Double Sync Replication

——对MySQL原生复制可靠性的改进

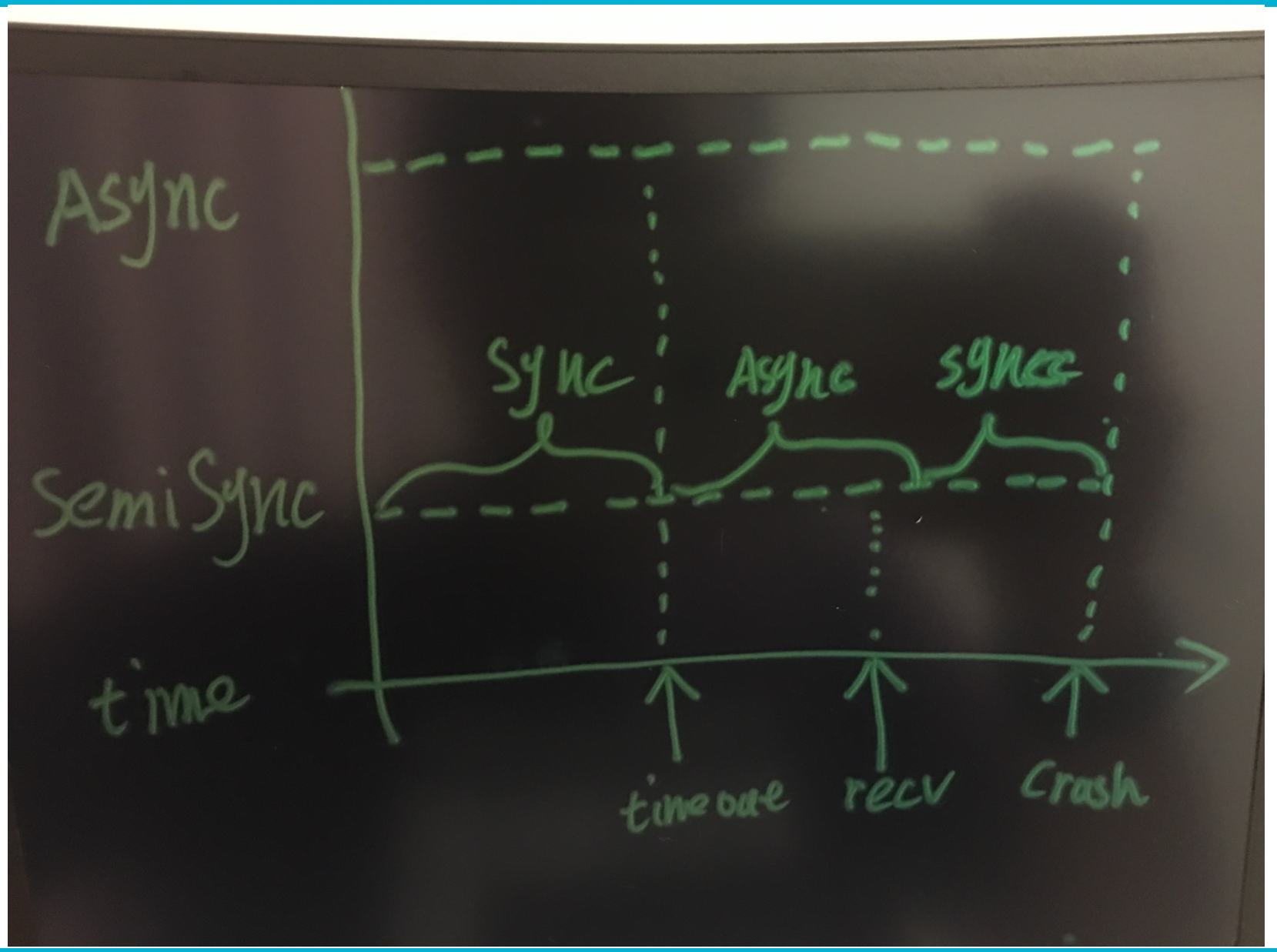
阿里巴巴云计算 彭立勋

异步复制/SemiSync存在的问题

- 备库无法及时得知主库的状态
  - 主库事务提交并不需要备库ACK
  - 备库无法得知拖取的是否是最新的日志
  - 宕机后无法利用备库信息得知是否跟主库一致

- SemiSync是否解决了这个问题
  - 主库事务提交需要备库ACK
  - 网络超时后备库降级为异步复制
  - 超时设太小，则经常发生超时
  - 超时设太大，则经常导致主库hang
  - 网络恢复后需要追赶日志，追赶期间备库状态依然不可知
  - 无法得知宕机时备库是否跟主库是SemiSync状态
  - 依然无法得知备库是否跟上主库

异步复制/SemiSync存在的问题



我们要达成的目标

- 在主机（可用性5个9）、网络（可用性5个9）没有同时宕机时，备库可以得知自己的状态（跟主库同步 或 没有跟主库同步）
- 在确认跟主库不同步时，通知应用参与数据补偿，并且告知所缺数据范围
- 在确认跟主库同步时，可以保证备库执行到跟主库一致状态
- **核心：避免备库状态不可知！**

