KunlunBase

KunlunBase Fullsync原理

赵伟

泽拓科技（深圳）有限责任公司

KunlunBase

目 录

KunlunBase

01 为什么需要Fullsync

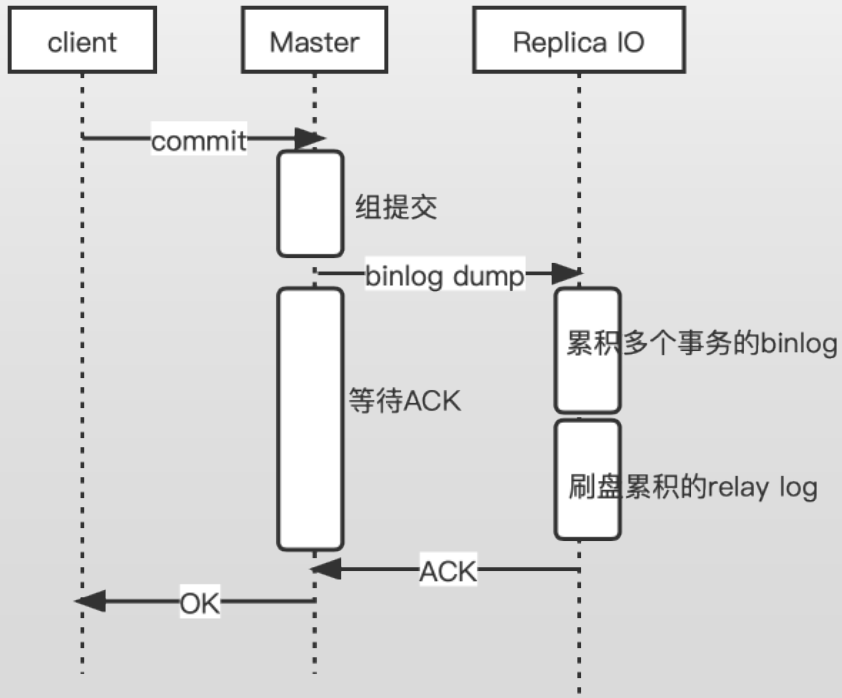
- **可能的故障**
  - 主节点硬件故障，断电，OS重启，误杀进程
  - 主备节点磁盘坏块，磁盘全毁
  - 网络分区、断连、拥塞
- **金融级高可靠性的要求**
  - 故障时数据库集群可以读写
    - 自动探测主节点状态，发现主节点故障时自动选主和主备切换
  - 故障时不丢失数据
    - 一致性（Consistency） > 可用性（Availability）
  - 高性能，低延时
  - 集群长期不间断工作自维护

01 为什么需要Fullsync

- **Semisync的问题**
  - 主节点故障后不可以重新加入集群 --- 集群节点数量无法快速恢复
  - 备机ACK超时自动退化为异步，一致性(Consistency) < 可用性(Availability)
    - 无法实现金融级强一致性
  - 备机ACK前没有fsync relay log
    - 备机OS重启（例如 机房/机架/服务器断电）会导致已确认事务的binlog丢失
  - 占用工作线程等待备机ACK，连接较多时需要启动大量线程
  - 不完备，需要外部组件配合实现高可用
    - 只确保备机收到binlog，不能完成主节点故障发现，选主和主备切换
- **MySQL Group Replication的问题**
  - 备机确认收到binlog后，主节点才写入binlog到binlog文件
    - 好处：主备节点同步前进，并且其binlog始终保持一致
    - 问题：事务提交前持锁等待时间很长，阻塞冲突的事务，并显著增大本事务延时
  - 占用工作线程等待备机ACK，连接较多时需要启动大量线程

