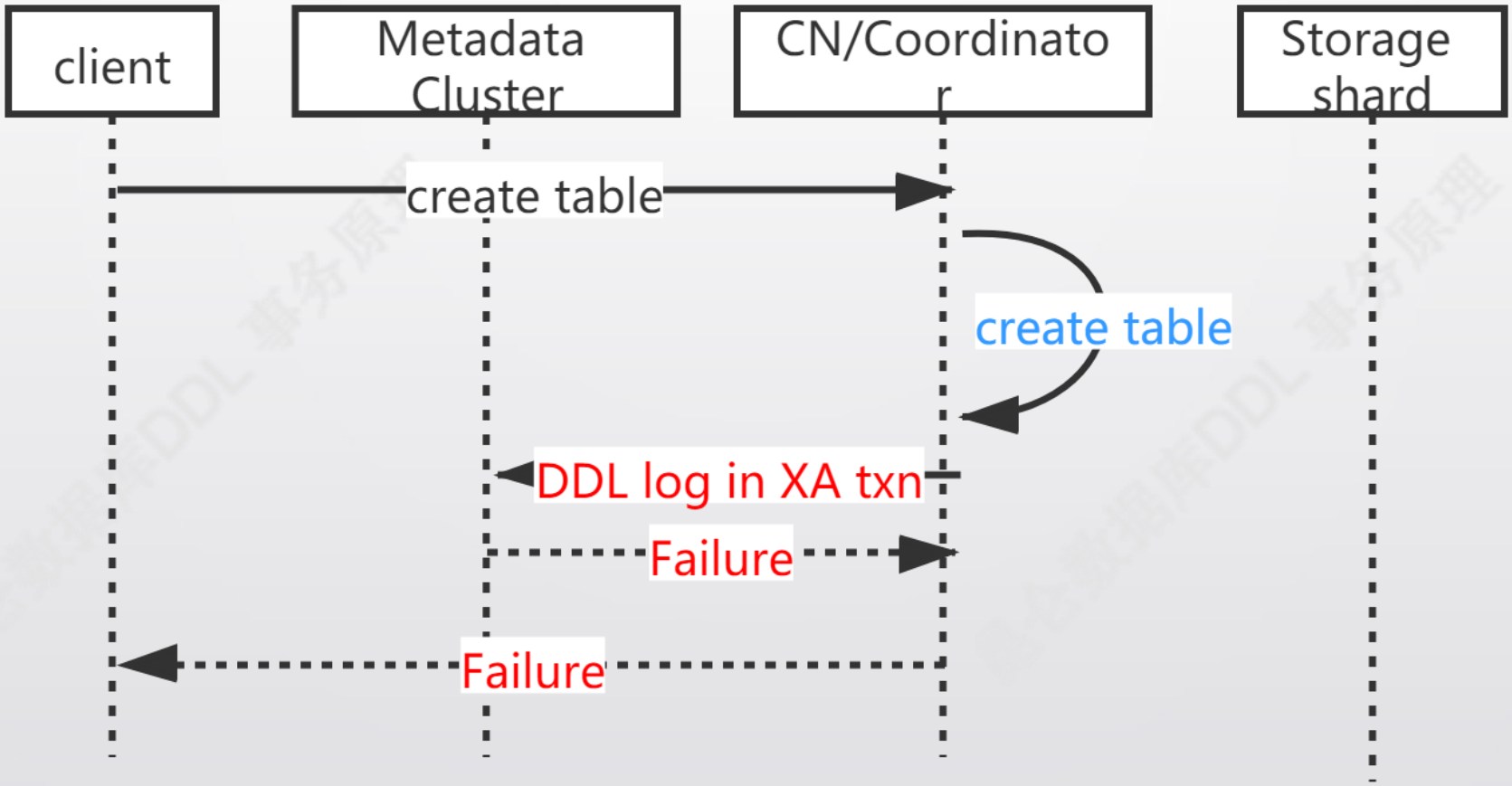
昆仑DDL事务与集群全局元数据一致性

- 计算节点DDL容灾恢复
  - 1: 计算节点本地事务回滚
  - 4: 执行坐标对比和DDL 日志重放
    - pg_ddl_log_progress.max_op_id_done_local
- 存储集群DDL容灾恢复
  - MySQL-8.0 DDL是独立的原子语句
    - 不可回滚，不可位于其他事务中
  - 2: 错误日志辅助手工回滚
- cluster_mgr
  - 3:容灾后处理：回滚元数据集群残留的事务分支

