腾讯金融级数据库TDSQL分析

关于本人

- 潘安群
- 腾讯 – TEG – 计费平台部
- 2007年加入腾讯， 10年以上的Linux后台Server开发经验，目前主要从事分布式Cache，实时大数据处理引擎，分布式MySQL(TDSQL)设计和开发工作

关于TDSQL

米大师
数据层解决方案

联机交易
数据层解决方案

金融云
金融数据库

目录

金融场景的数据库需求

一分不差的银行级业务

——***高一致性，零数据丢失***

7 * 24 小时的不间断服务

——***自动容灾，高可用***

百亿级的账户，订单数据

百亿级的日交易流水

十万级别每秒并发

毫秒级交易响应

——***易伸缩，高并发***

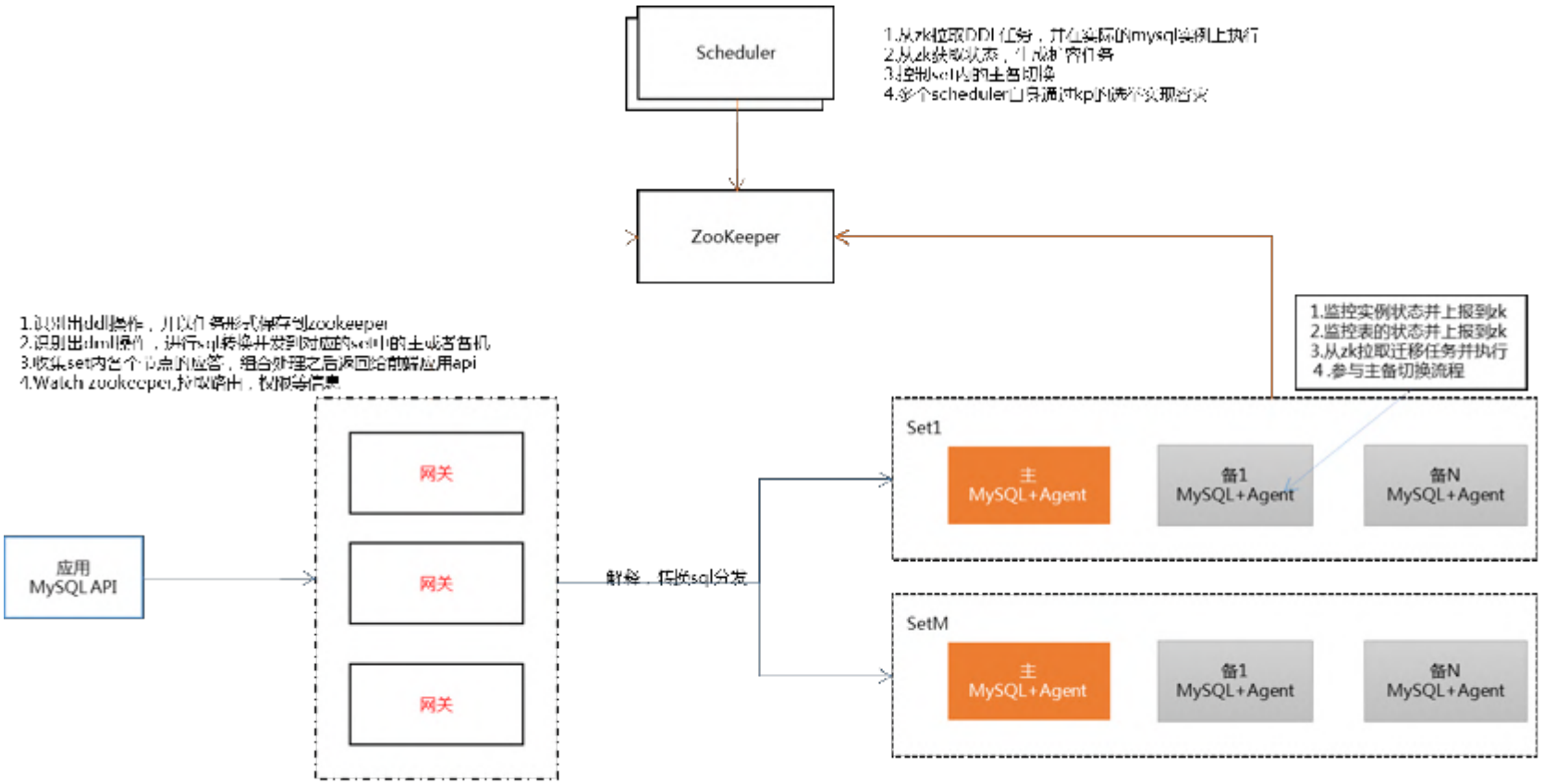
基于MySQL打造TDSQL的背景

- 存储平台的易用性（TBOSS平台1.0→4.0）
- SSD的大规模使用
- SQL的通用性 VS NoSQL的特例化
- MySQL 5.5 →5.6 →5.7 的大幅提升