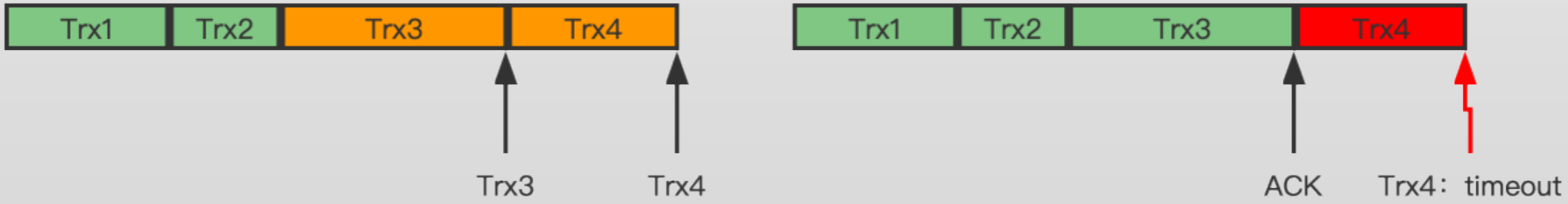
02 Fullsync的基本原理

- **主节点上事务完成提交后等待备机收到其binlog的ACK才返回给客户端（kunlun-server）**
  - 主节点上完成提交的任何事务都有至少1个备机（可以配置）收到其完整的binlog并持久存储
  - 等待ACK的时机：主节点上事务完成提交流程后返回状态给客户端之前
    - 已经完成3阶段提交流水线
- **备机汇集多个事务的binlog一次性写入relay log文件并且刷盘（fsync）然后发送ACK**
- **需要等待fullsync ACK的语句**
  - 所有做binlog事务提交语句
  - DDL语句，autocommit DML语句
  - commit，XA COMMIT ... ONE PHASE
  - XA PREPARE



03 Fullsync机制主节点逻辑

- **主节点上事务等待Fullsync ACK的细节**
  - 等待的内容：本事务已经被足够的备机收到并刷盘，ACK的binlog位置 >= 本事务的binlog位置
  - 等待的方法： 不阻塞占用工作线程，挂起会话（THD）直到ACK到来
    - 工作线程继续去处理其他连接的请求
  - 等待ACK 超时后的处理：Fullsync HA更加健壮可靠灵活
    - 考虑网络或者IO 随机偶发的抖动
    - 支持选择优先 Consistency 或者 Availability
  - 收到ACK的处理：某个后台工作线程执行目标会话（THD）的事务提交流程中的剩余操作
    - 发送OK包给客户端，客户端语句返回
  - 等待ACK超时的处理：在会话中返回 “fullsync等待超时” 错误给客户端



03 Fullsync机制主节点逻辑

- **可靠性与一致性级别（fullsync_consistency_level)**
  - 2*N + 1个节点的集群，应该设置 fullsync_consistency_level为N
    - 可以抵抗N个节点同时故障 --- 至少还有1个节点含有最新事务的数据
  - 实践：用期望的一致性级别（N）来规划集群节点数量（2*N+1）
- **超时后的其他处理（disable_fullsync_on_slave_ack_timeout）**
  - true：退化为异步
  - false：设置实例为只读
  - 默认不启用该处理，由cluster_mgr来处理
- **重要的状态变量**
  - fullsync_replica_ack_upto_file & fullsync_replica_ack_upto_offset
  - fullsync_replica_fully_acked_upto_file & fullsync_replica_fully_acked_upto_offset
-

03 Fullsync机制备节点逻辑

- **备机接收binlog的处理逻辑**
  - 在磁盘负载、性能以及数据一致性 之间取得平衡
  - 以事务为单位write&fsync binlog
    - fsync：断电或者OS重启后binlog仍然持久保存
  - ACK：主节点binlog上的一个位置
    - 备机发送ACK：该位置之前的binlog已经持久保存，其所属事务可以返回成功
  - ACK的发送方法：SQL 语句 或者 扩展的 client API （发送命令）
    - SLAVE server_id CONSISTENT TO file_index offset
      - general log默认不记录该语句
    - mysql_send_binlog_ack() （COM_BINLOG_ACK）
      - 更快，但是需要KunlunBase的mysql client lib

