PhxSQL设计与实现

高可用强一致的MySQL集群

主讲人：陈俊超

目录

- 个人介绍
- 背景
- 思路
- 实现
- 效果

个人介绍

- 个人信息
  - 2011年加入腾讯微信
  - 微信后台中心高级工程师

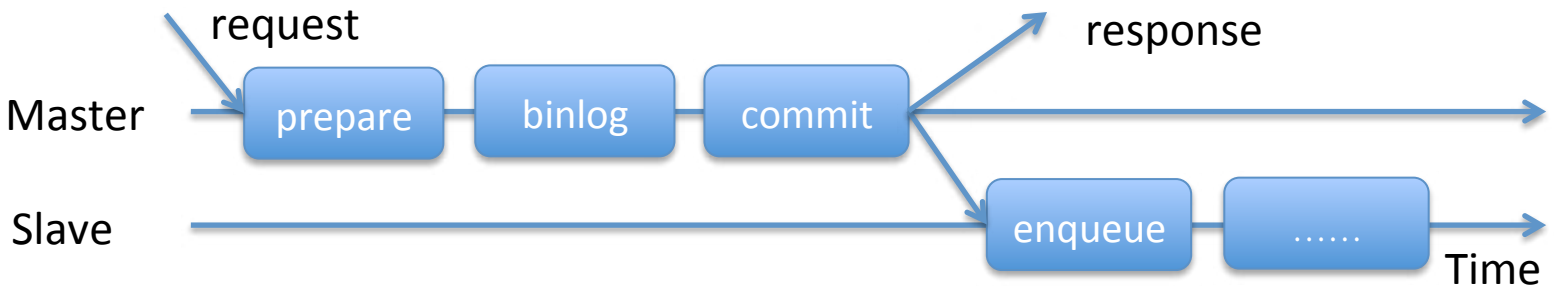
- 个人经历
  - 负责微信后台核心模块的设计和开发
    - 查看附近的人，摇一摇，朋友圈
    - 微信群聊，关系链
  - 负责微信开源项目PhxSQL的设计和开发

原生MySQL的容灾缺陷

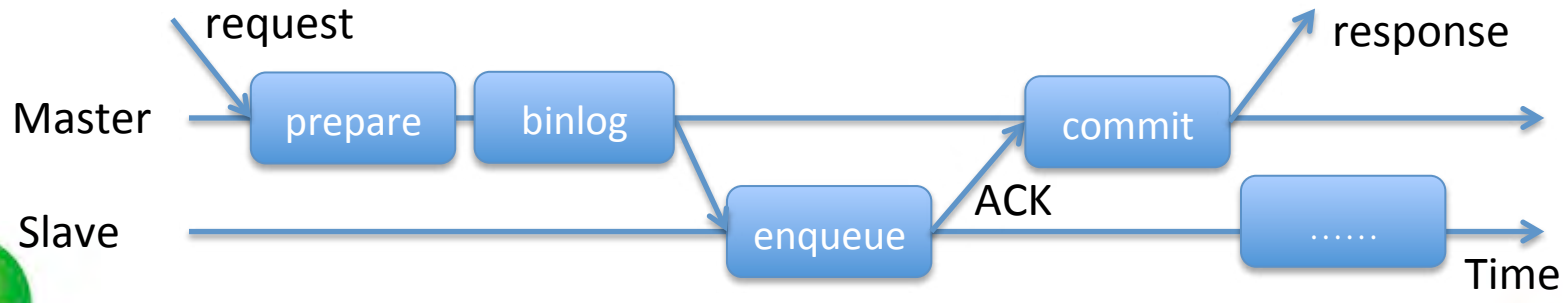
无法同时满足高可用和强一致

MySQL常见复制方案

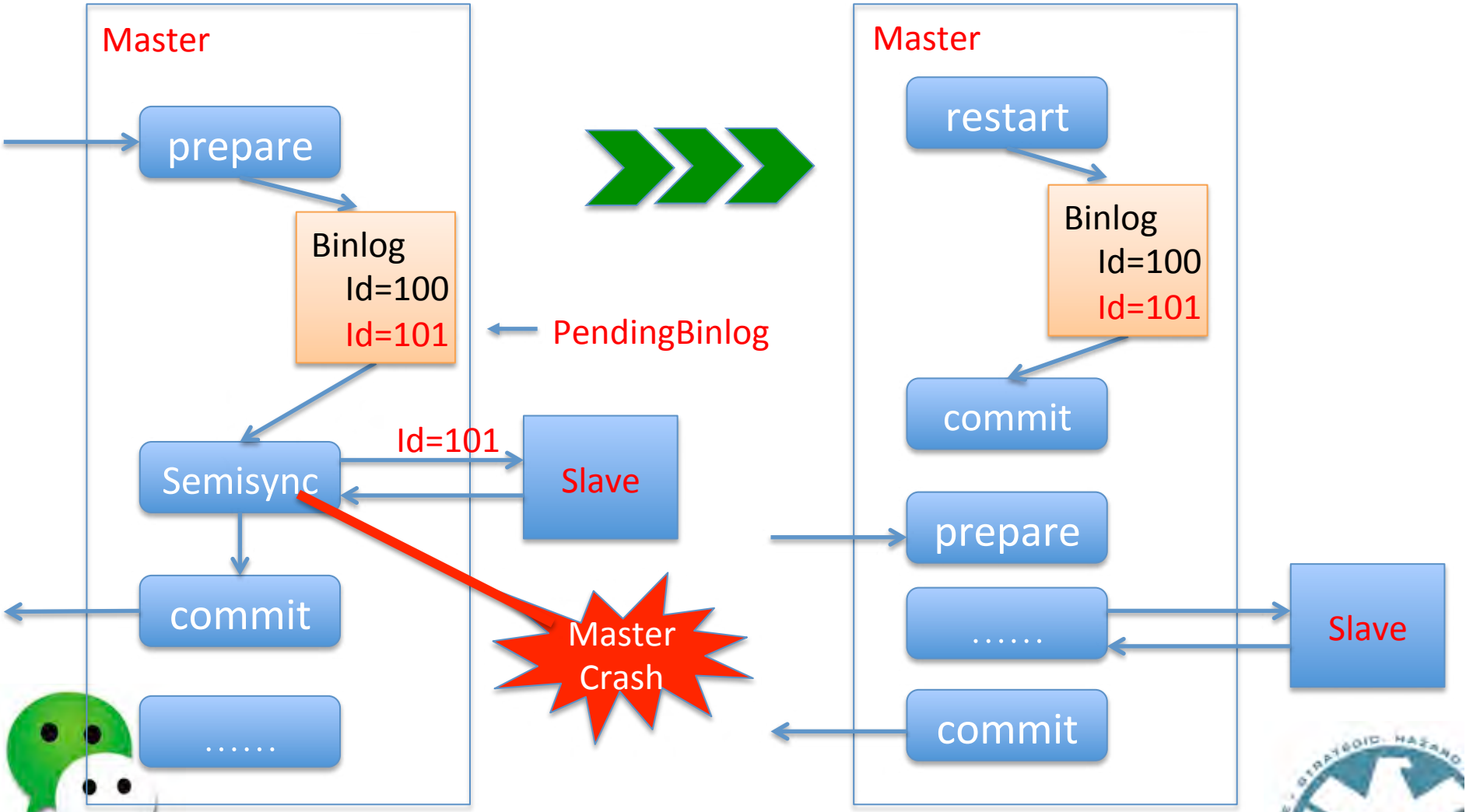
- 异步复制无法保证主备数据一致



- MySQL 5.7半同步增强了主备数据一致

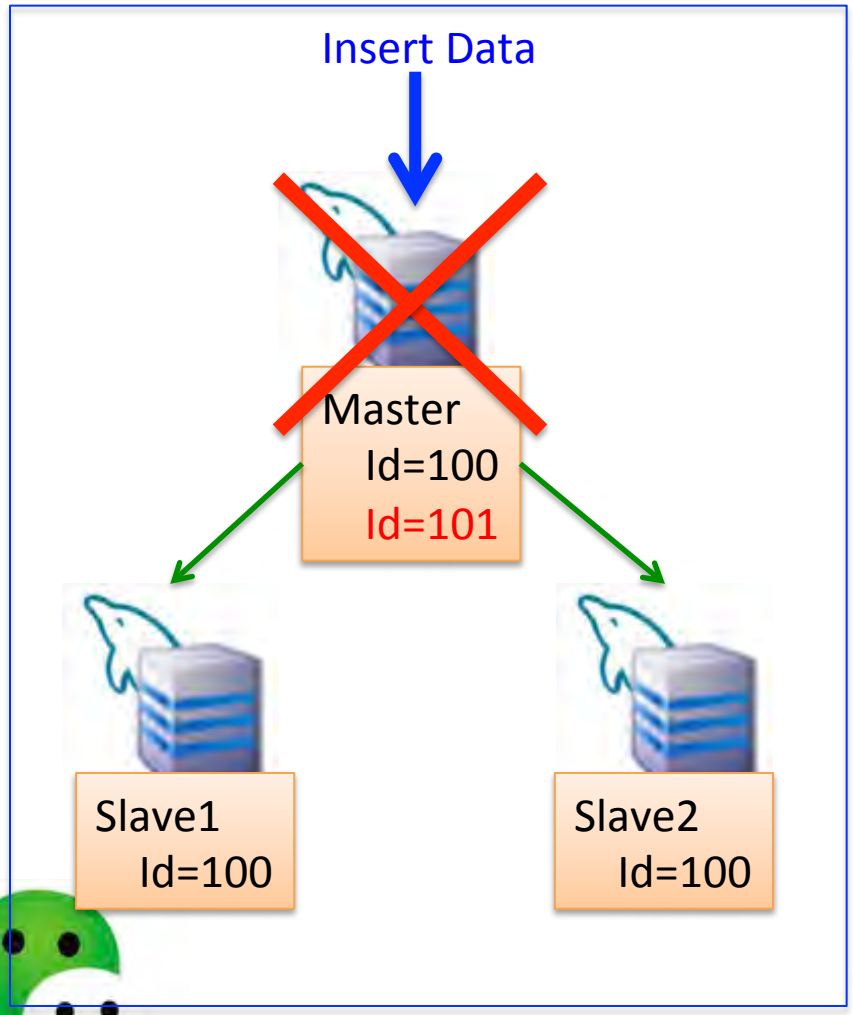


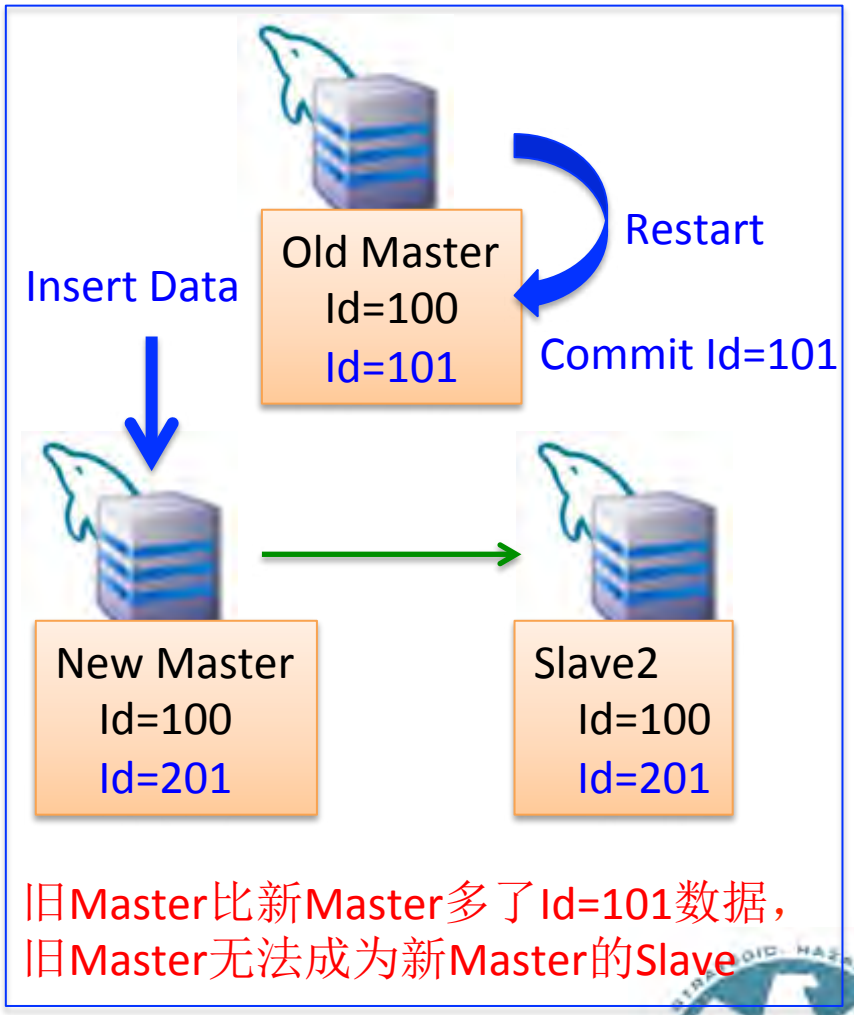
重启流程不执行半同步



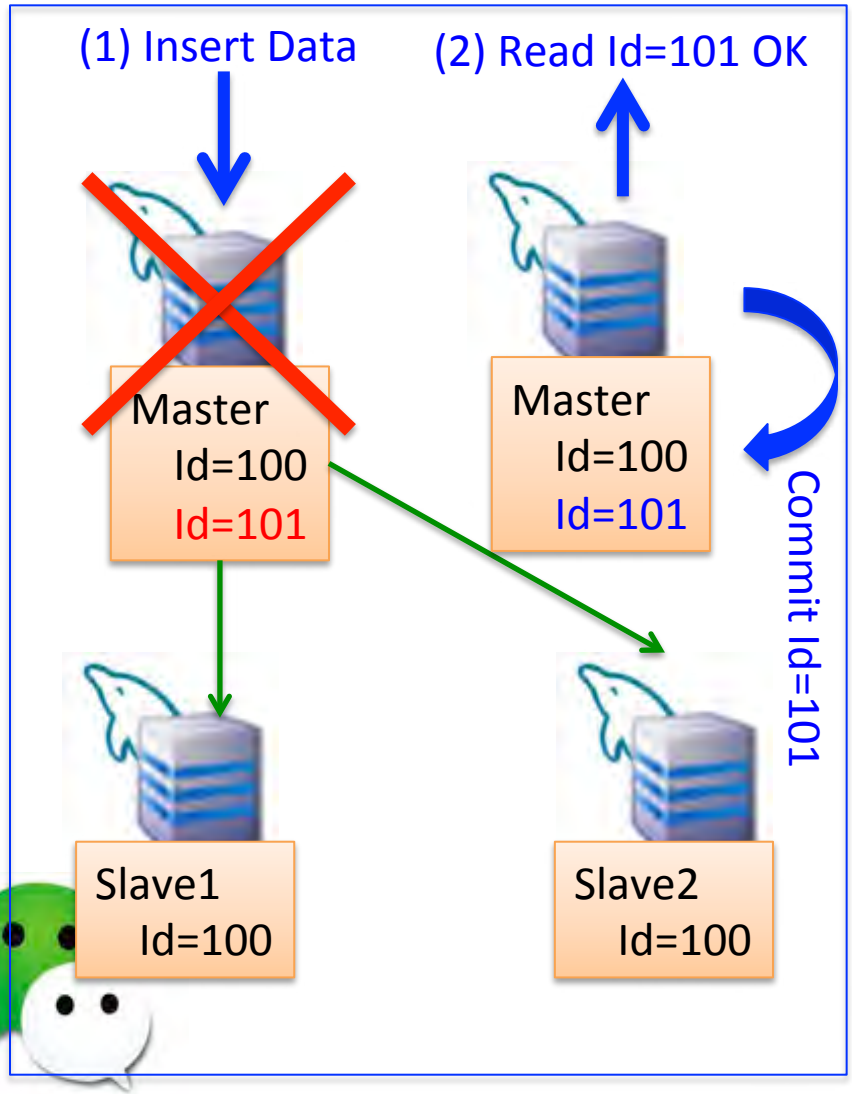
- http://bugs.mysql.com/bug.php?id=80395
- https://mariadb.atlassian.net/browse/MDEV-162