基于MySQL打造TDSQL的背景

- 继续通过MySQL API和SQL接口访问集群
- 节点异常自动切换，切换过程保证数据零丢失，管好钱袋子
- 按需自动扩容/缩容，以支撑业务爆发式增长，扩容过程对业务基本上无感知
- 业务之间支持隔离，集群自身具备流控机制
- 配套工具完善：时耗分析、慢查询分析、冷备及恢复中心、异常SQL拦截、流控等

TDSQL整体架构


Scheduler
1.从zk拉取DDL任务，并在实际的mysql实例上执行
2.从zk获取状态，生成新的任务
3.控制set内的主备切换
4.多个scheduler之间通过tkp协调保证只有一个
ZooKeeper
1.监控实例状态并上报到zk
2.监控表的状态并上报到zk
3.从zk拉取迁移任务并执行
4.参与主备切换流程
1.识别出ddl操作，以任务形式保存到zookeeper
2.识别出dml操作，执行sql转换并发到对应的set中的主或者备机
3.收集set内各个节点的应答，组合处理之后返回给前端应用api
4.Watch zookeeper,获取事件，获取事件信息
应用
MySQL API
网关
网关
网关
解析，按照sql分发
Set1
主
MySQL+Agent
备1
MySQL+Agent
备N
MySQL+Agent
SetM
主
MySQL+Agent
备1
MySQL+Agent
备N
MySQL+Agent

