# how to make withdraw operation and update balances in transaction? #80

New issue

Open **LittlePeng** opened this issue on Apr 19, 2018 · 5 comments

**LittlePeng** commented on Apr 19, 2018 • edited

比如用户要提现 100BTC，这时需要写一条提现记录和给用户balance上减去100BTC，要更新的两条记录分别在MySQL和matchengine ， 我理解的应该使用2PC来完成：

- 1: RPC CMD_BALANCE_QUERY 确认用户余额，如果余额足够，则写入 prepare withdraw 记录
- 2: RPC CMD_BALANCE_UPDATE 扣款，如果：
  - 成功: commit withdraw 更新
  - 失败: abort withdraw 取消
  - 超时: 使用相同business_id重试

存在几率：2PC完成用户提现成功，但CMD_BALANCE_UPDATE 操作还处于 me_persist.c 队列未能落地，如果此时matchengine宕机，balance就没能扣款。

想到个解决思路是：由于问题在于CMD_BALANCE_UPDATE返回成功时并不能确保balance扣款成功。那么就不依赖CMD_BALANCE_UPDATE返回成功， commit withdraw时把operlog在事务里面也写一遍不就行了么。operlog被多记录一次也没事，business_id 保证更新操作的幂等性，但发现operlog 的id字段由matchengine生成不是自增主键，并不能这么做。

另外一些想法：

1. 保证写操作持久性
   现在balance和orders的写入RPC返回时，并不保证数据落地，操作有:CMD_BALANCE_UPDATE、CMD_ORDER_xx, 最多可能会丢失 MAX_PENDING_OPERLOG 100条记录，而且matchengine 宕机后balance_history、order_history、deal_history 可能数据还可能对不齐。

   可借鉴 MySQL group commit 实现，写操作等对应的日志刷库成功后再异步返回，保证持久性的同时也不影响吞吐量。当然增加点延时在所难免，正常也能控制在100ms内。

2. matchengine 支持 slave 模式
   matchengine 是单点、而且是单线程，需要处理整个交易所的所有balance和order请求，吞吐量存在瓶颈应该给减轻负担。很多时候balance、order读操作都是允许一定的延时，如果matchengine 支持slave 通过kafka replication到多个节点以提供读操作。

👍 2

**haipome** commented on Apr 24, 2018 Member

**Assignees**

No one assigned

**Labels**

None yet

**Projects**

None yet

**Milestone**

No milestone

**Development**

No branches or pull requests

**3 participants**

1：关于“CMD_BALANCE_UPDATE返回成功时并不能确保balance扣款成功”这个问题确实存在的，实现时对数据的完备性和效率做了一些取舍。未来考虑通过更好的办法来保证操作记录落地。

2：“而且 matchengine 宕机后balance_history、order_history、deal_history 可能数据还可能对不齐。”这个问题也是存在的，但相对于问题1没有那么严重。

3、这个是在计划中的。

👍 1

**LittlePeng** commented on Apr 25, 2018 • edited Author

多谢回答，期待新的版本。Great Work，这个项目构架思路真的很赞👍👍👍👍

本想交易引擎如果如果不采用IOE，就得引入复杂分布式事务系统。 关键在于全量的 balance 和 orders 数据量很小，通过可以通过本地内存事务来实现；好像也可以用mnesia集群或者用全量缓存的rocksdb(LSM-Tree类存储引擎)+Raft来搞。

👍 1 😄 1 ❤️ 1

**tornodo** commented on May 9, 2018

目前的情况是如何临时解决这俩问题？🤦
等出了意外再给用户解释就晚了。😭

**haipome** commented on May 11, 2018 Member

系统设计的时候在效率和一致性做了取舍，选择了效率。
系统本身是严格按照顺序对 operlog 进行落地的，如果真的出现了异常情况，会丢失最新不超过1秒的交易记录，这个是可以接受的。

❤️ 2

**tornodo** commented on May 11, 2018

收到，感谢。😄🙏