攻破SemiSync的缺点

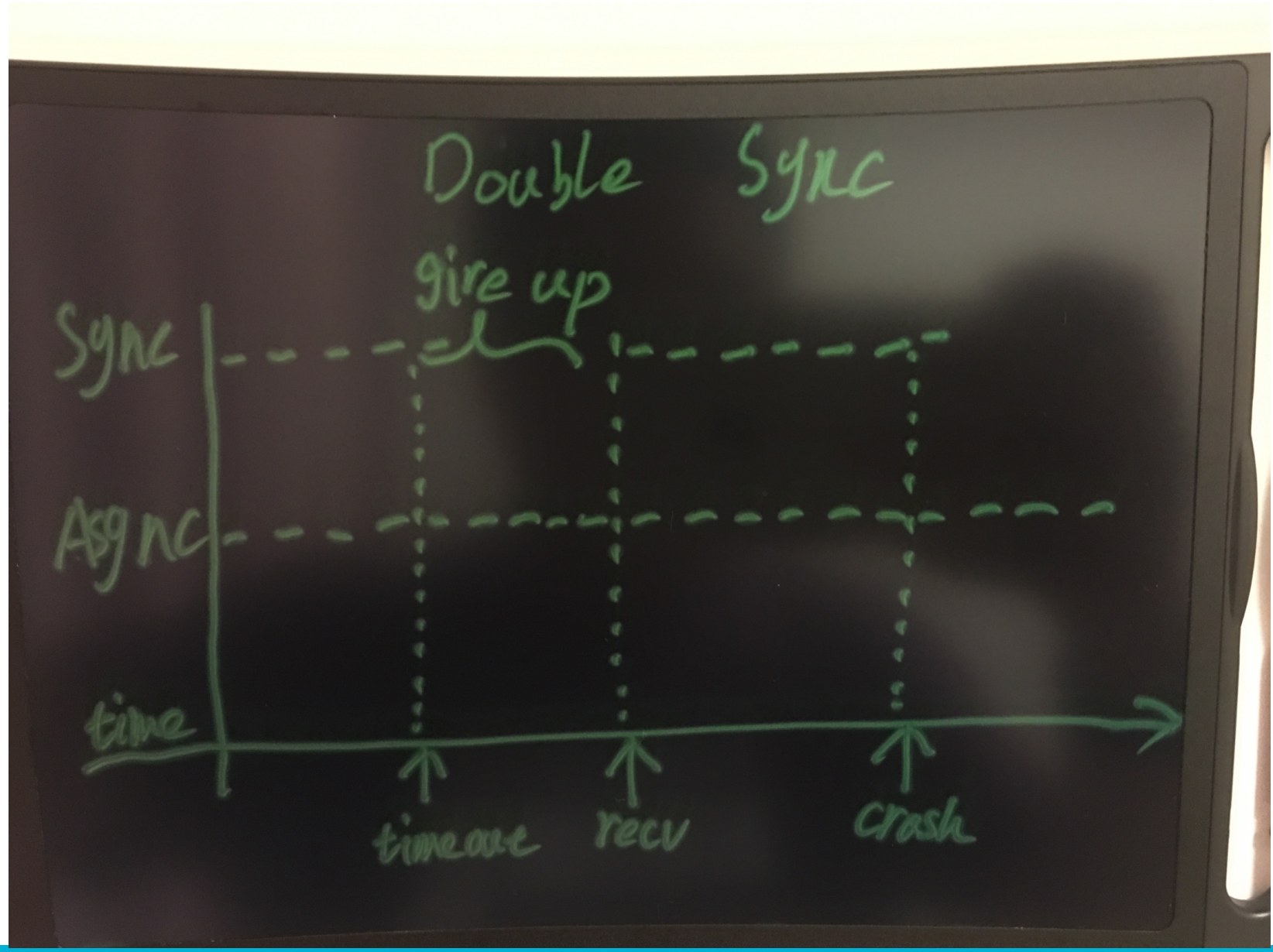
- SemiSync一旦超时断开，即使网络恢复，依然需要补偿拖取断开期间的日志
  - 如果SemiSync超时断开，网络恢复后不再补偿数据，只发最新日志，如何？
  - 只要宕机时网络正常，备库始终会知道主库最新位点
  - 依此可以判断备库是否跟主库日志有差异

- 备库如果只接收最新数据，那么中断期间的数据如何处理？
  - 异步复制可以在不影响主库的情况下拖取日志
  - 利用异步复制的日志可以进行完整的日志回放

结合两种复制

- 异步复制
  - 拖取连续日志，保证备库接收的日志不中断
  - 接收到日志后直接执行
- 半同步复制
  - 拖取最新日志，保证备库始终知道最新的日志位置
  - 接收到日志后并不执行，只保留位置
- 一致性判断
  - 比较异步复制和半同步复制的日志段，可以判断备库日志可否连续接上