TDSQL的容灾机制

- 目标
- 问题
- 存活监控
- 同步与性能
- 切换流程
- 跨城容灾

容灾机制

- 自动切换
- 自动恢复
- 主备一致性
- 跨IDC容灾

容灾决策

谁要切换？

- 如何发现故障？
- 如何确定是否需要切换？

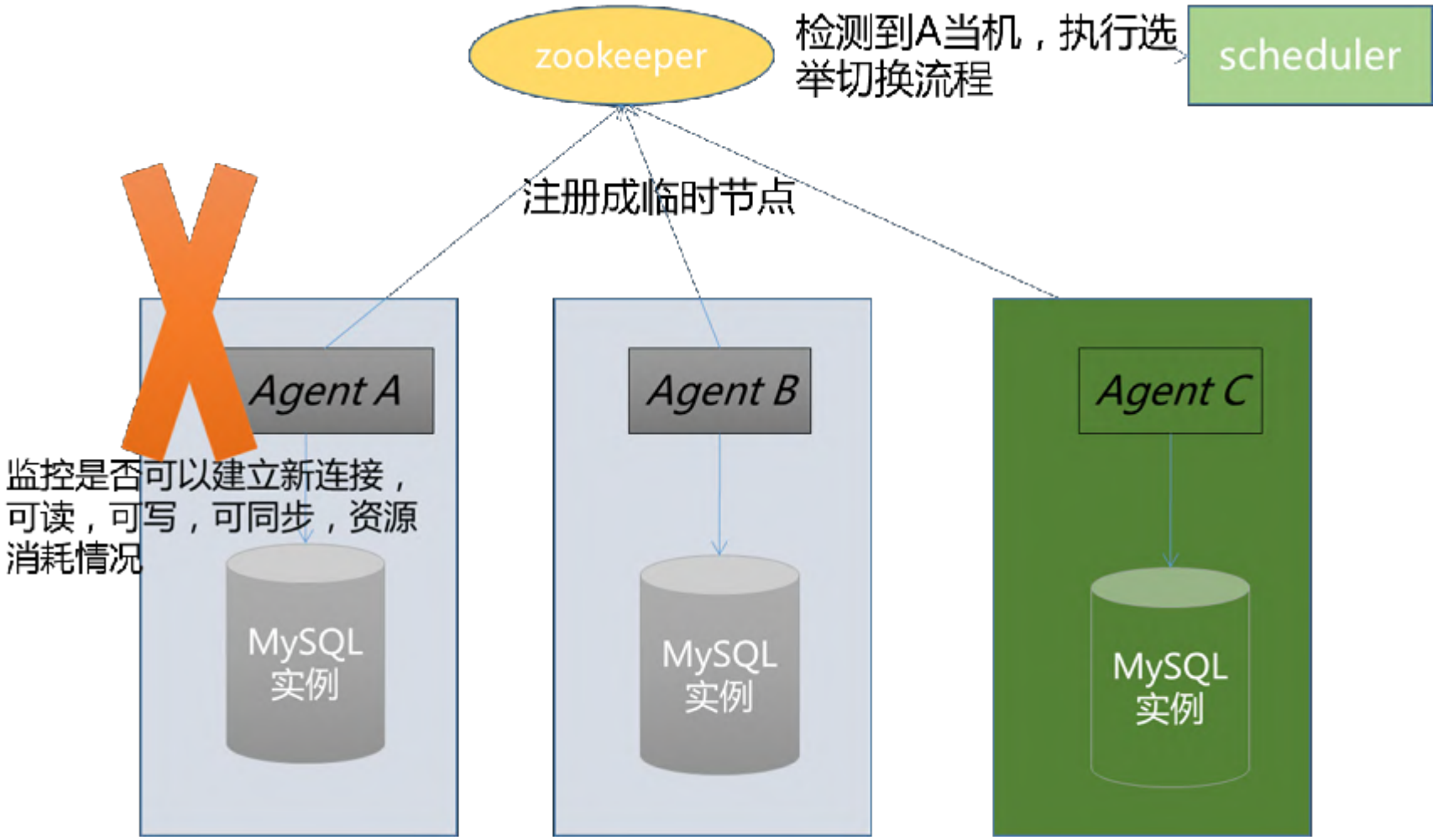
怎么切换？

- 如何保障数据一致性？
- 如何切请求？

如何恢复？

- 如何重建SET？

节点存活监控

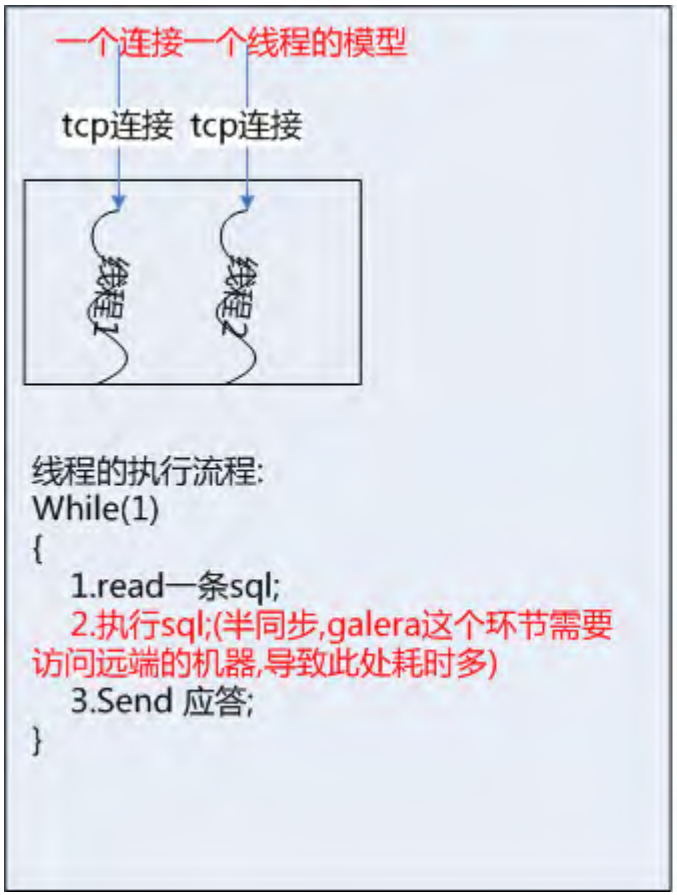

zookeeper
检测到A当机，执行选举切换流程
scheduler
注册成临时节点
Agent A
Agent B
Agent C
监控是否可以建立新连接，可读，可写，可同步，资源消耗情况
MySQL 实例
MySQL 实例
MySQL 实例

关于一致性与零丢失（1）

- MySQL异步复制
- MySQL半同步复制
  - 退化异步？
  - 一主多备？
  - 跨IDC性能？

主备复制方案（跨IDC）	TPS	时耗(ms)
异步	20,000	<10
半同步	2,200	4~600
网易innosql（半同步）	4,500	4~500
MariaDB Galera Cluster	6,000	4~10000