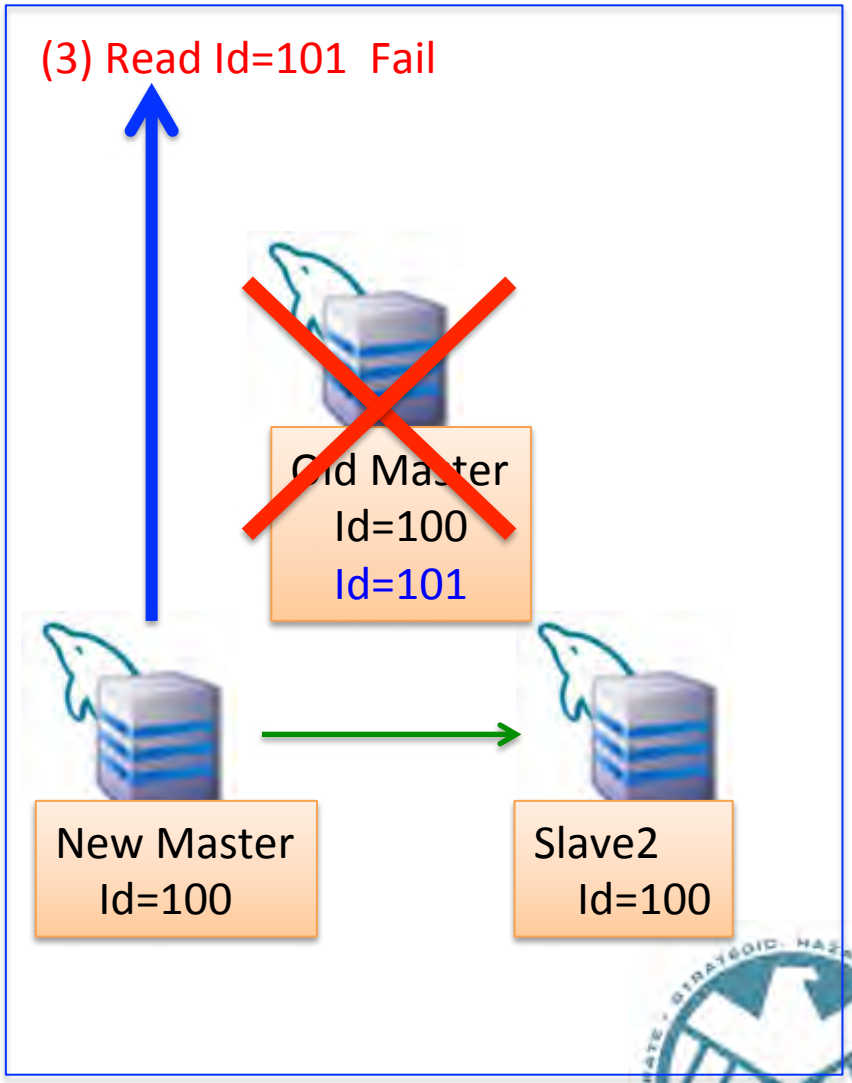
Master切换导致数据不一致



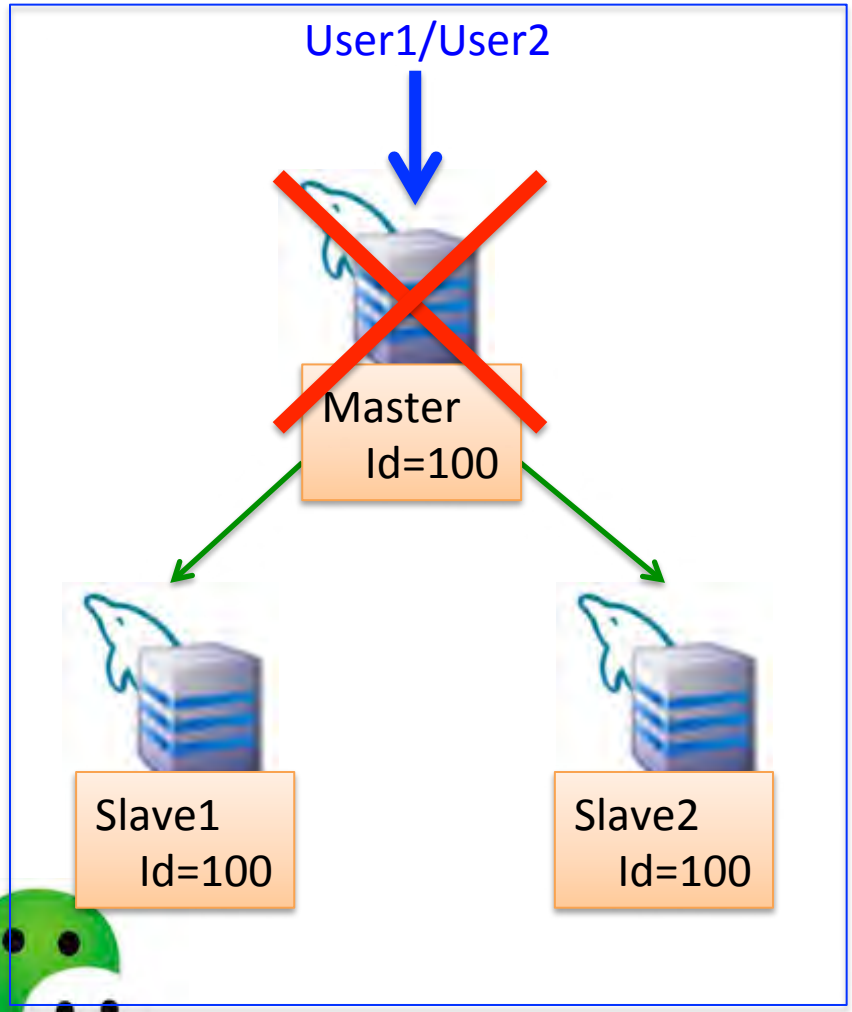


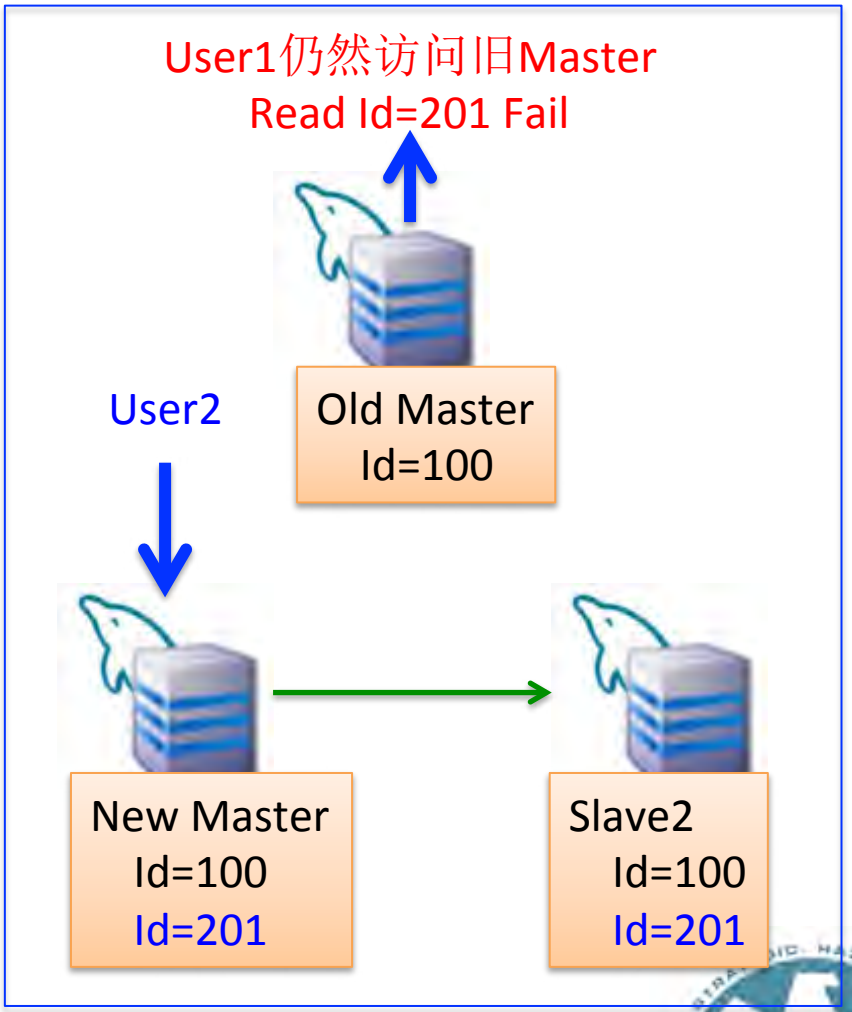
Master切换导致幻读



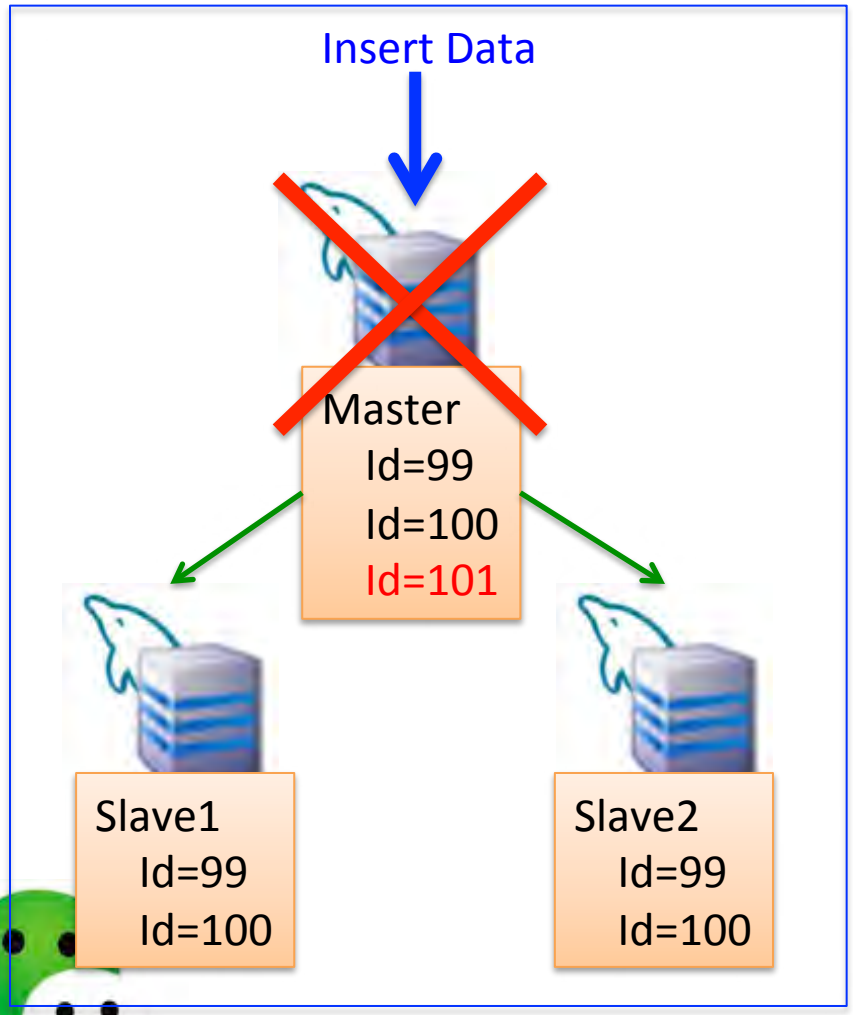


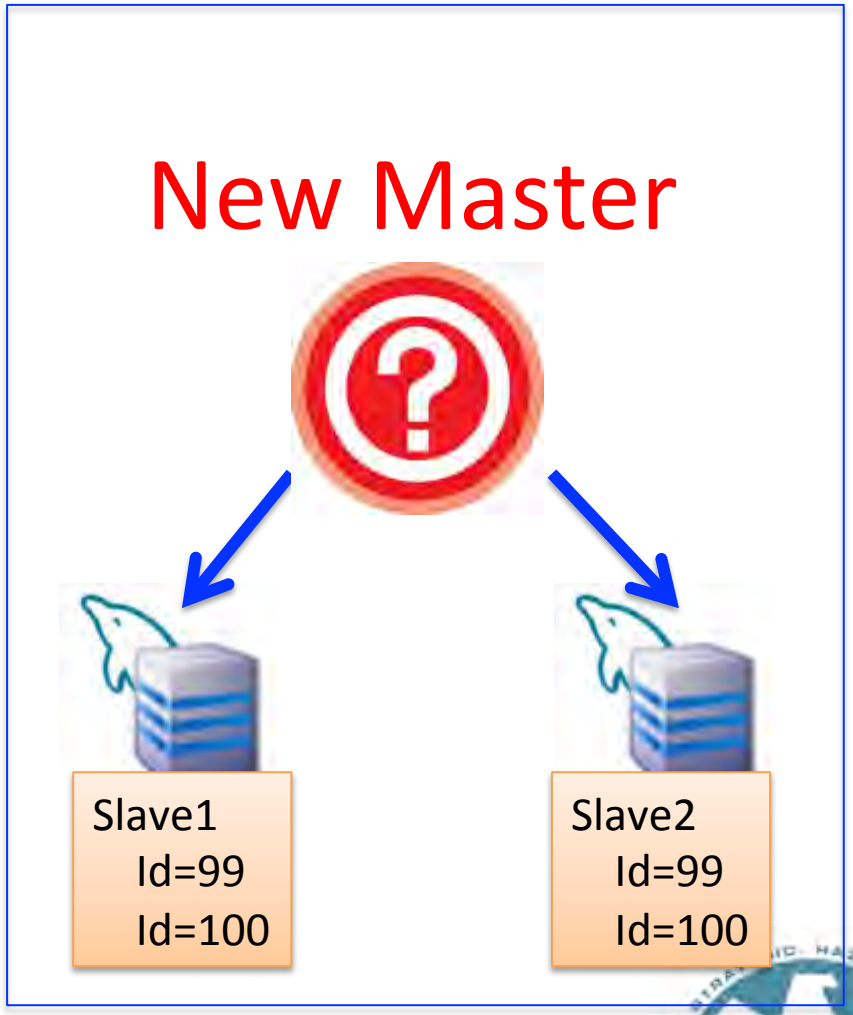
Master切换导致调用端分裂





MySQL缺乏自动选主机制





MySQL无法同时满足高可用和强一致

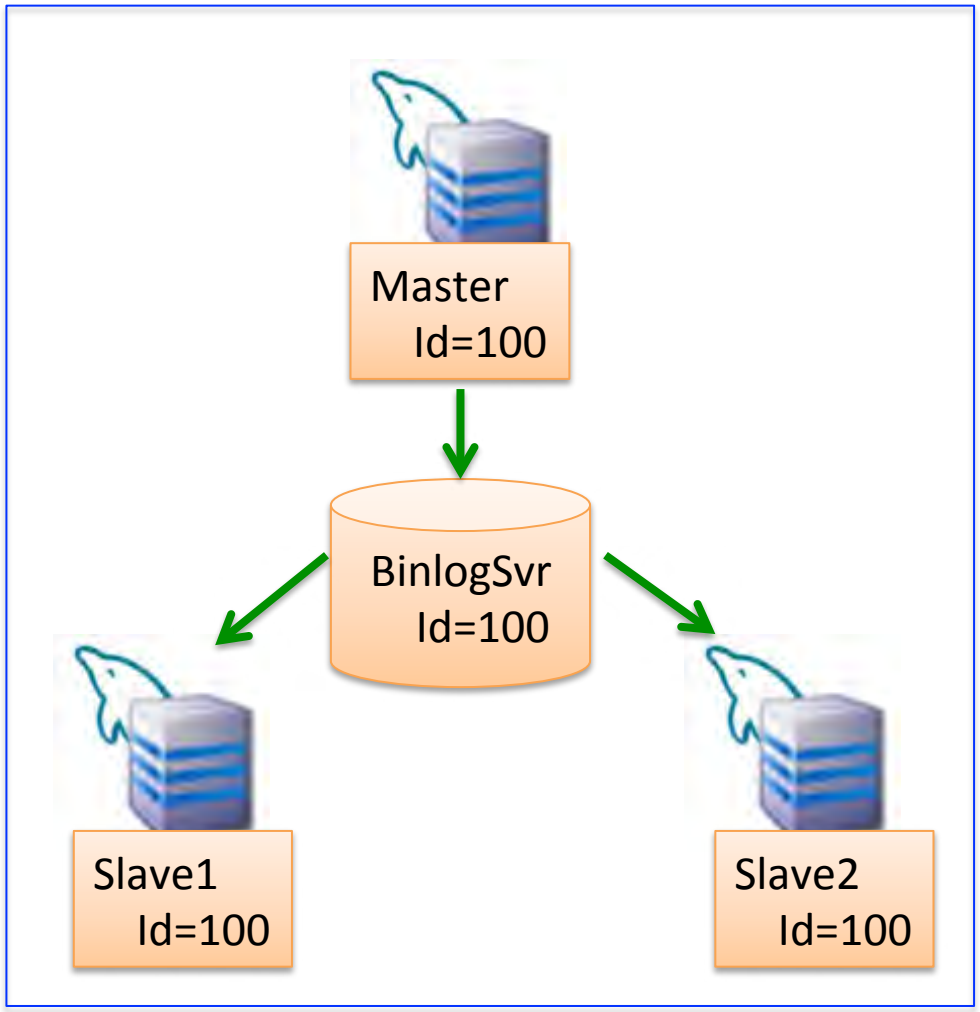
- 主备数据不能保证一致
  - Master重启导致主备数据不一致
  - Master重启导致幻读
- 不能保证高可用
  - Master切换导致调用端分裂
  - 缺乏自动选主机制