结合两种复制



两个通道如何做到？

- 多源复制可以在一个Slave上创建多个独立通道分别进行复制

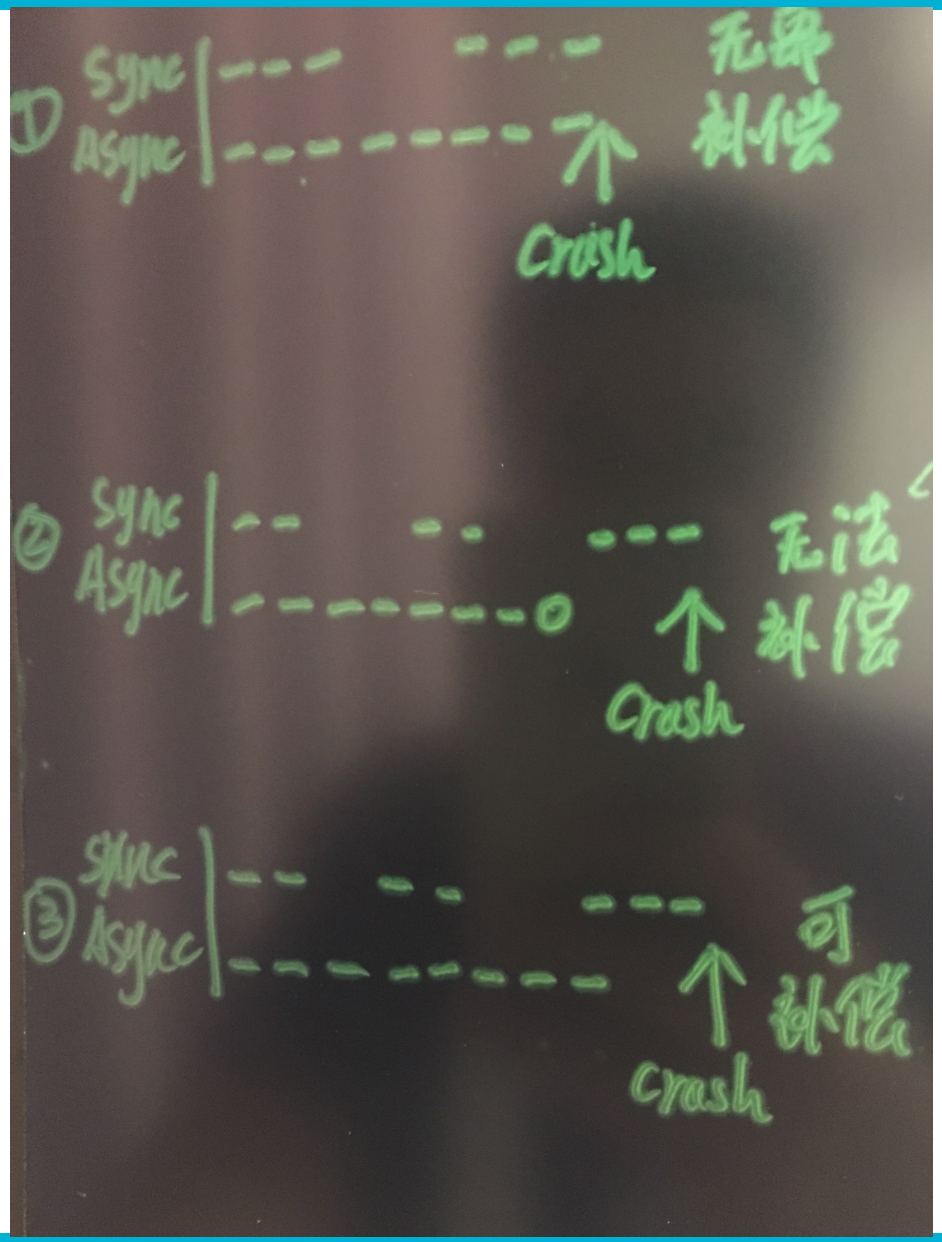
- 问题1：同一个ServerID发起两个通道到Master，Master会认为是原Slave断开没有主动发起close连接，从而会踢掉先连上的通道
- 解决：可以将SemiSync通道伪装一个ServerID，避免被踢

- 问题2：一个Slave同时有一个非SemiSync通道和一个SemiSync通道，而SemiSync设置是保存在全局的
- 解决：把SemiSync改为Per-Channel的设置，将SemiSyncSlave类转移到Master_info结构体中

如何判断两个通道日志是否连续

- 利用两个通道收到的GTID序号作对比
- 利用两个通道收到日志的Log_file_name和Log_file_pos
- 如果半同步通道的日志起始点小于等于异步通道结束点，那么备库其实有完整的日志，反之备库无法跟上主库

如何判断两个通道日志是否连续



如何补偿数据

- 利用半同步通道收到的日志，在异步通道应用完日志后，启用半同步通道应用日志
- 利用GTID来过滤重复Event

THANKS!