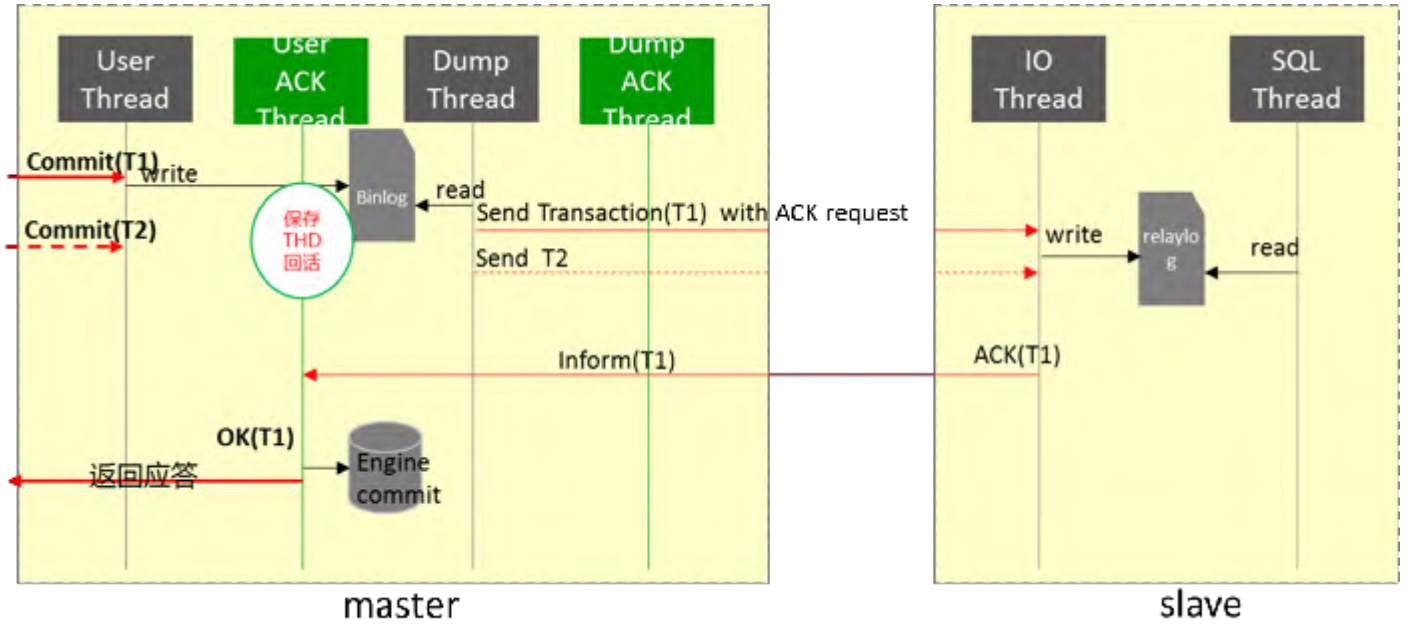
半同步的性能问题



线程的执行流程:
While(1)
{
1.read一条sql;
2.执行sql;(半同步,galera这个环节需要访问远端的机器,导致此处耗时多)
3.Send 应答;
}

线程的执行流程:
While(1)
{
1.取一条通过epoll检测到有可读事件的连接
//线程与连接不再一一对应,每次工作都要取一条可用连接
2.read一条sql;
3.执行sql;(半同步,galera这个环节需要访问远端的机器,导致此处耗时多)
4.Send 应答,绑定到epoll中
}

关于一致性与零丢失（2）



主备复制方案（跨IDC）	TPS	时耗(ms)
异步	20,000	<10
半同步	2,200	4~600ms
异步化改造后的半同步	9,500	99.9%的<30ms 少量毛刺，最大达到600
网易innosql（半同步）	4,500	4~500ms
MariaDB Galera Cluster	6,000	4~10000ms