- 二选一
  - 要强一致，不做主备切换
  - 借助MHA实现高可用，容忍主备数据不一致

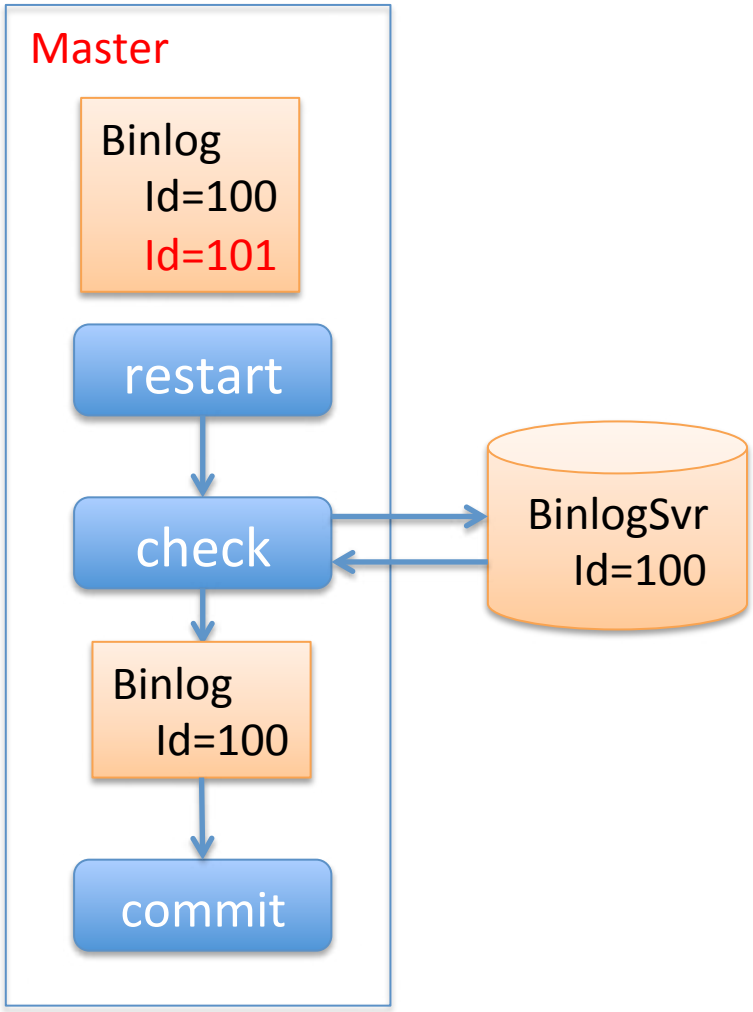
PhxSQL设计思路

可靠日志存储和请求透传

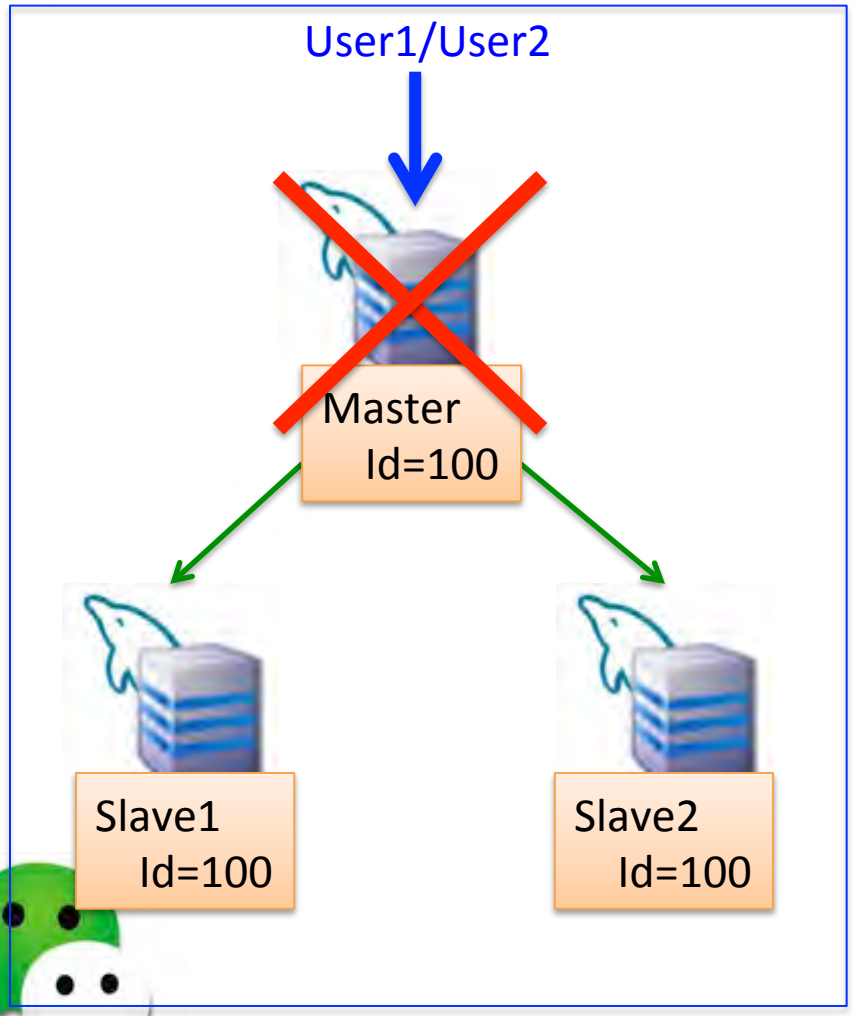
以“可靠日志存储”为中心的架构

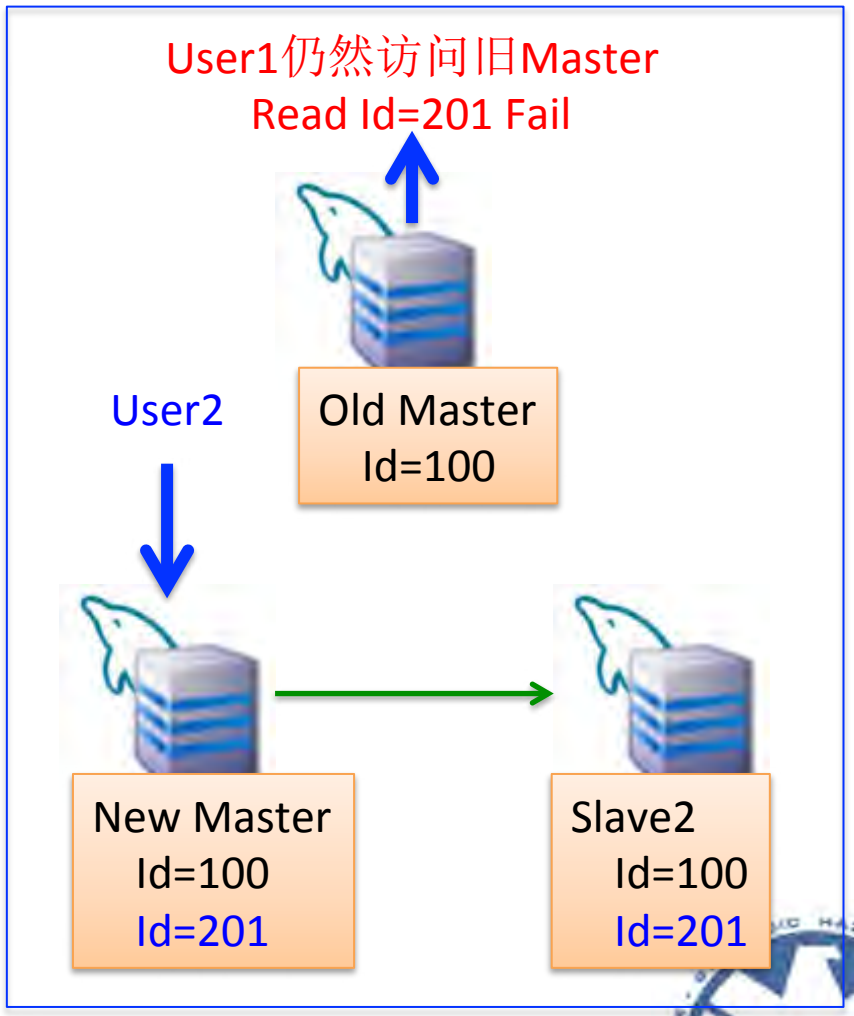


用“可靠日志存储”校准重启过程

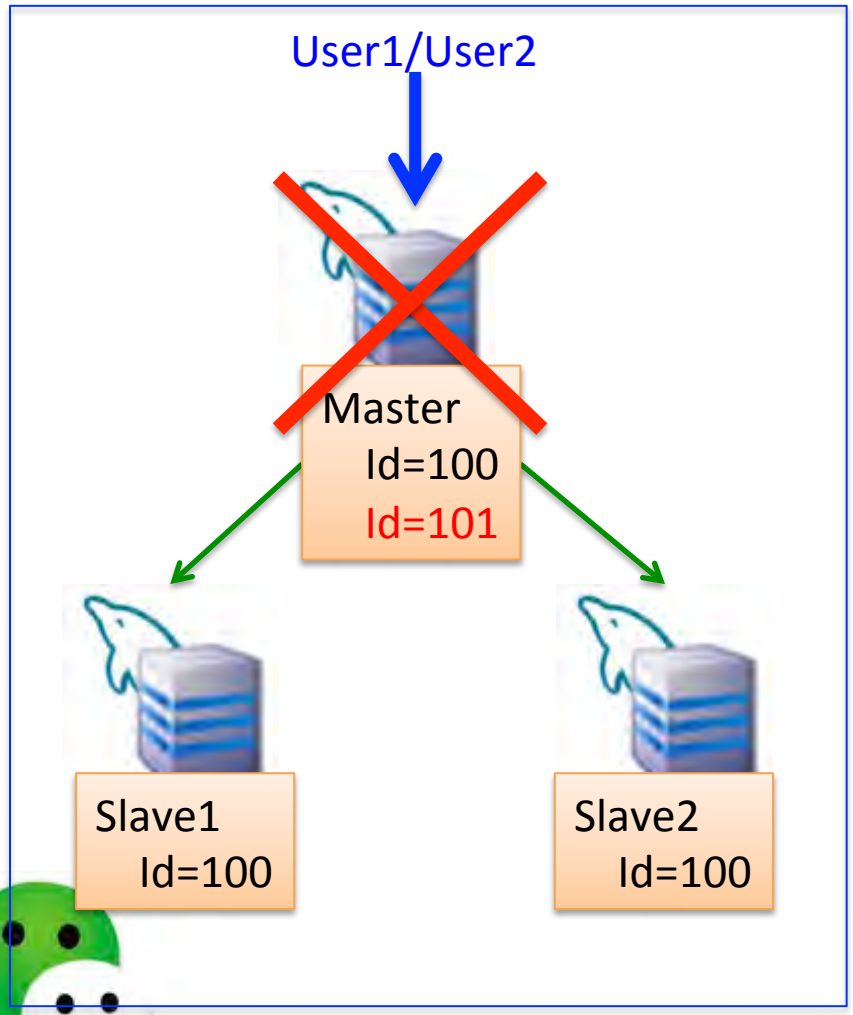


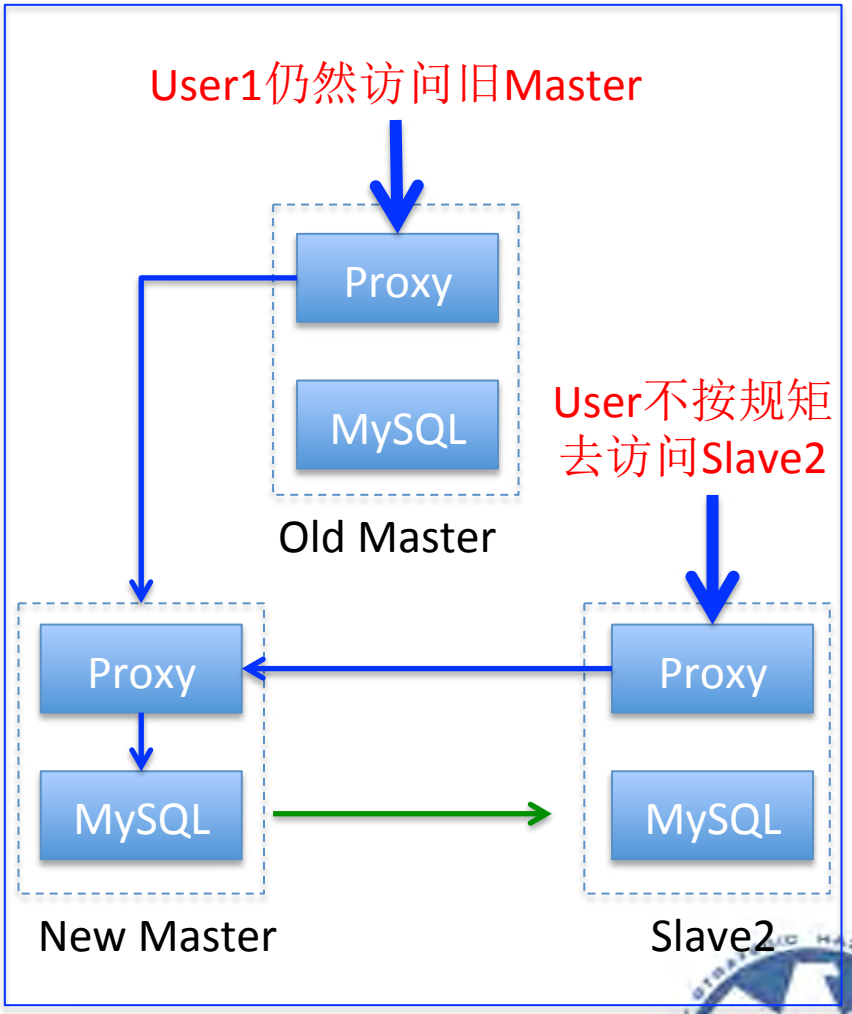
回顾：调用端分裂场景


User1/User2
Master
Id=100
Slave1
Id=100
Slave2
Id=100

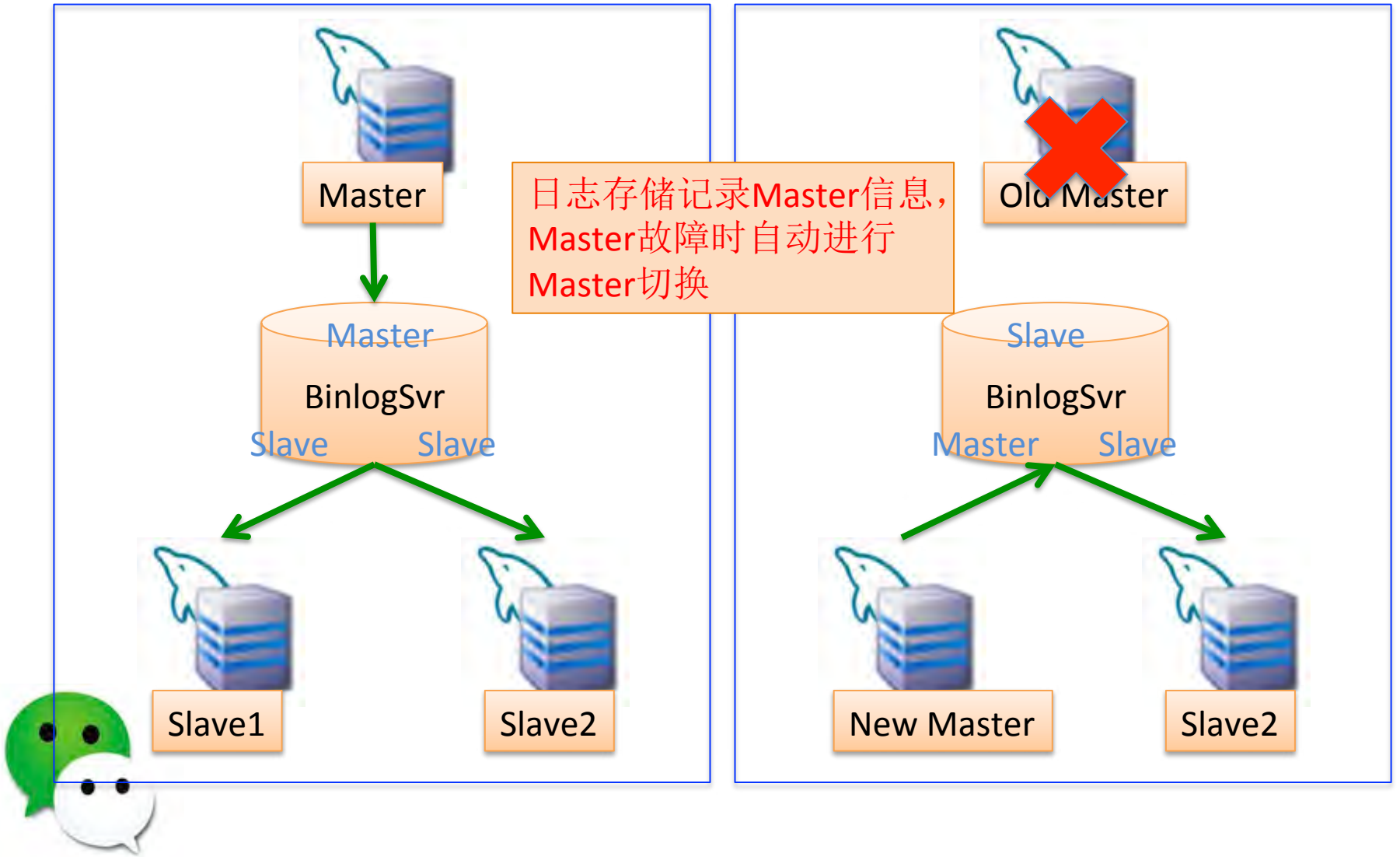

User1仍然访问旧Master
Read Id=201 Fail
User2
Old Master
Id=100
New Master
Id=100
Id=201
Slave2
Id=100
Id=201

用Proxy透传解决调用端通知问题

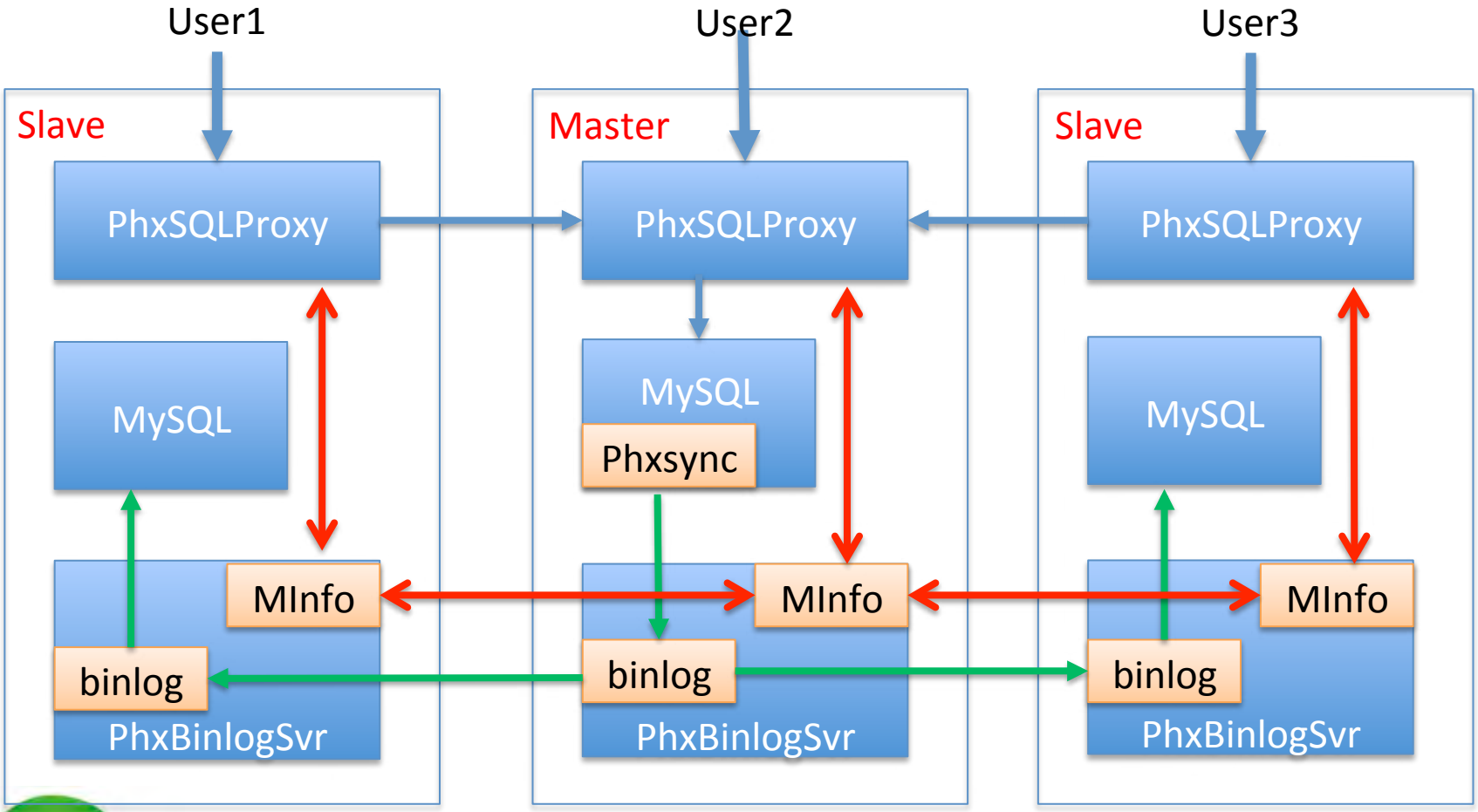




自动选主


Master
Old Master
日志存储记录Master信息，
Master故障时自动进行
Master切换
Master
BinlogSvr
Slave
Slave
Slave
BinlogSvr