03 Fullsync机制备节点逻辑

KunlunBase

- **备机fullsync控制**
  - fullsync_relaylog_fsync_ack_level：灵活安排备机节点的功能和角色

fsync ack level	fsync relay log	发送ACK
0	否	否
1	否	是
_x0004_2	是	是

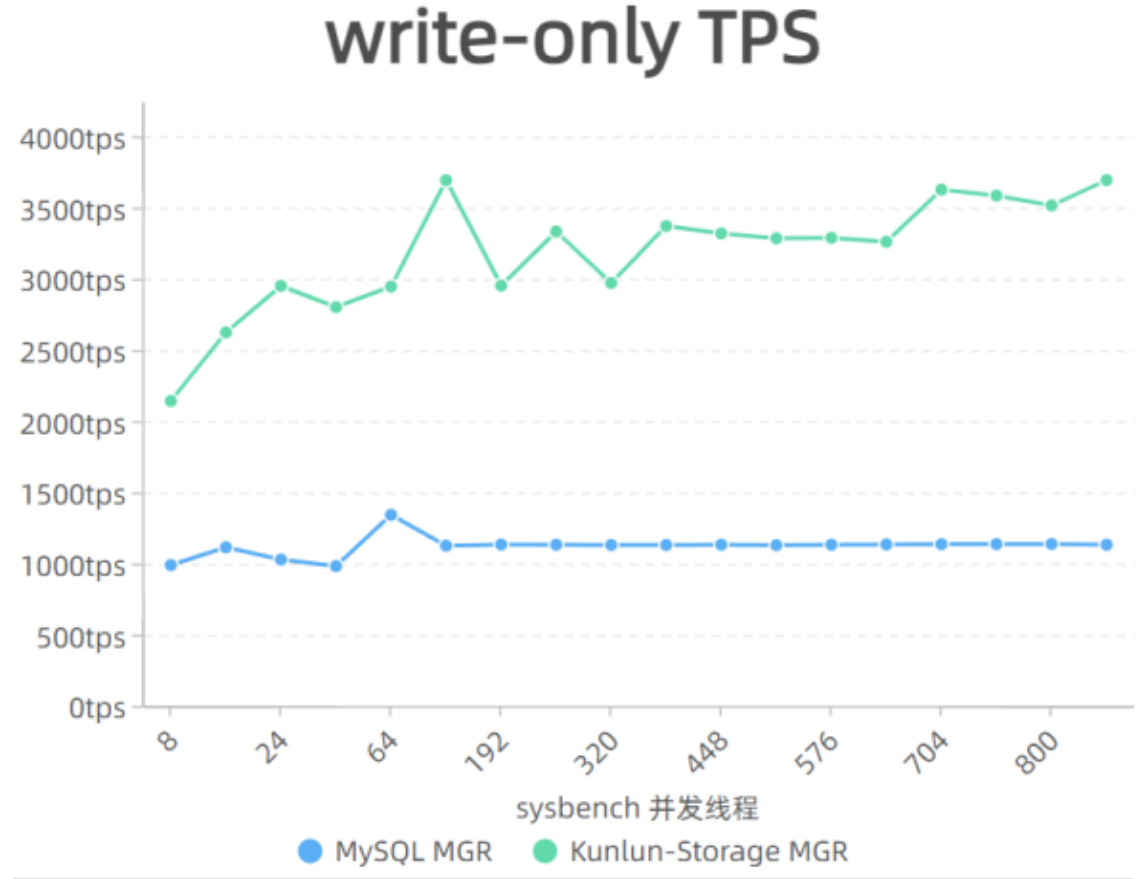
  - skip_fullsync_replica_acks_older_than
    - 不ACK老事务，避免不必要地消耗IO网络资源
  - 多个事务成组 fsync&ack：降低IO开销
    - 累积足够数量的binlog：fullsync_fsync_ack_least_event_bytes
    - 或者 累积足够数量的事务：fullsync_fsync_ack_least_txns
    - 避免等太久：fullsync_fsync_ack_wait_max_milli_secs
    - 观察效果：fullsync_num_txns_in_acked_group