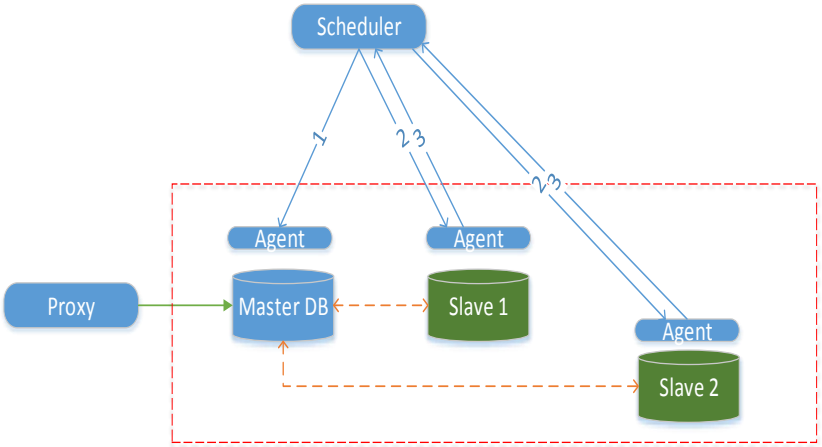
容灾切换流程（切换）

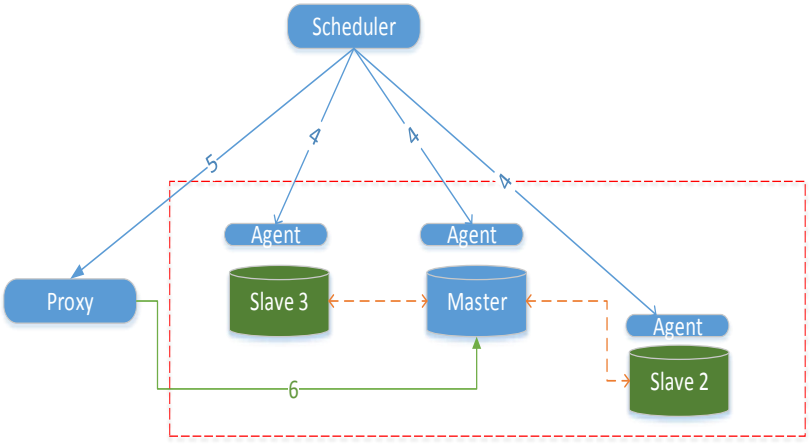
原则：

1、主机可读可写，备机只读，备机可以开放给业务查询使用

2、任何时刻同一个SET不能有两个主机

3、 宁愿拒绝服务，不提供错误的服务，追求CAP中的C，必要的时候牺牲部分A





1、主DB降级为备机（杀死当前所有session,设置只读，如因为当机原因没有收到下次重新启动也会执行这个流程），同时会给网关下发暂时没有主节点的路由

2、参与选举的备机停止io线程之后上报最新的binlog点

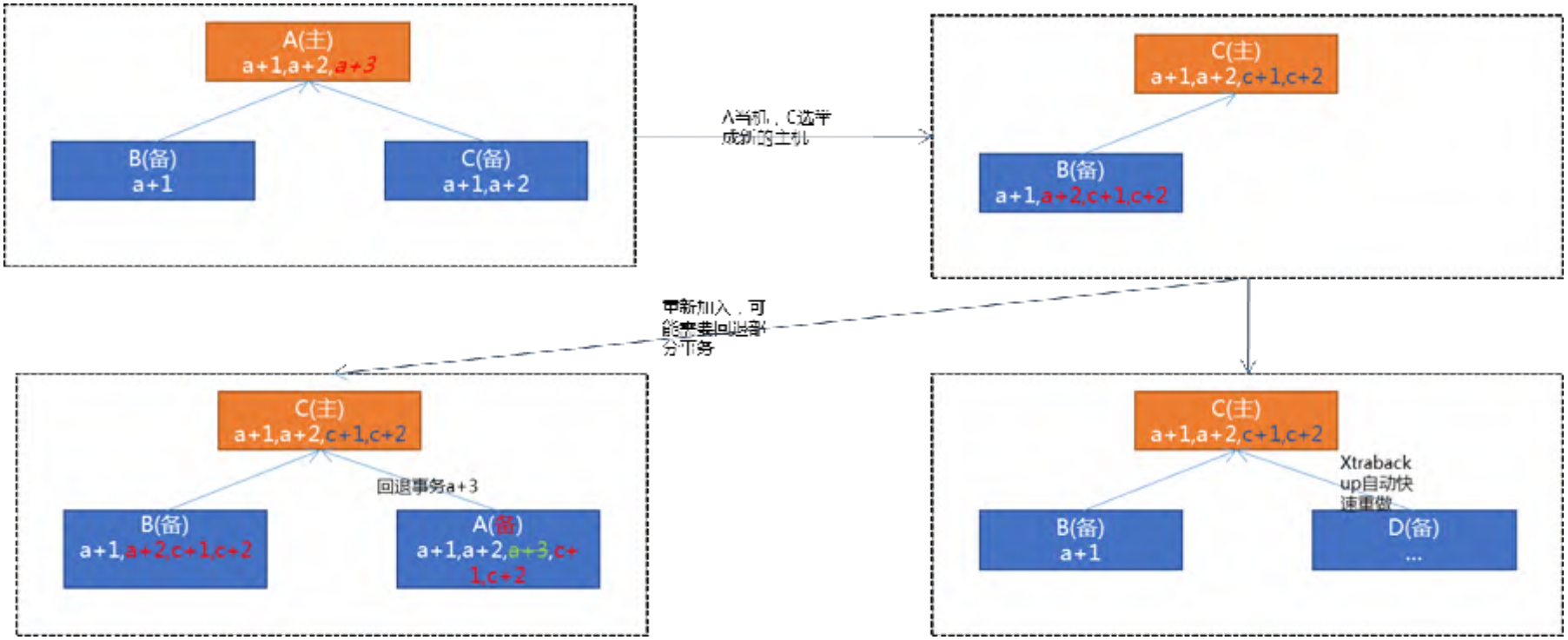
3、scheduler收到binlog点之后，选择出binlog最大的节点(可能同时有 2 个)并要求对应的机器加载完relay log。当收到加载完relay log信息之后，则选择这个应答的节点为主机；