Master
Slave
Slave1
Slave2
New Master
Slave2

PhxSQL系统架构图



PhxSQL实现

关键算法和流程

PhxSQL实现

- Proxy的功能和实现要点
- Phxsync的功能和实现要点
- Binlogsvr的功能和实现要点

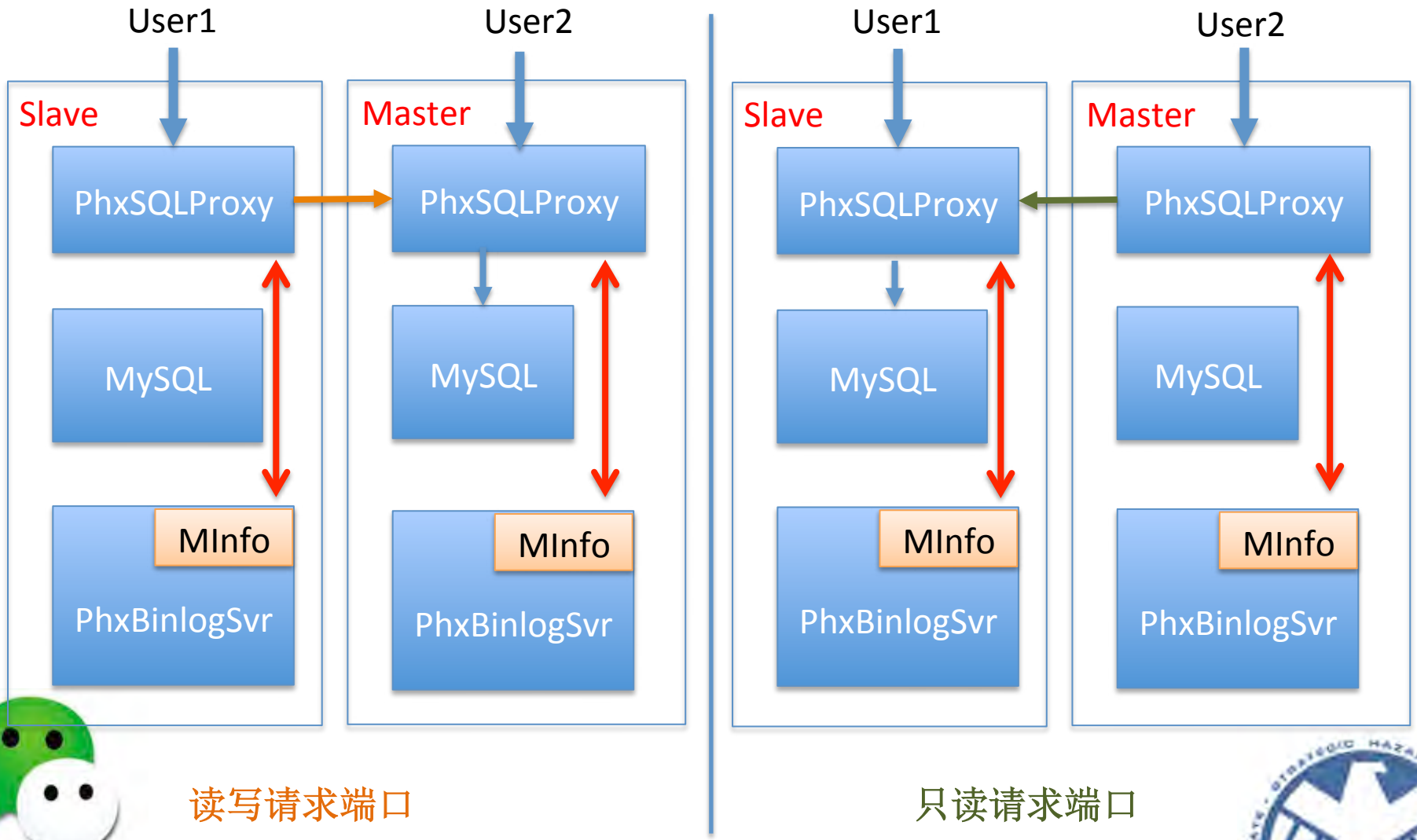
Proxy要具备的功能

- 请求透传
  - 读写端口请求透传
    - 非Master节点收到的请求透传给Master节点
    - Master节点收到的请求直接透传给MySQL
  - 只读端口请求透传
    - Master节点收到的请求透传给非Master节点
    - 非Master节点收到的请求直接透传给MySQL

Proxy实现要点

- 请求透传
  - 读写端口请求透传
  - 只读端口请求透传
- 高性能
  - 整个集群的读写性能与单机MySQL相当
- 完全兼容MySQL
  - 调用端不需要修改
  - 使用1:1模型兼容MySQL事务管理
  - 透传源IP兼容IP权限设置

请求透传



读写请求端口

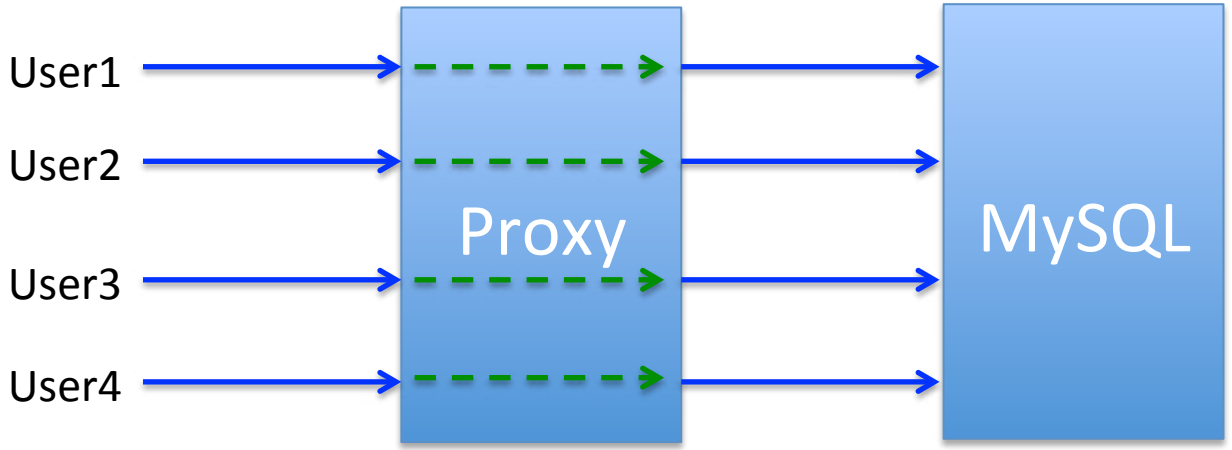
只读请求端口

用libco保证Proxy高性能

- Libco是微信团队开源的一个高性能协程库
  - 用同步方式写代码，实现异步代码的性能
  - 支持千万级的并发连接
- 项目地址
  - https://github.com/tencent-wechat/libco