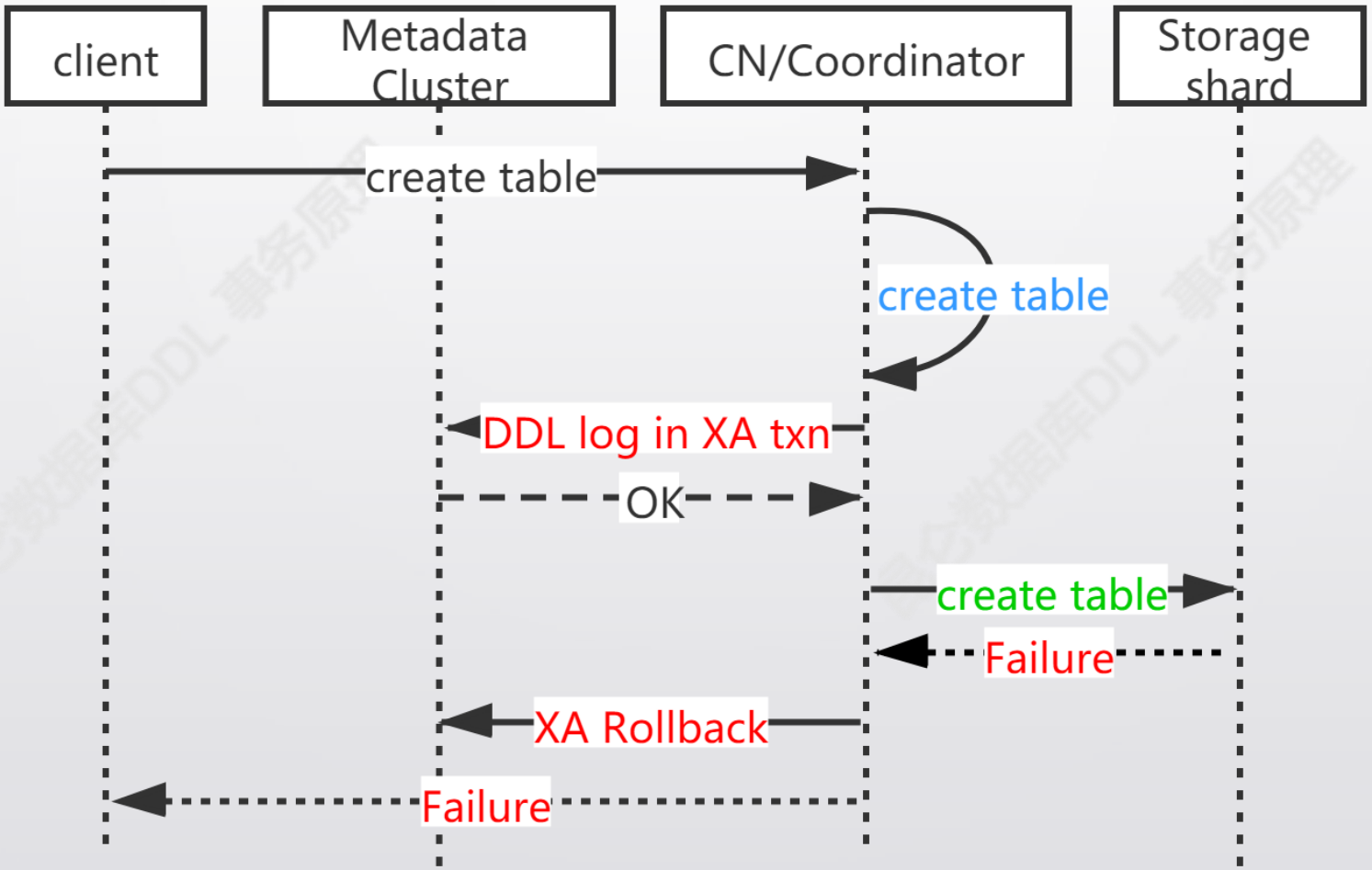
```
postgres=# select*from pg_ddl_log_progress;
  dbid    | ddl_op_id | max_op_id_done_local
----------+-----------+----------------------
        0 |         0 |                    0
    13160 |      3064 |                 8926
   131074 |      3044 |                 3042
  1156377 |      8572 |                 8913
   262147 |      3062 |                 3062
   262148 |      3063 |                 3063
   270336 |      3064 |                 3064
(7 rows)
```