4、重建主备关系

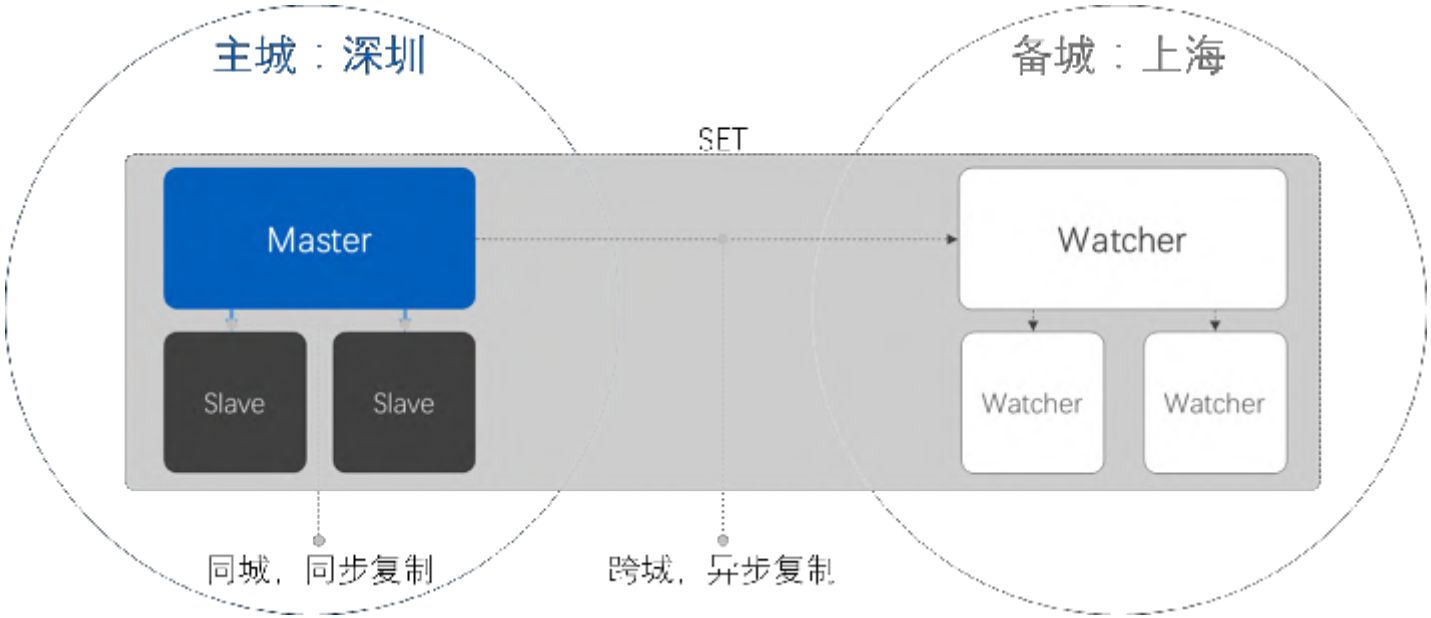
5、修改路由

6、请求发给新的主机

容灾切换流程（恢复）


A(主)
a+1,a+2,a+3
B(备)
a+1
C(备)
a+1,a+2
A当机，C选举
成新的主机
C(主)
a+1,a+2,c+1,c+2
B(备)
a+1,a+2,c+1,c+2
重新加入，可
能需要回退部
分事务
C(主)
a+1,a+2,c+1,c+2
回退事务a+3
B(备)
a+1,a+2,c+1,c+2
A(备)
a+1,a+2,a+3,c+
1,c+2
C(主)
a+1,a+2,c+1,c+2
Xtraback
up自动快
速重做
B(备)
a+1
D(备)
...