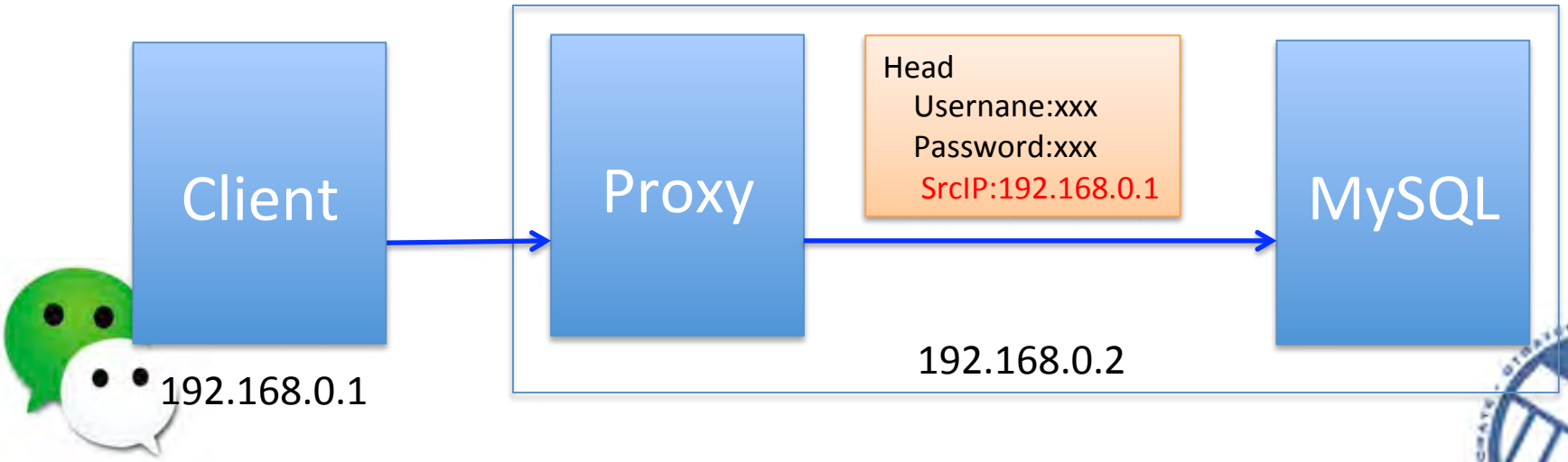
Proxy使用1:1模型兼容MySQL事务管理

- MySQL事务管理基于连接
  - 同一个事务的所有请求通过同一个连接通信
  - 在事务处理中途连接丢失，事务将被rollback
    - http://dev.mysql.com/doc/refman/5.6/en/innodb-autocommit-commit-rollback.html
- Proxy使用1:1连接模型能实现完全兼容



Proxy透传源IP兼容权限设置

- MySQL需要源IP
  - 权限管理基于（用户，源IP）对
  - MySQL通过socket句柄获取源IP
- Proxy透传IP给MySQL
  - 利用MySQL协议HEAD保留字段



Phxsync要具备的功能

- 重启时校准本地binlog
- 正常运行时提交binlog到BinlogSvr

Phxsync插件实现要点

- 基于插件形式，便于更新MySQL官方版本，减少后续维护成本
  - 抽象before_binlog_init新接口用于校准binlog
  - 使用原有的after_flush接口提交binlog
  - 提交补丁给MySQL官方
    - http://bugs.mysql.com/bug.php?id=83158

BinlogSvr要具备的功能

- 可靠日志存储
  - 接收MySQL Master推送的binlog并同步到其他节点
  - MySQL Slave利用原生复制协议拉取binlog
  - 保证各节点的数据一致性
- 维护Master信息
  - 监控PhxSQL节点的状态
  - 自动切换Master
  - 保证各节点的数据一致性

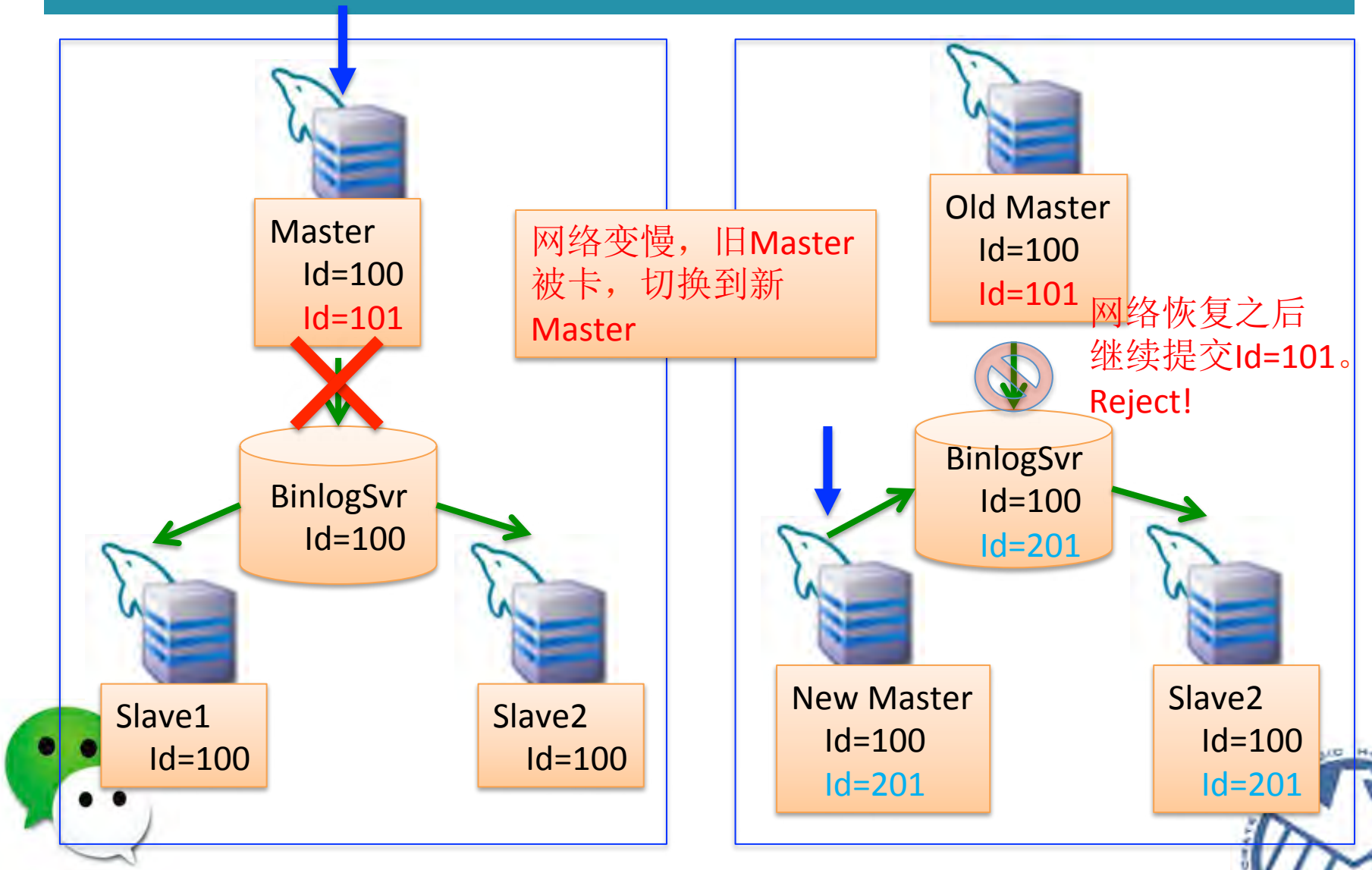
BinlogSvr实现要点

- Paxos协议
  - 保证各节点的数据一致
  - 保证集群机器超过一半存活还能服务
- Binlog存储
  - 通过校验IP防止非Master提交
  - 通过乐观锁防止拜占庭错误
- 兼容MySQL Slave复制协议
  - BinlogSvr支持MySQL的原生复制协议
- Master管理
  - 选举和续期

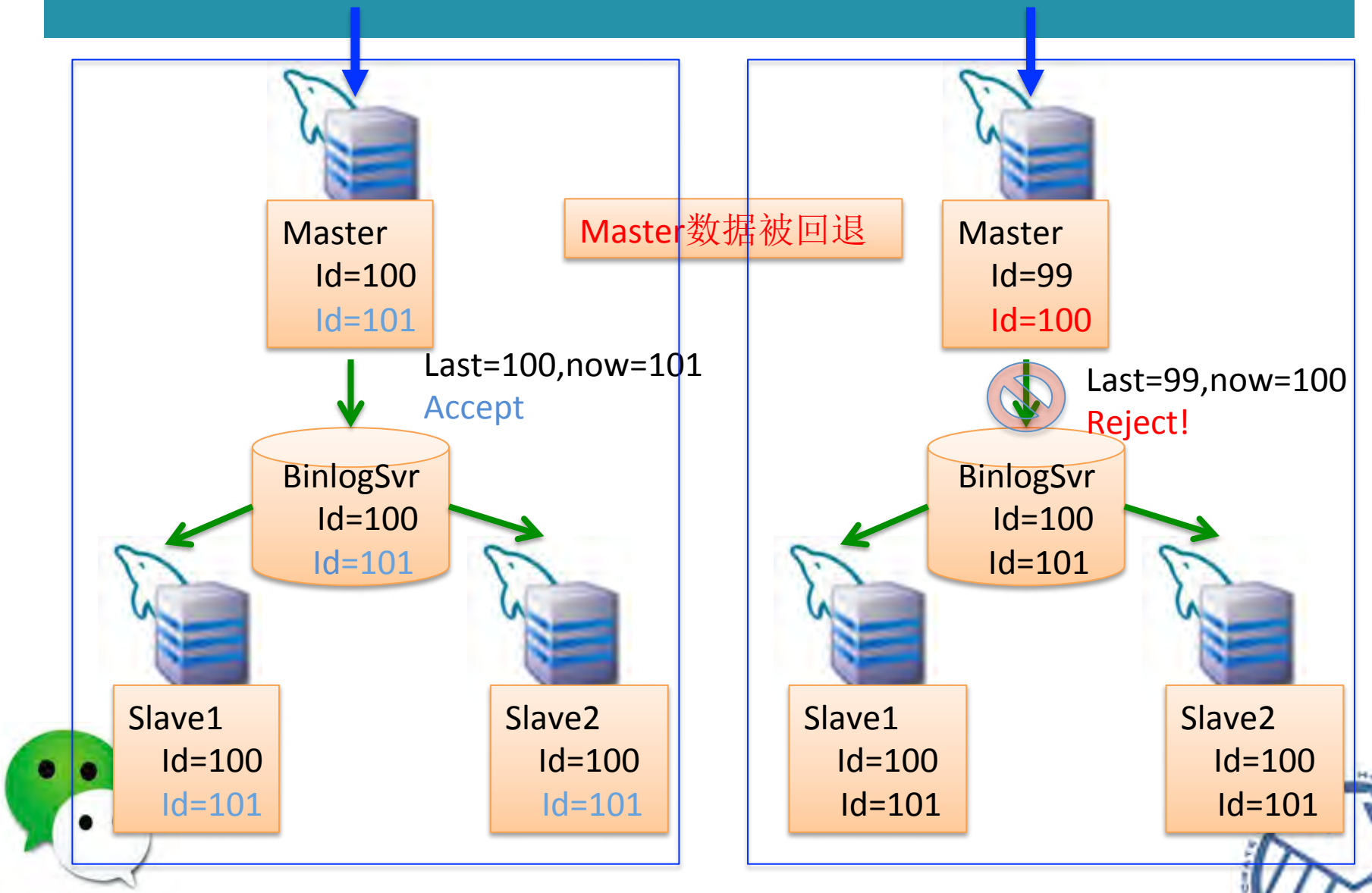
使用PhxPaxos库

- PhxPaxos是微信团队开源的Paxos类库
  - 高性能
  - 功能完善
  - 稳定性经过大规模验证
  - 接口方便易用
- 项目地址
  - https://github.com/tencent-wechat/phxpaxos