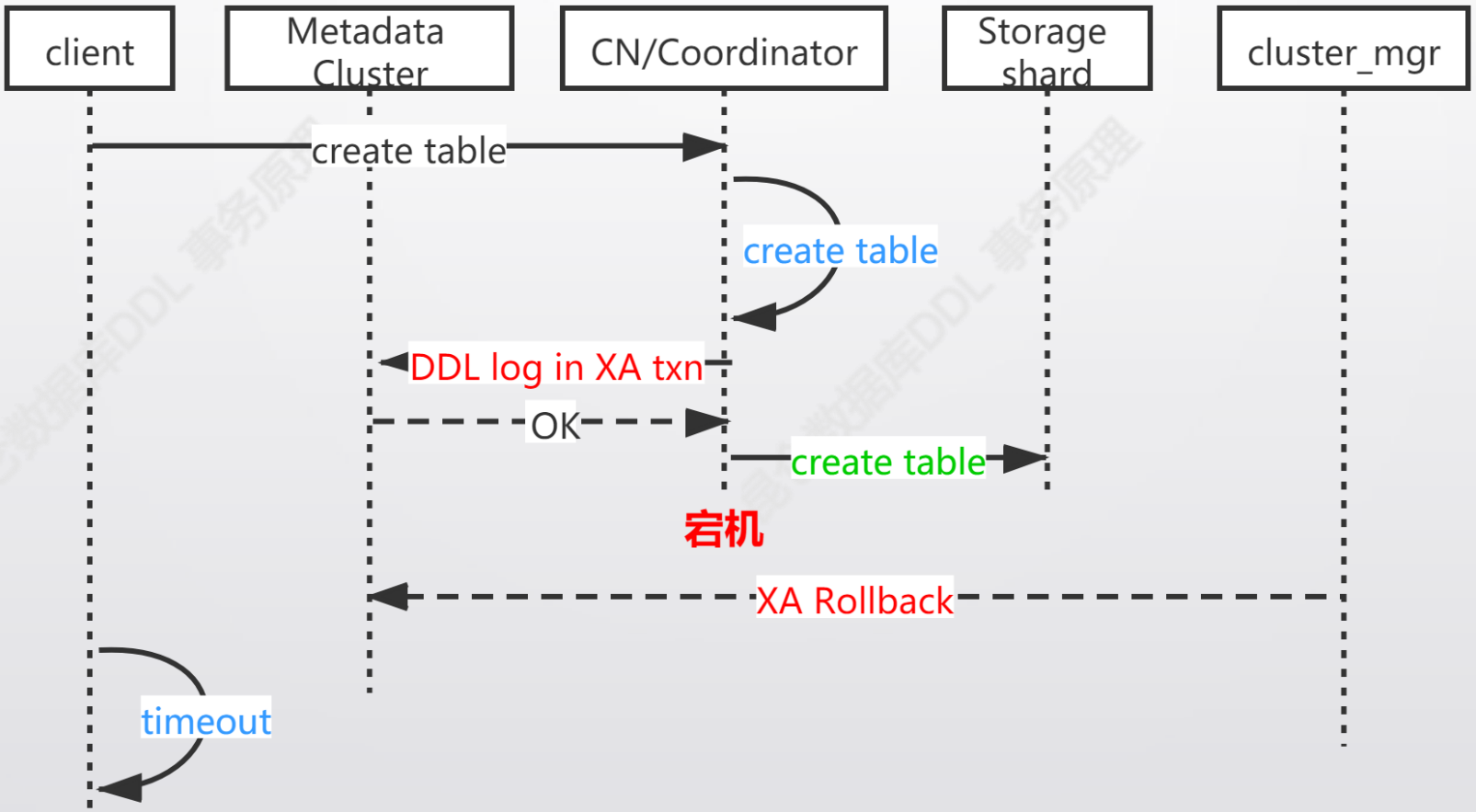
昆仑DDL事务与集群全局元数据一致性 -- 错误处理



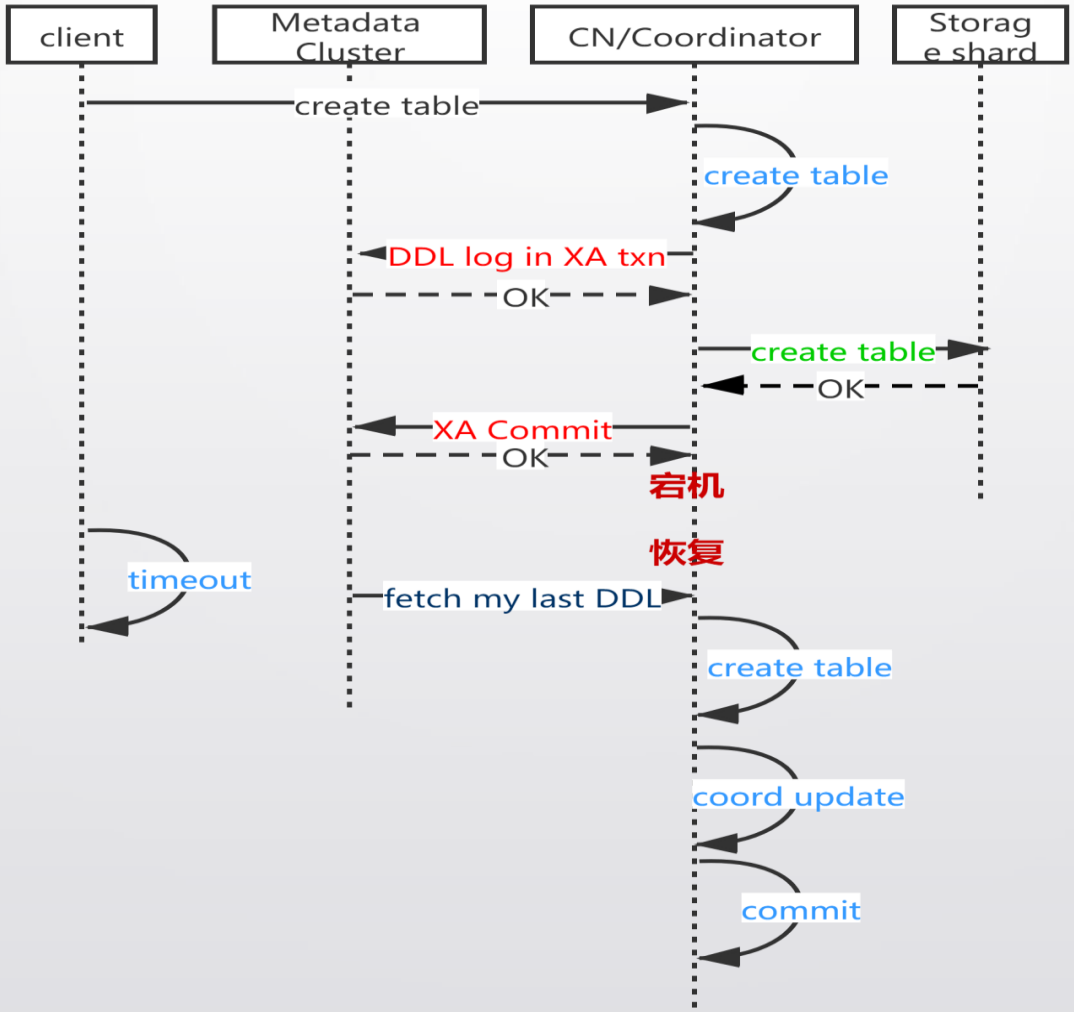
昆仑DDL事务与集群全局元数据一致性 -- 错误处理



昆仑DDL事务与集群全局元数据一致性 -- 错误处理



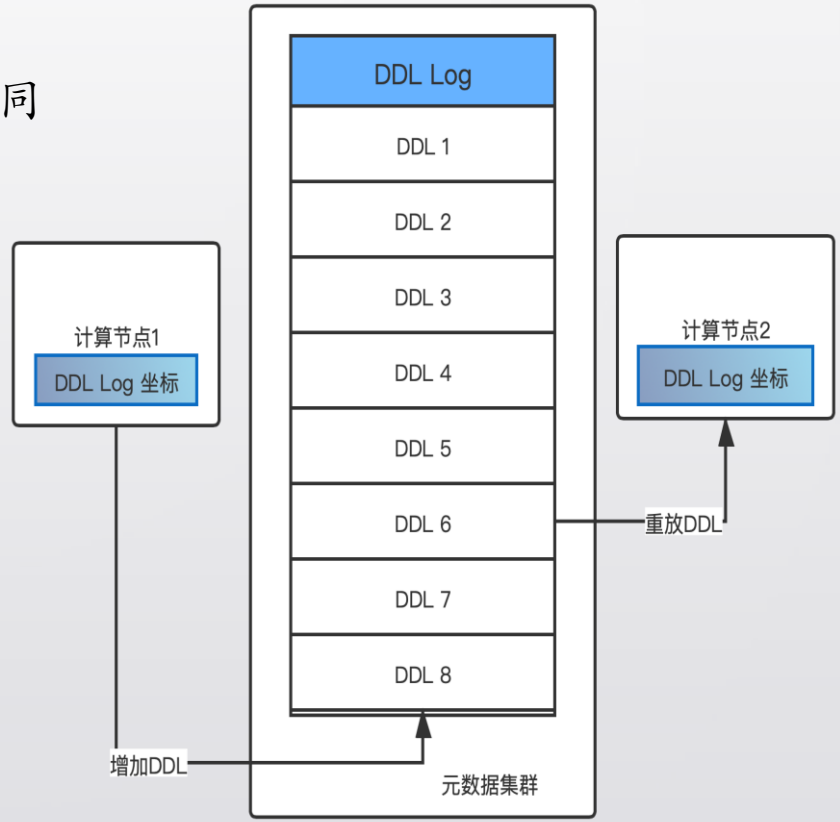
昆仑DDL事务与集群全局元数据一致性 -- 错误处理



昆仑计算节点DDL语句复制及其一致性保障

- DDL语句复制
  - 从元数据集群的DDL log获取
  - 复制其他计算节点执行过的DDL
  - DDL在所有计算节点并发执行
  - DDL在所有计算节点实际执行顺序全局相同
  - 复制过程随时可以中断和恢复续传
    - 复制坐标：pg_ddl_log_progress.ddl_op_id
    - 没有遗漏和重复
  - 每个DB一个复制进程

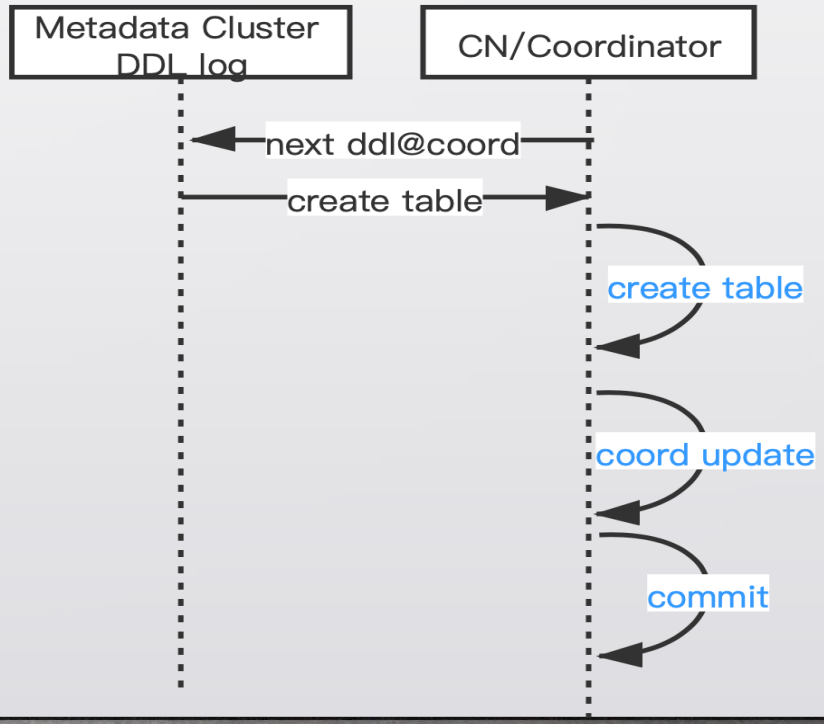
```
postgres=# select*from pg_ddl_log_progress;
  dbid   | ddl_op_id | max_op_id_done_local
---------+-----------+----------------------
       0 |         0 |                    0
   13160 |      3064 |                 8926
  131074 |      3044 |                 3042
 1156377 |      8572 |                 8913
  262147 |      3062 |                 3062
  262148 |      3063 |                 3063
  270336 |      3064 |                 3064
(7 rows)
```



昆仑计算节点DDL语句复制及其一致性保障

- DDL语句复制的一致性保障
  - 每一个DDL语句复制任务作为一个事务来执行：
    - 执行DDL
    - 更新本地执行坐标 pg_ddl_log_progress.max_op_id_done_local
    - DDL复制事务在任意位置中断：事务回滚
  - 总是从复制坐标开始复制
    - pg_ddl_log_progress.ddl_log_id

```
postgres=# select*from pg_ddl_log_progress;
  dbid   | ddl_op_id | max_op_id_done_local
---------+-----------+----------------------
       0 |         0 |                    0
   13160 |      3064 |                 8926
  131074 |      3044 |                 3042
 1156377 |      8572 |                 8913
  262147 |      3062 |                 3062
  262148 |      3063 |                 3063
  270336 |      3064 |                 3064
(7 rows)
```



Thank You