跨城容灾


主城：深圳
备城：上海
SET
Master
Slave
Slave
Watcher
Watcher
Watcher
同城，同步复制
跨城，异步复制

TDSQL的扩容机制

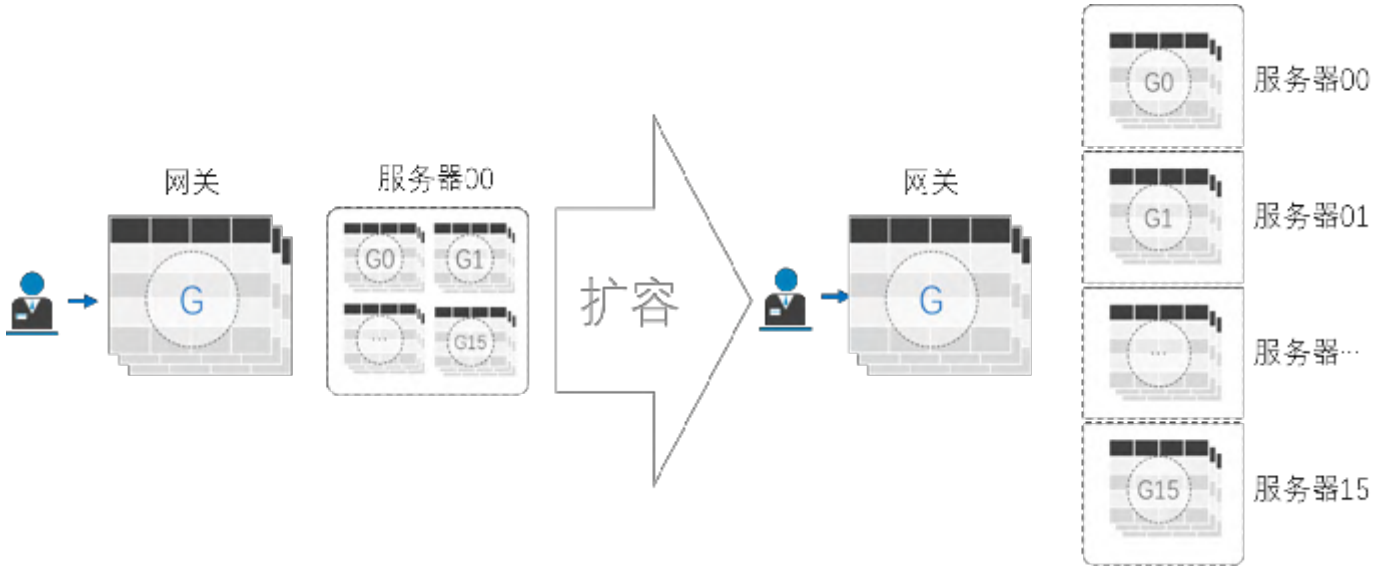
- 水平扩容
- 垂直扩容

扩容机制

- 水平扩容（Group_sharding）
- 垂直扩容（No_sharding）

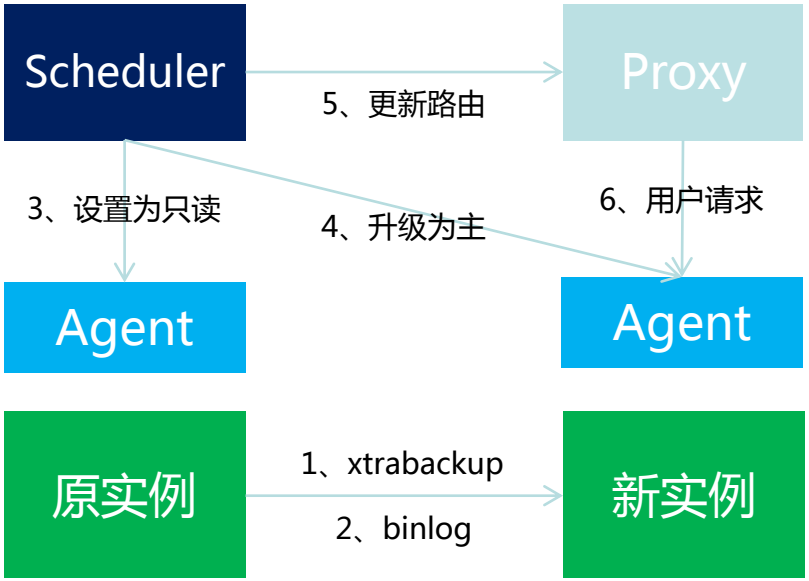
比较项	Group_sharding	No_sharding	MySQL
透明分库分表	✔	✘	✘
自动主备切换（7*24）	✔	✔	✘
高一致性保障	✔	✔	✘
自动容量伸缩	✔	✔(scale up)	✘
跨IDC/跨城容灾	✔	✔	✘