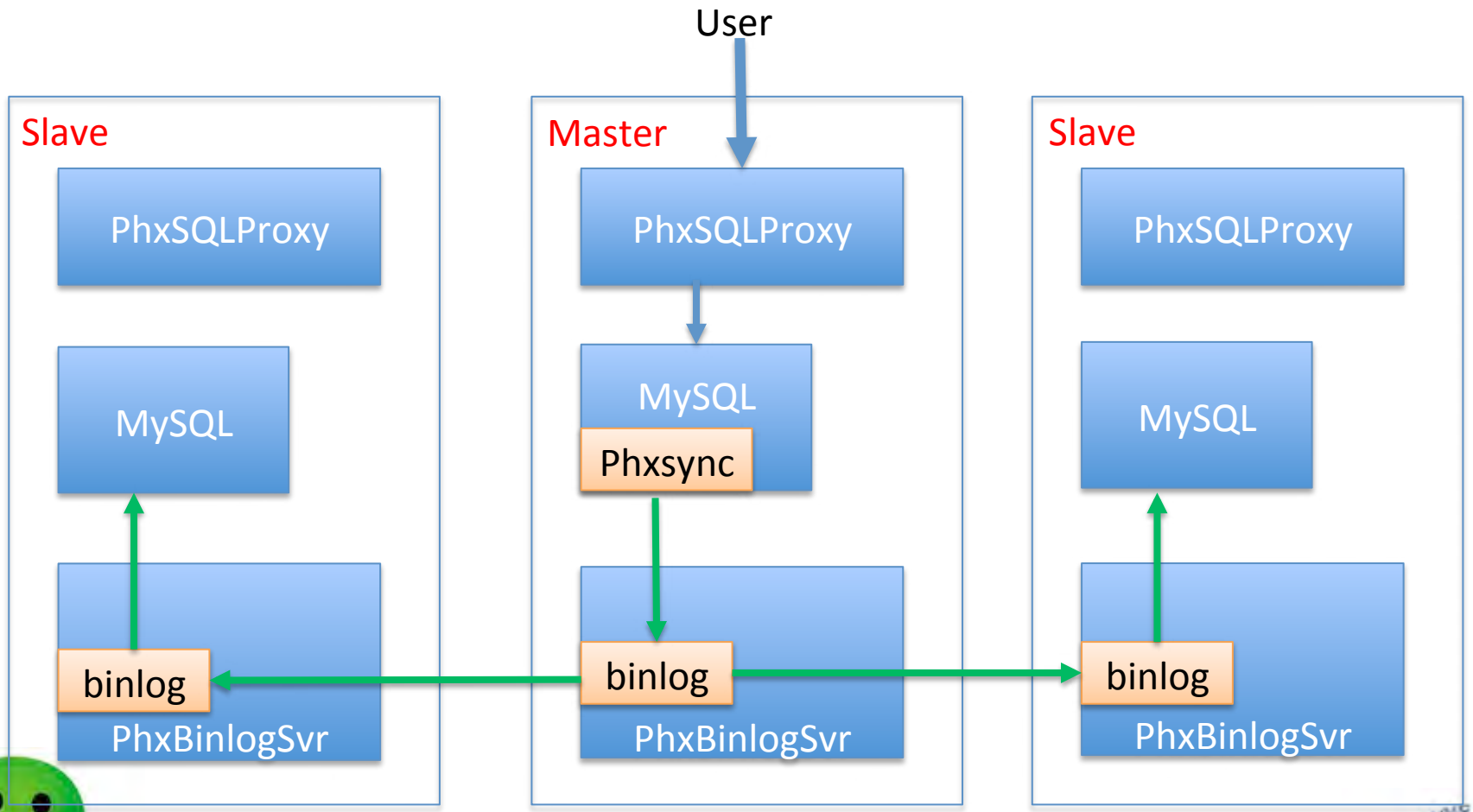
通过校验IP防止非法Master错误提交



用乐观锁防止拜占庭错误



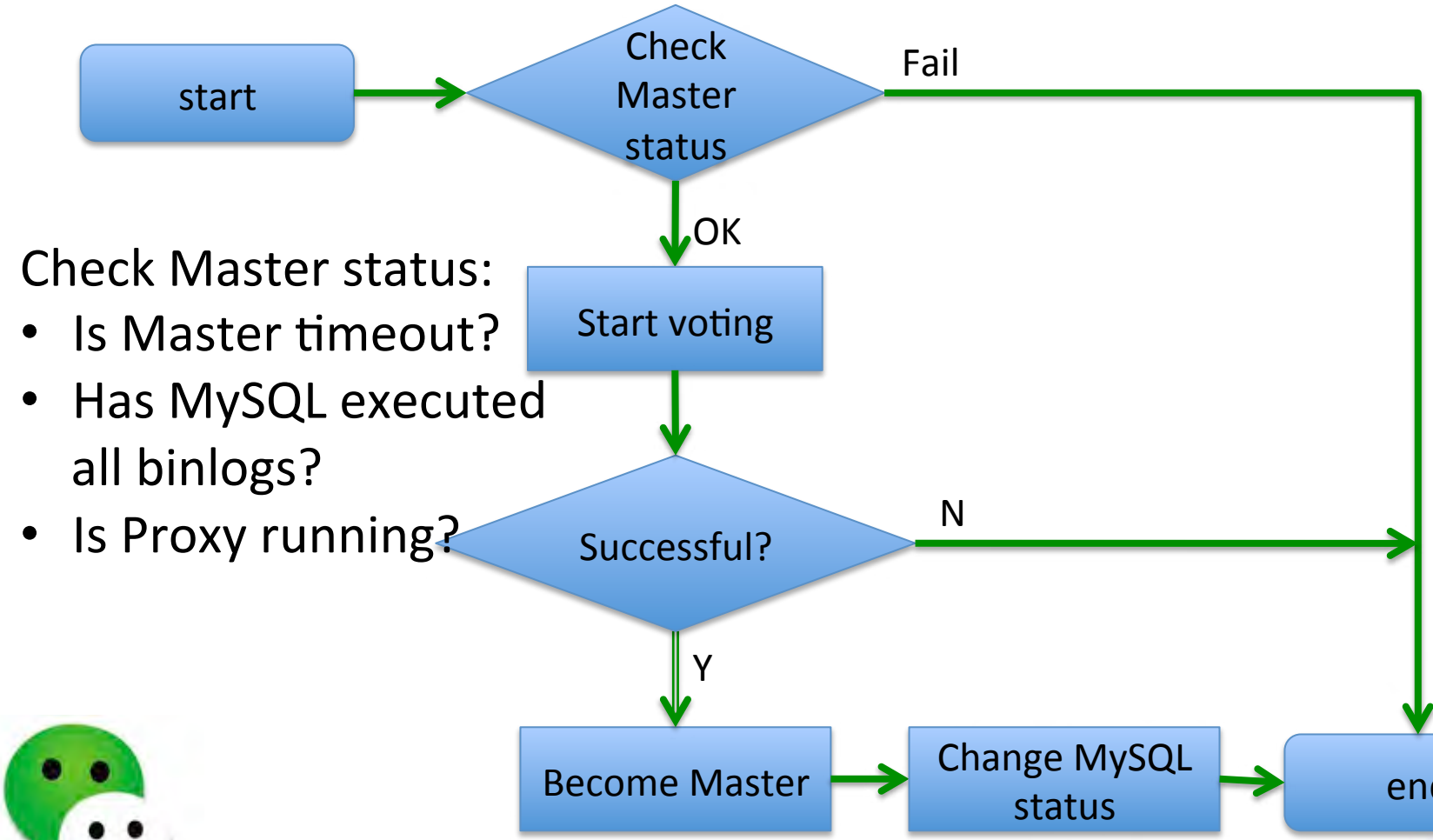
BinlogSvr支持MySQL的异步复制协议



用PhxPaxos实现Master管理

- Master管理
  - Mysql已执行完所有binlog
  - 通过paxos协议选举新master
  - Master定期续期
  - 旧master的租约过期后，重新选举新的master
  - 监控mysql的状态

Master自动切换



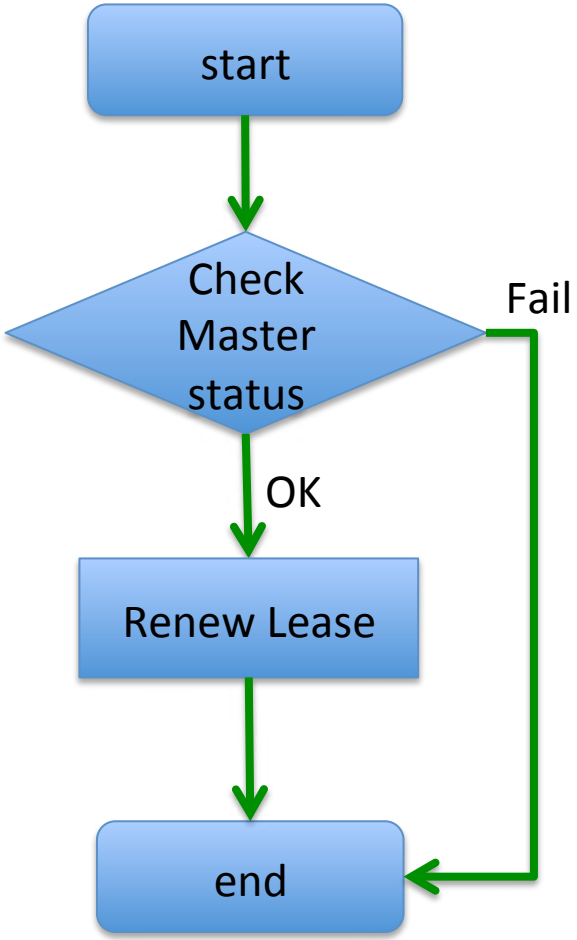
Check Master status:

- Is Master timeout?
- Has MySQL executed all binlogs?
- Is Proxy running?

Master定期续租

Check Master status:

- Is MySQL healthy?
- Is Proxy running?



PhxSQL效果

容灾演习和性能

PhxSQL保证主备数据一致

通过比较PhxSQL集群中各节点的数据（MySQL binlog, PhxPaxos, Binlogsvr) 得知各节点数据一致。

MySQL三机binlog数据的md5sum

```
allocuin1[qq]:~/phxsql/tools>
5sum
791d144d728a551a12a2b901e77c0594  -
allocuin2[qq]:~/phxsql/tools>
5sum
791d144d728a551a12a2b901e77c0594  -
allocuin3[qq]:~/phxsql/tools>
5sum
791d144d728a551a12a2b901e77c0594  -
```

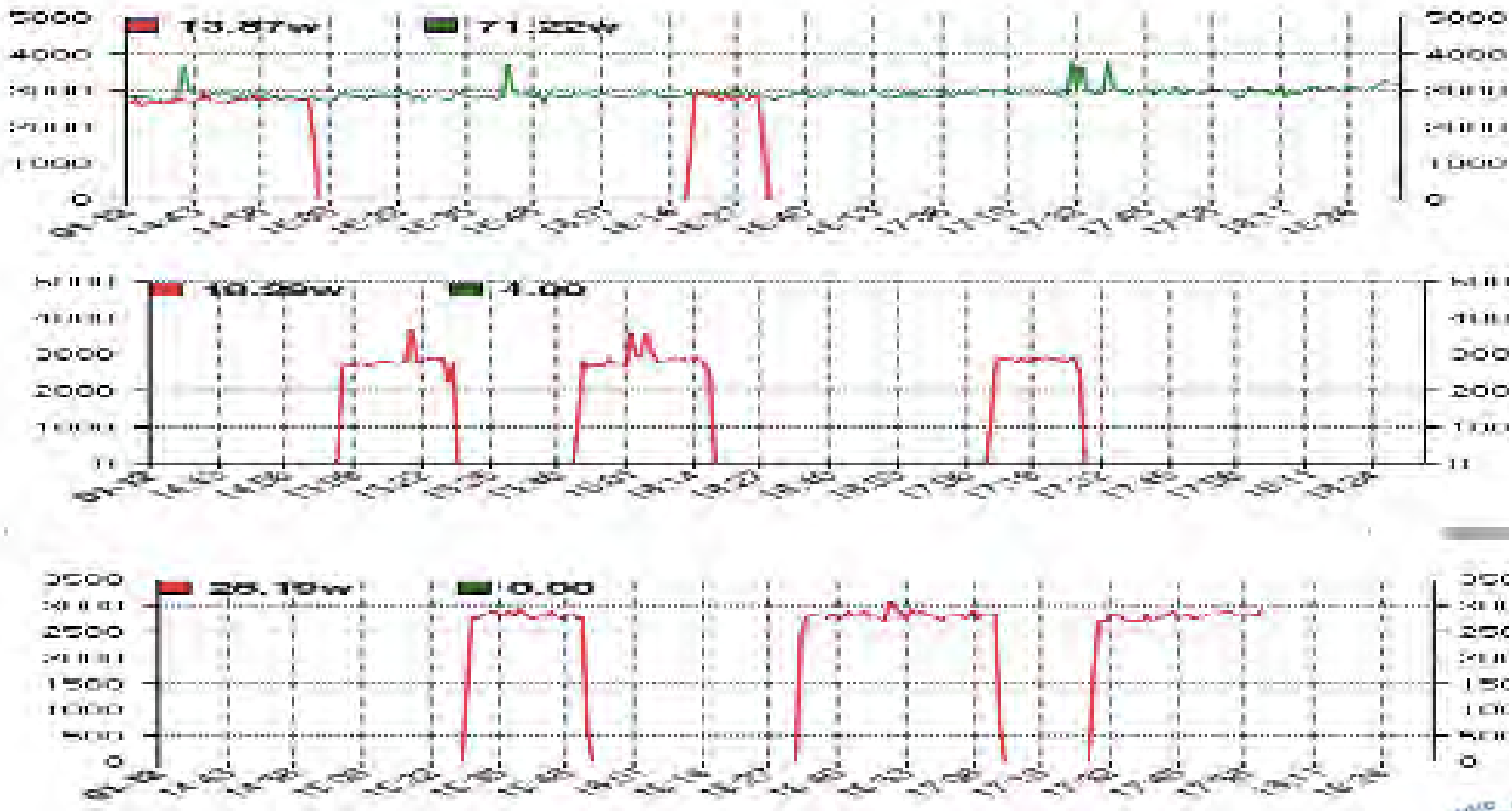
PhxPaxos三机数据的checksum

```
allocuin1[qq]:~/phxsql/tools>
<2> <phxbinlogsvr(14429,14438)> 18:18:05 004
m 246356992 new checksum 3784655537
allocuin2[qq]:~/phxsql/tools>
<2> <phxbinlogsvr(1214,2916)> 18:18:04 983
246356992 new checksum 3784655537
allocuin3[qq]:~/phxsql/tools>
<2> <phxbinlogsvr(2004,2013)> 18:18:05 002
246356992 new checksum 3784655537
```

Binlogsvr三机数据的checksum

```
allocuin1[qq]:~/phxsql/tools>
<2> <phxbinlogsvr(14429,14438)> 18:08:59 852
38681247771 new checksum 38513051386812477771,
allocuin2[qq]:~/phxsql/tools>
<2> <phxbinlogsvr(1214,2916)> 18:08:59 847
681247771 new checksum 38513051386812477771,
allocuin3[qq]:~/phxsql/tools>
<2> <phxbinlogsvr(2004,2013)> 18:08:59 854
681247771 new checksum 38513051386812477771,
```

PhxSQL自动切换Master



PhxSQL自动切换Master

10 次测试结果平均值为：

```
( 34 + 27 + 26 + 27 + 25 + 26 + 26 + 38 + 28 + 38 ) / 10 = 29.5 s
```

已知目前租约时间设置为：

```
1   MasterLease = 20   // 20 s
```

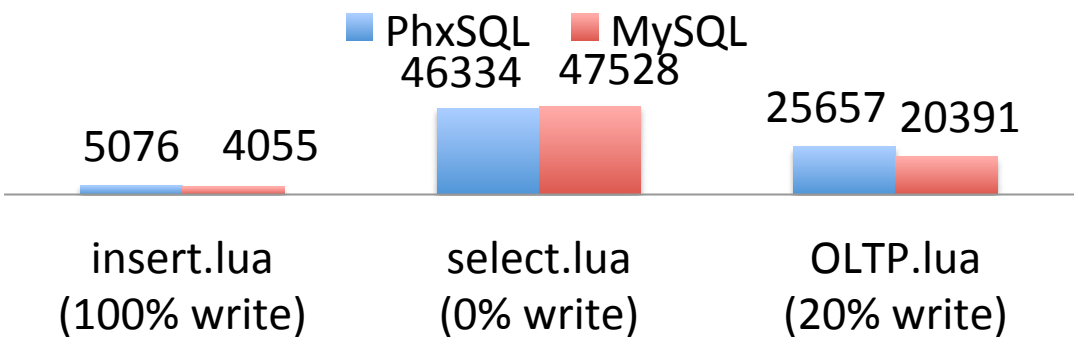
数据来源：http://chuzhiyan.com/2016/10/11/PhxSQL-FAQ

PhxSQL性能

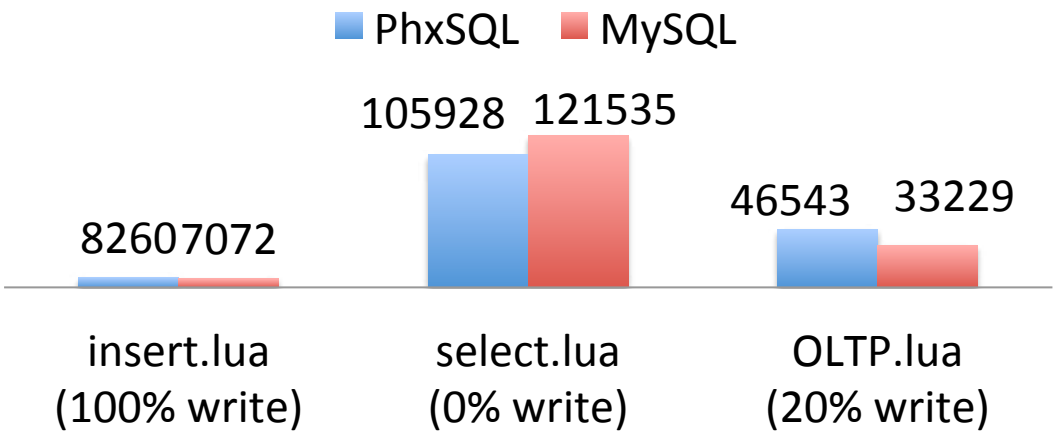
- MySQL版本
  - Percona 5.6.31-77.0
- 机器信息
  - CPU : Intel(R) Xeon(R) CPU E5-2420 0 @ 1.90GHz * 24
  - Memory : 32G
  - Disk : SSD Raid10
  - Ping Costs
    - Master -> Slave : 3 ~ 4ms
    - Client -> Master : 4ms
- 工具和参数
  - sysbench
  - --oltp-tables-count=10 --oltp-table-size=1000000 --num-threads=500
  - --max-requests=100000 --report-interval=1 --max-time=200