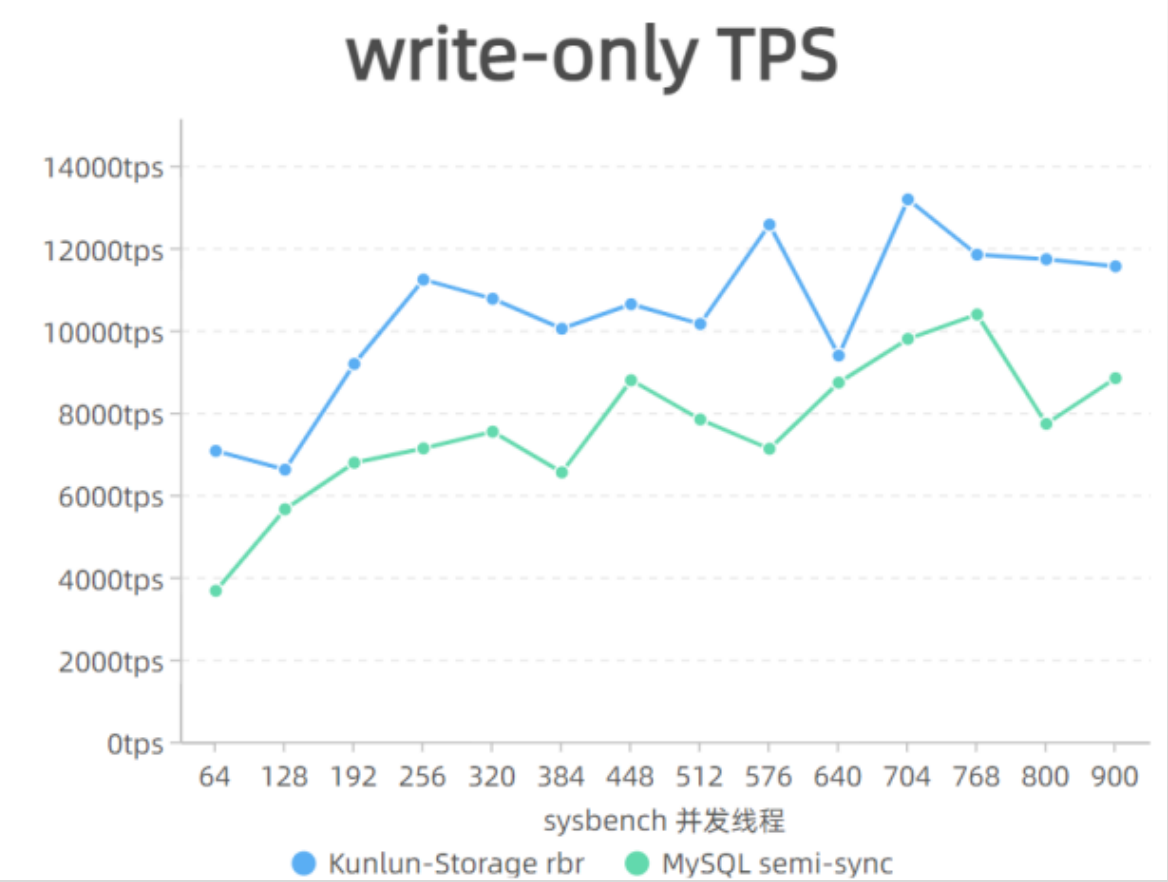
04 Fullsync HA简介

- 主节点可用性检查
  - 定期写入心跳
  - 写入失败的处理
- 选主
  - 条件：参与者数量，拥有最新数据
- 主备切换
- 旧主节点加入
  - 为什么需要闪回

04 Fullsync 性能

- sysbench写入测试 kunlun-storage fullsync 10倍于社区版MySQL MGR，高于社区版MySQL semisync 30%-100%





KunlunBase

Thank you

Q & A