Group_Sharding的水平扩容



- Group：即为绑定在一起的一组表，使用相同的Sharding Key
- 支持单Shard内跨表事务及Join

No_sharding的垂直扩容



- 单实例最大6T
- 服务中断时长10s

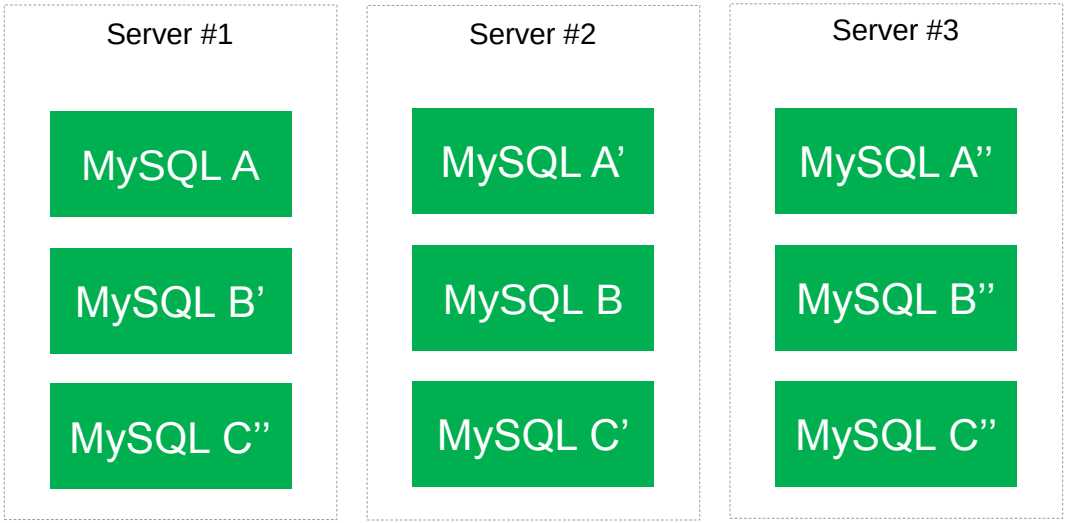
TDSQL的配套设施

- 冷备恢复中心
- 多租户多实例管理
- 数据库运营安全
- 管理配套工具

冷备恢复中心

- 基于HDFS的冷备中心
  - Binlog
  - 镜像
- 支持恢复到任意时间点
  - 取决于您购买的冷备服务
- 重建SET，并切换

多租户多实例管理



基于tlinux 2.0的cgroups隔离

- CPU
- Mem
- Net IO
- Disk IO

配套设施

安全：

- SQL及操作审计
- 完全的MySQL鉴权机制

管理：

- 数据导入工具
- 慢查询分析工具

TDSQL的运营

- 遇到的坑
- 运营数据

遇到的坑

- 自己的坑
  - 路由误删
- 开源的坑
  - MariaDB线程池
  - 其他常规BUG

运营情况

100亿
账户

10亿
日请求

5毫秒
响应

0
误差

800+
节点

? T
数据

5毫秒
响应

0
误差

TDSQL的下一步

- 配套工具的进一步完善
- 分布式事务
- 与社区的融合

SDCC
中国软件开发者大会
SOFTWARE DEVELOPER CONFERENCE CHINA

CSDN
全球最大中文IT社区

Thanks

sdcc.csdn.net

SDCC
中国软件开发者大会
SOFTWARE DEVELOPER CONFERENCE CHINA

CSDN
全球最大中文IT社区

欢迎各位大牛加入我们！

微信：pananq

sdcc.csdn.net