PhxSQL的QPS性能对比

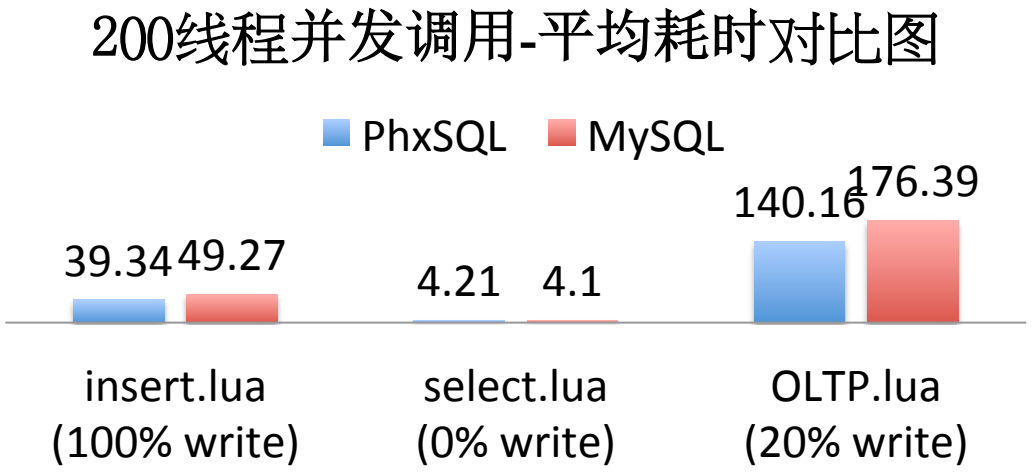
200线程并发调用-QPS对比图

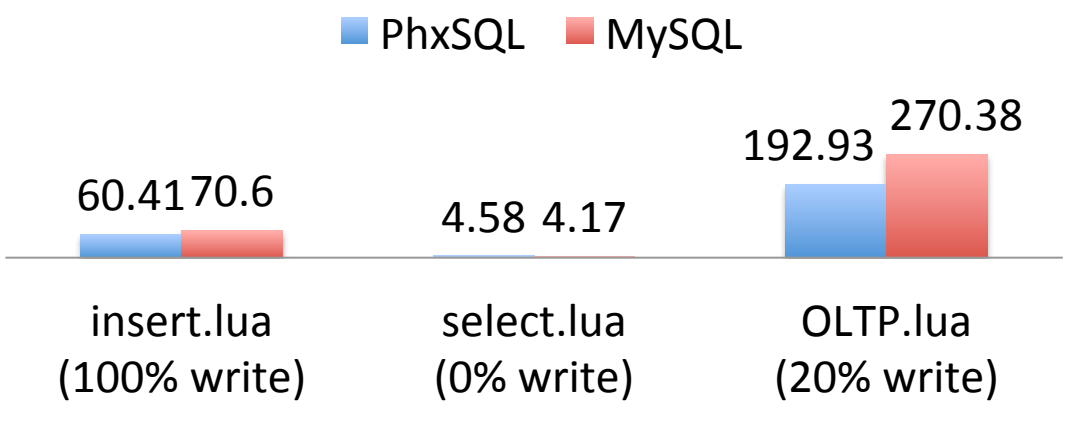


500线程并发调用-QPS对比图



PhxSQL的平均耗时性能对比





PhxSQL成功案例

- QQ邮箱(域名邮箱)域名记录服务器
  - 单个集群调用峰值40w/min
  - 写请求平均耗时在20ms以下
  - 读写比为20：1

- 机器配置
  - intel(r) xeon(r) cpu x3440 @ 2.53ghz 8 core， 8g ram

PhxSQL特点总结

- 同时满足数据强一致和服务高可用
- 性能优于半同步
- 完全兼容MySQL

PhxSQL下一步计划

- 适配MySQL 5.7

THANKS

Q&A

PhxSQL

PhxRPC

PhxPaxos

3个臭皮匠胜过一个猪脚亮
仅有3人的PhxSQL团队
努力在寻找命运中的第4人

我的微信号：junechen
联系我们：
phxteam@tencent.com

Github地址：
https://github.com/tencent-wechat/phxsql

附录

有可能被提问的

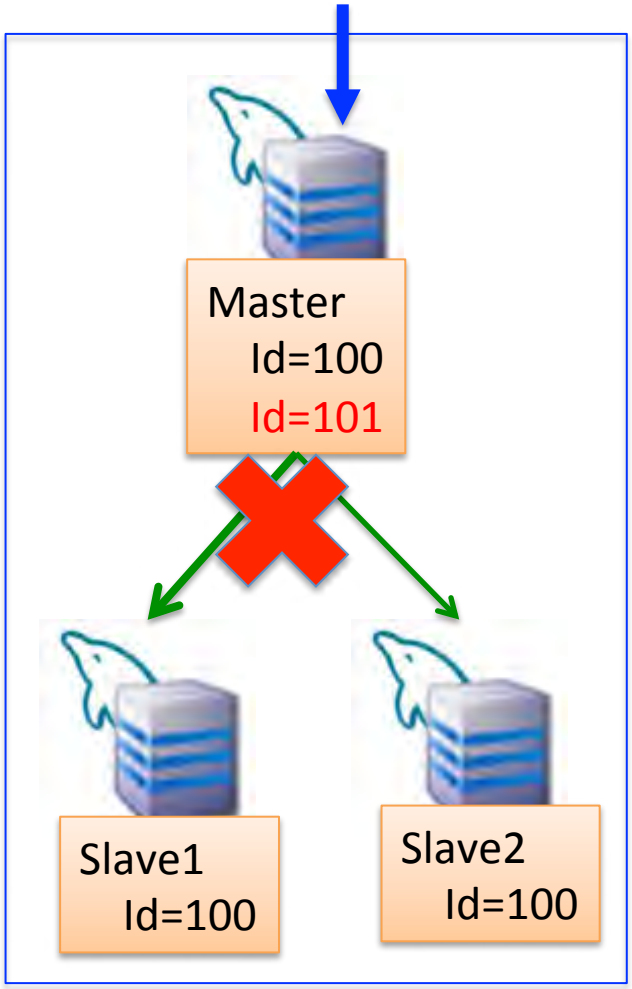
修正Master重启过程不能解决问题

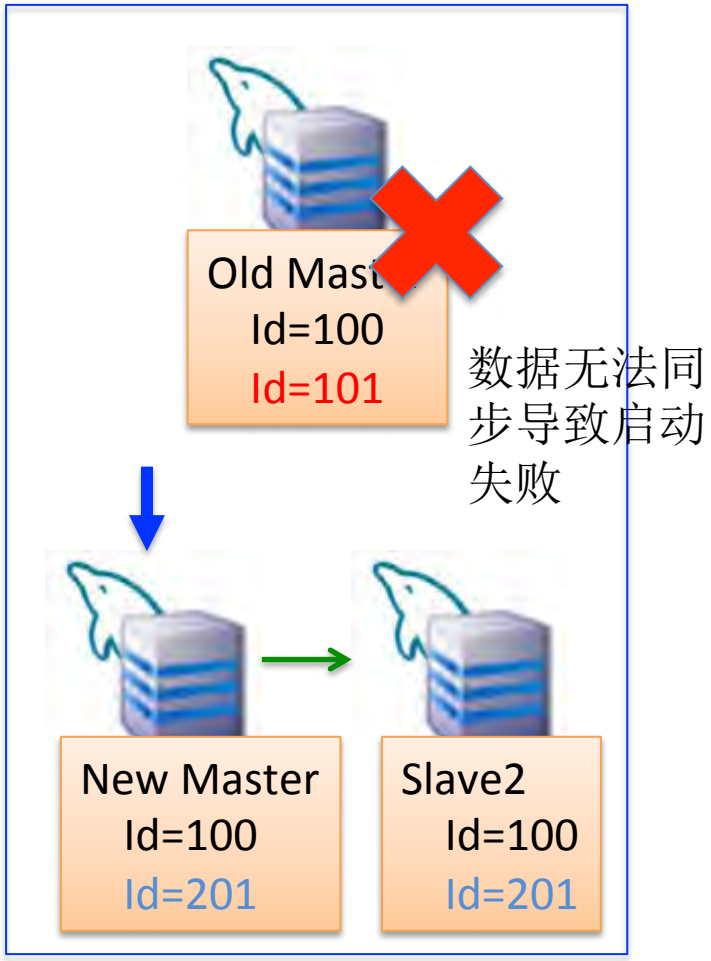
不能同时满足高可用和强一致

修正Master重启的方法

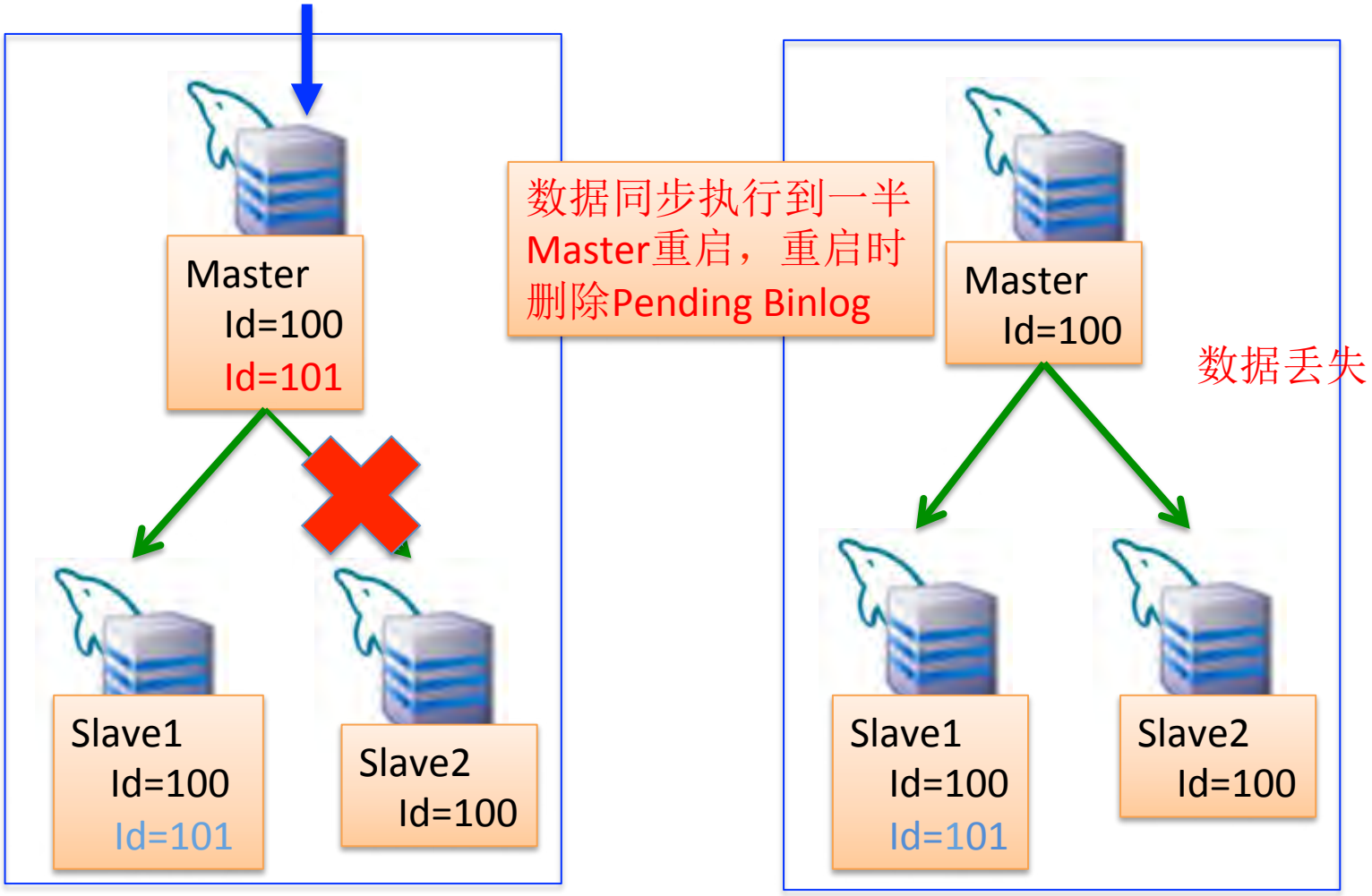
- 重启时执行半同步
- 重启时直接删除PendingBinlog
- 重启时直接提交PendingBinlog
- 重启时校准binlog

Master启动时执行半同步

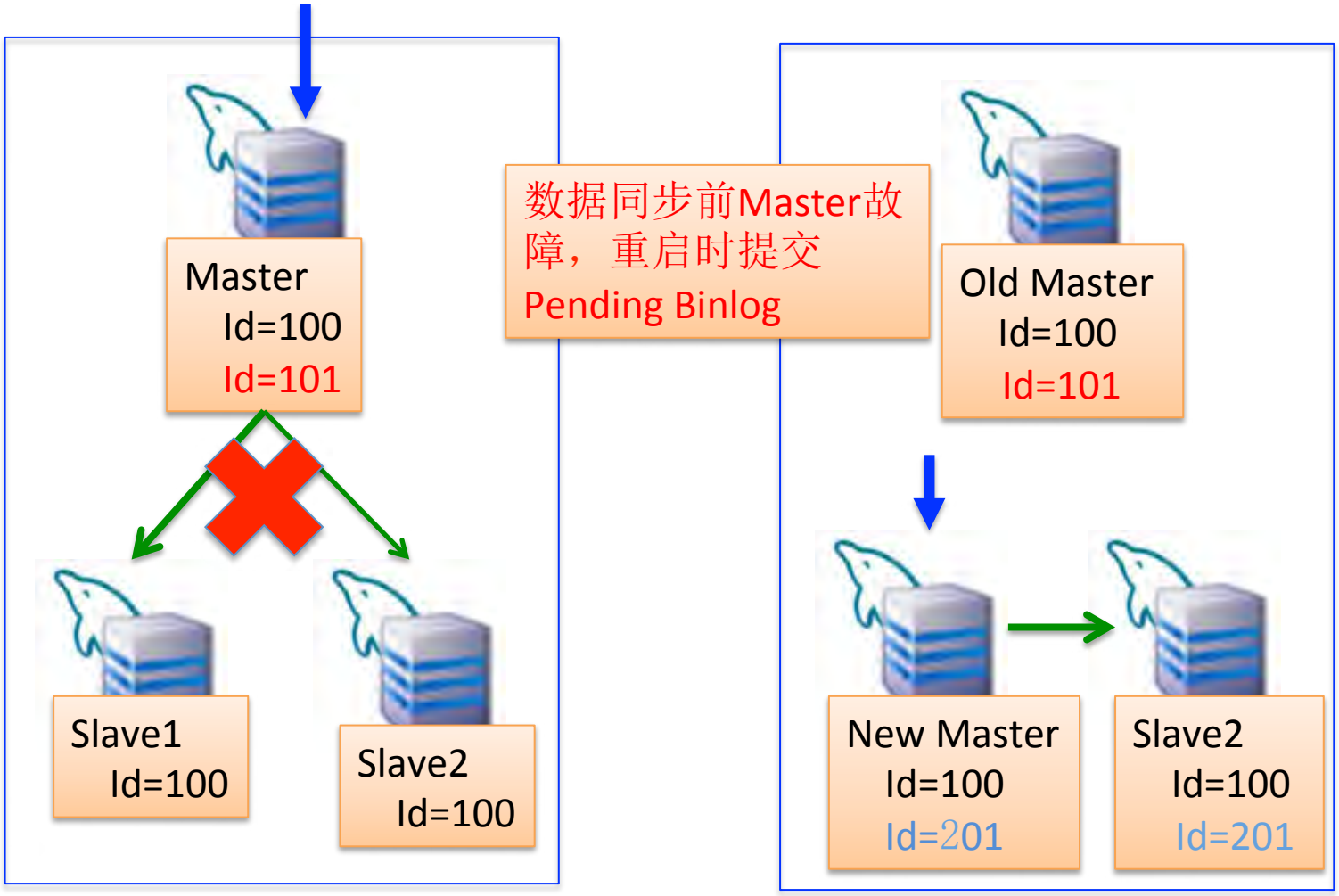




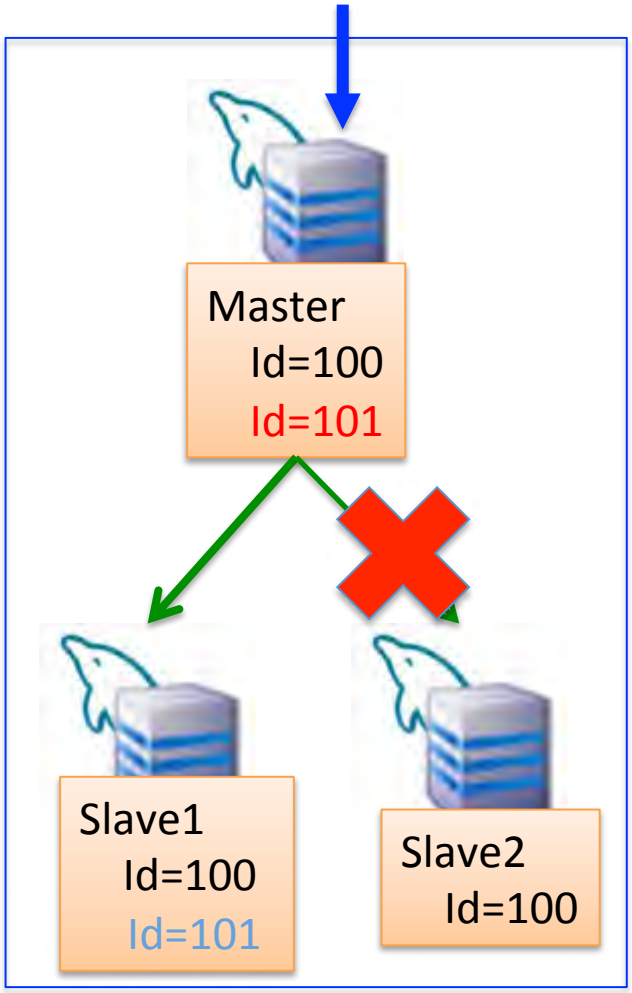
Master启动直接删除Pending Binlog

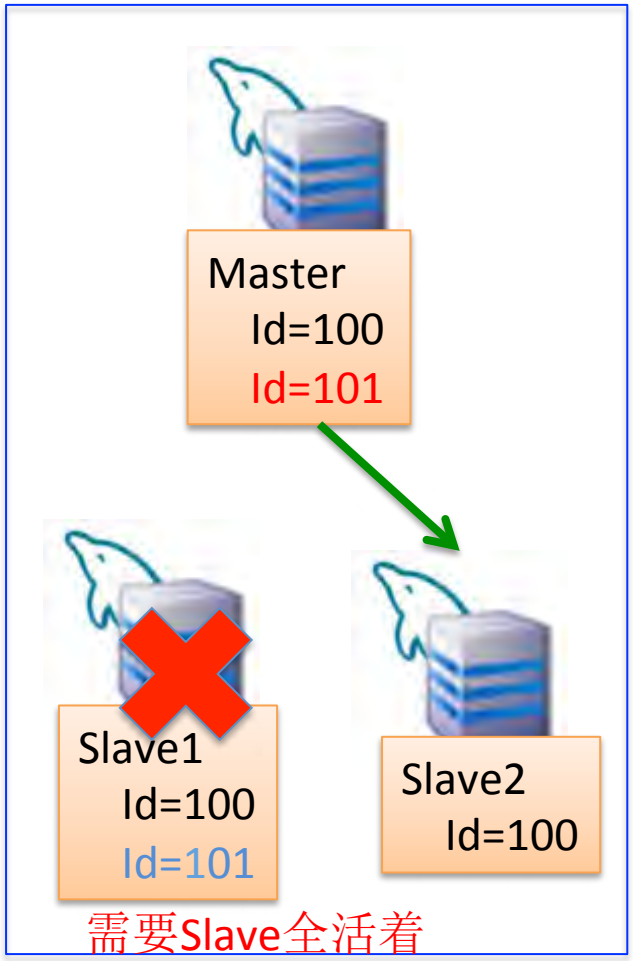


Master启动直接提交Pending Binlog

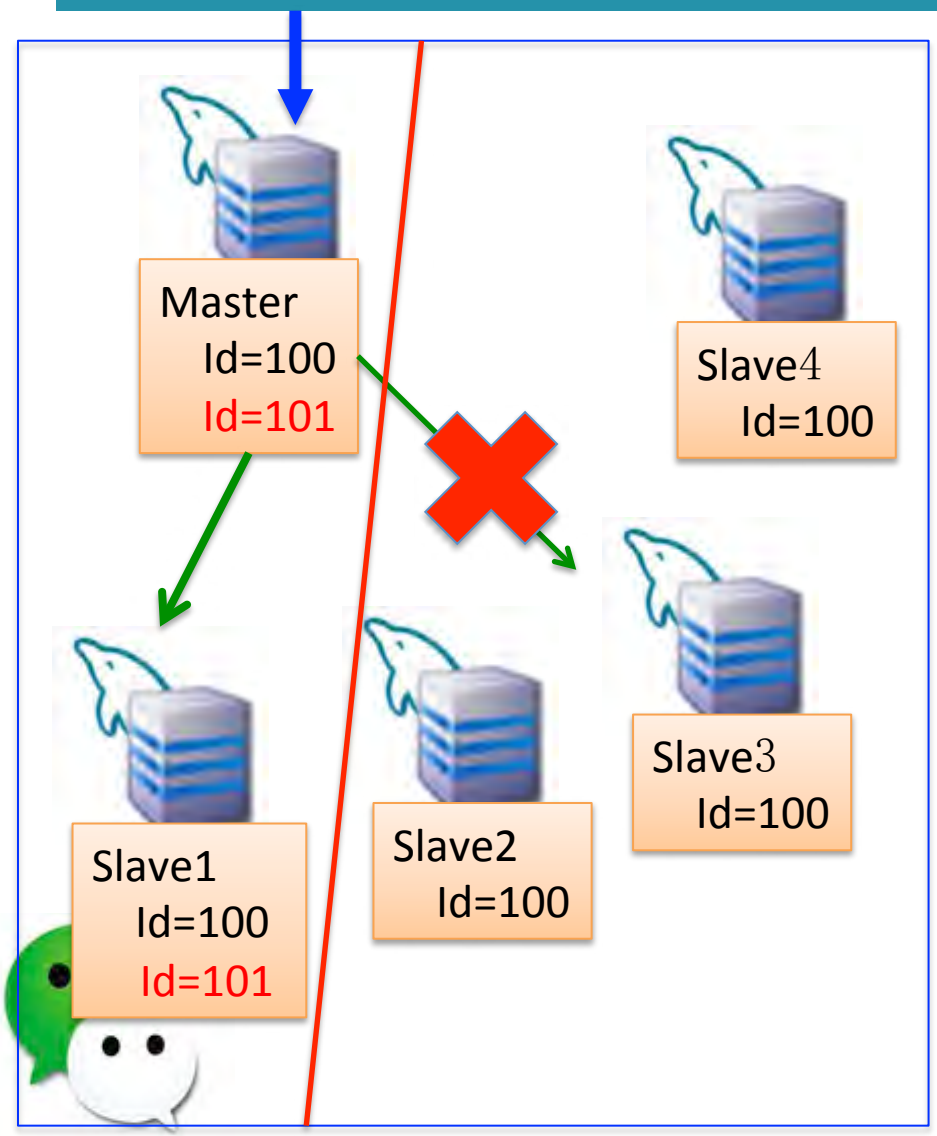


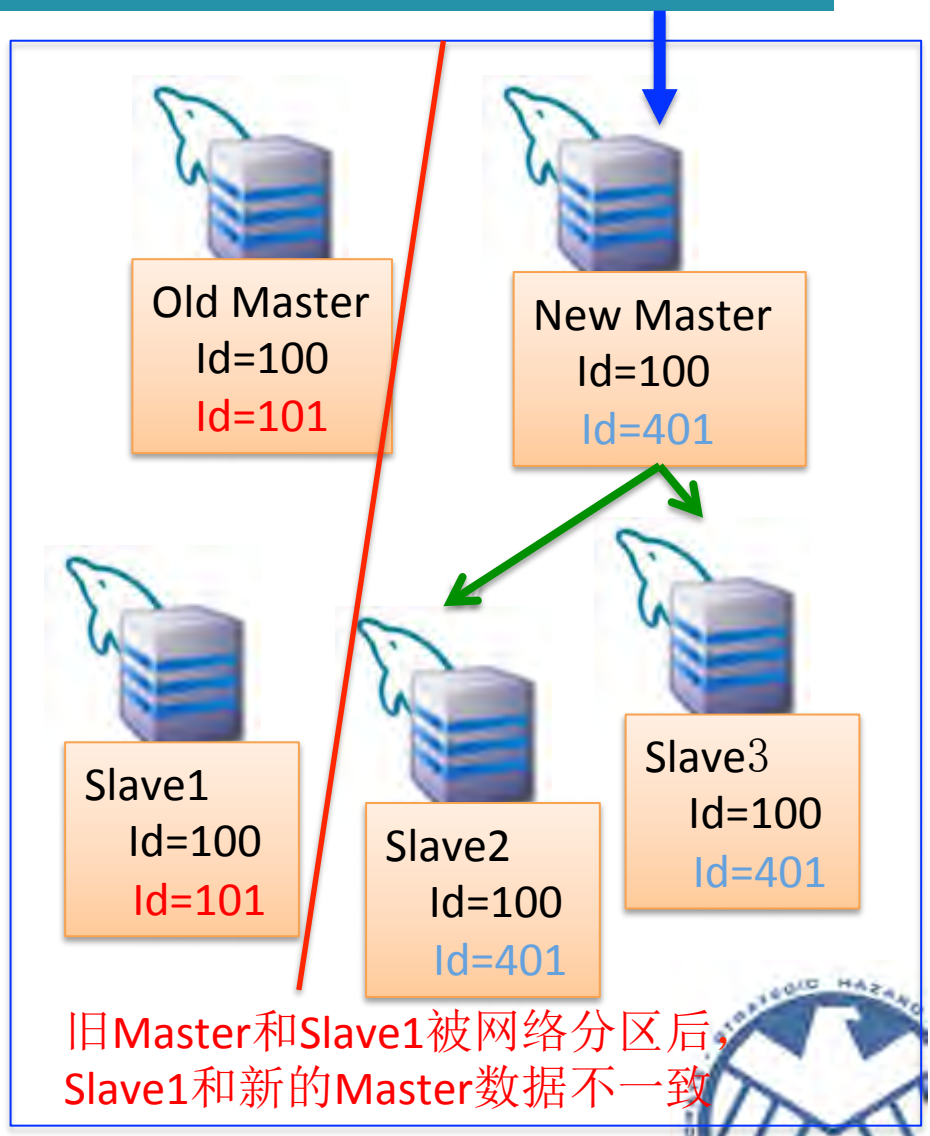
Master启动时校准Binlog





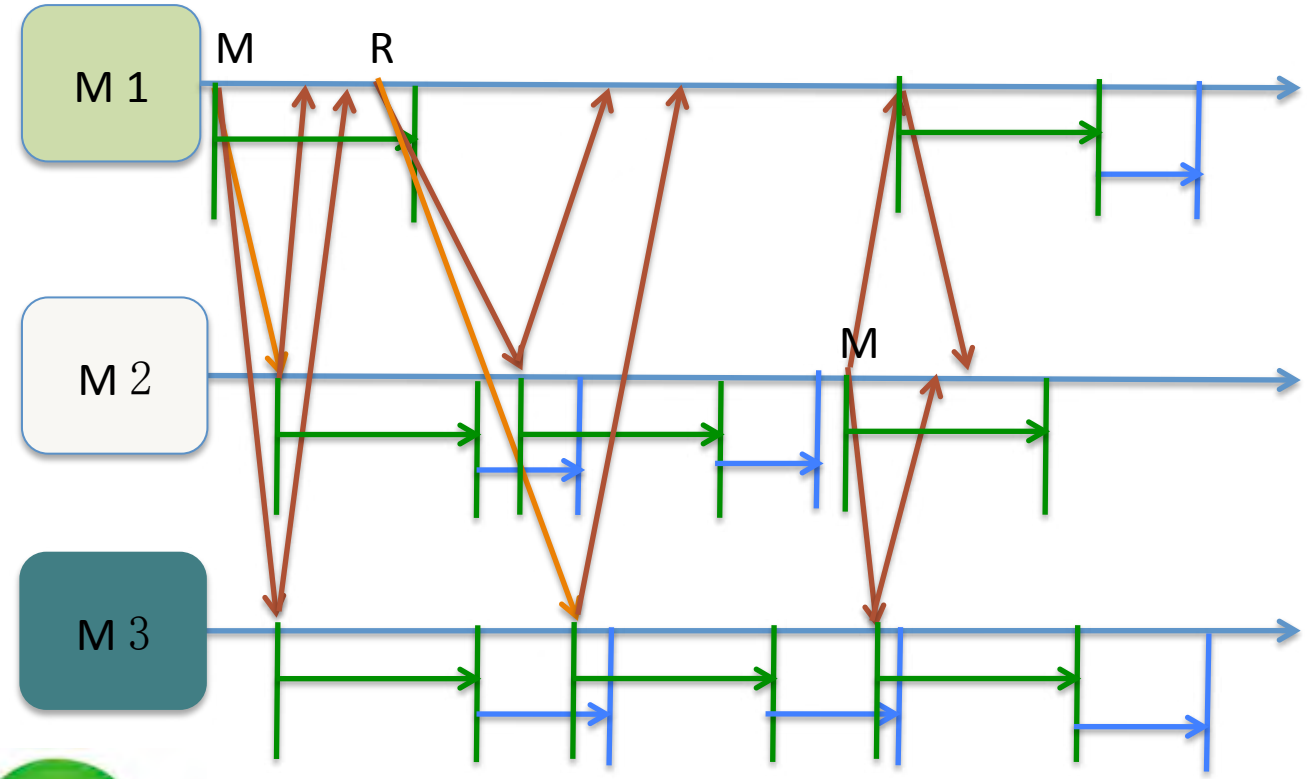
网络分区导致Slave和Master数据不一致





旧Master和Slave1被网络分区后，
Slave1和新的Master数据不一致

Master 租约设计



- Master租约表Master存活时间.
- Master 定期续约.
- Slave上加上额外的Master lease避免同时申请成为Master。

M: Vote Master R: Renew lease