

饿了么异地多活的数据实施-DRC

饿了么框架工具部-陈永庭

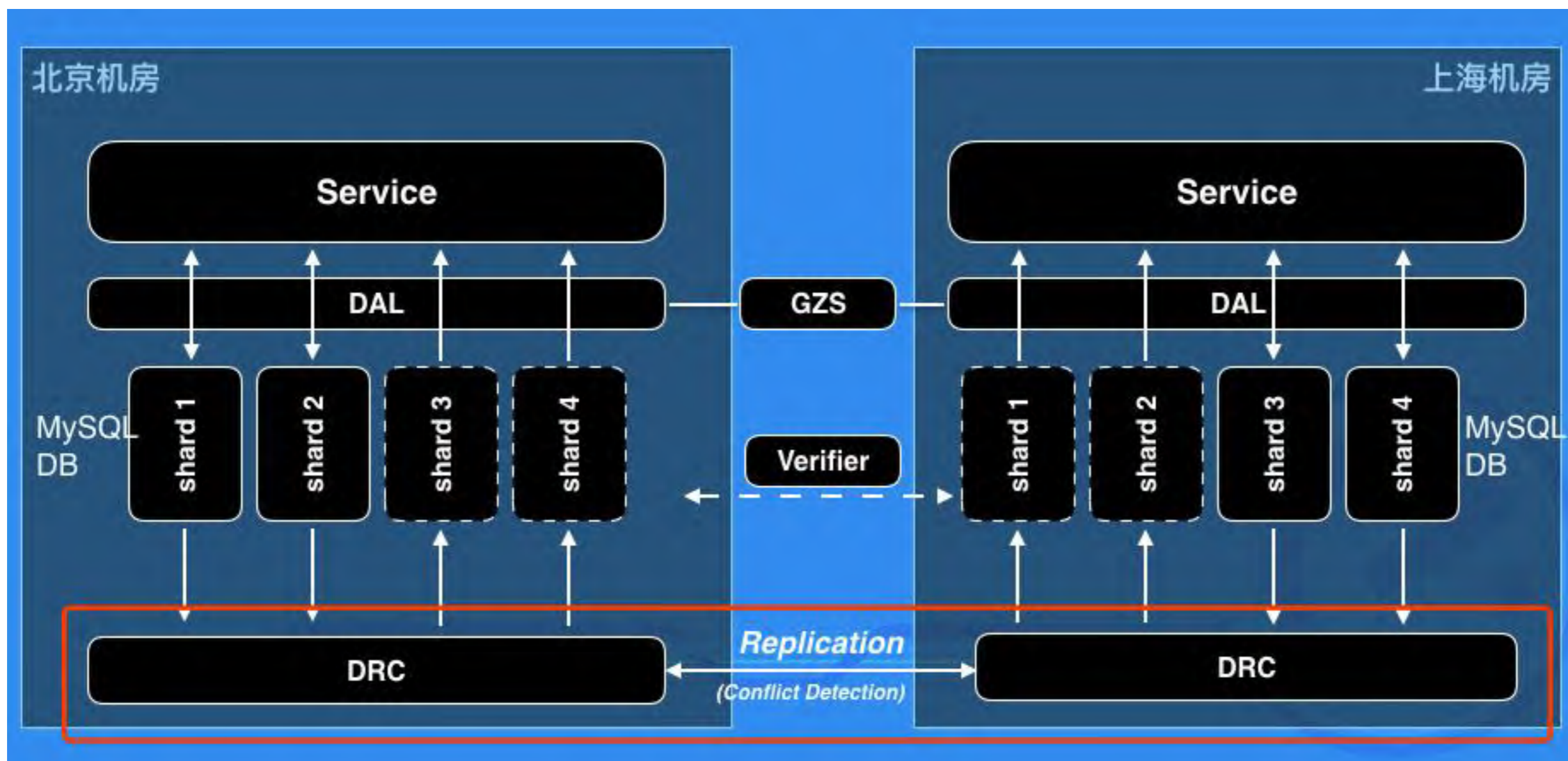
个人介绍

- 陈永庭
 - 2016.6入职饿了么、框架工具部高级架构师
 - 2016.9启动异地多活项目调研、有幸全程参入了多活的设计和实施
 - 目前主要服务于公司**DRC**项目的设计和研发，**DRC**组件支撑多活项目、提供低延迟、高吞吐的异地数据复制；以及为全产线业务服务提供数据变更消息通知服务

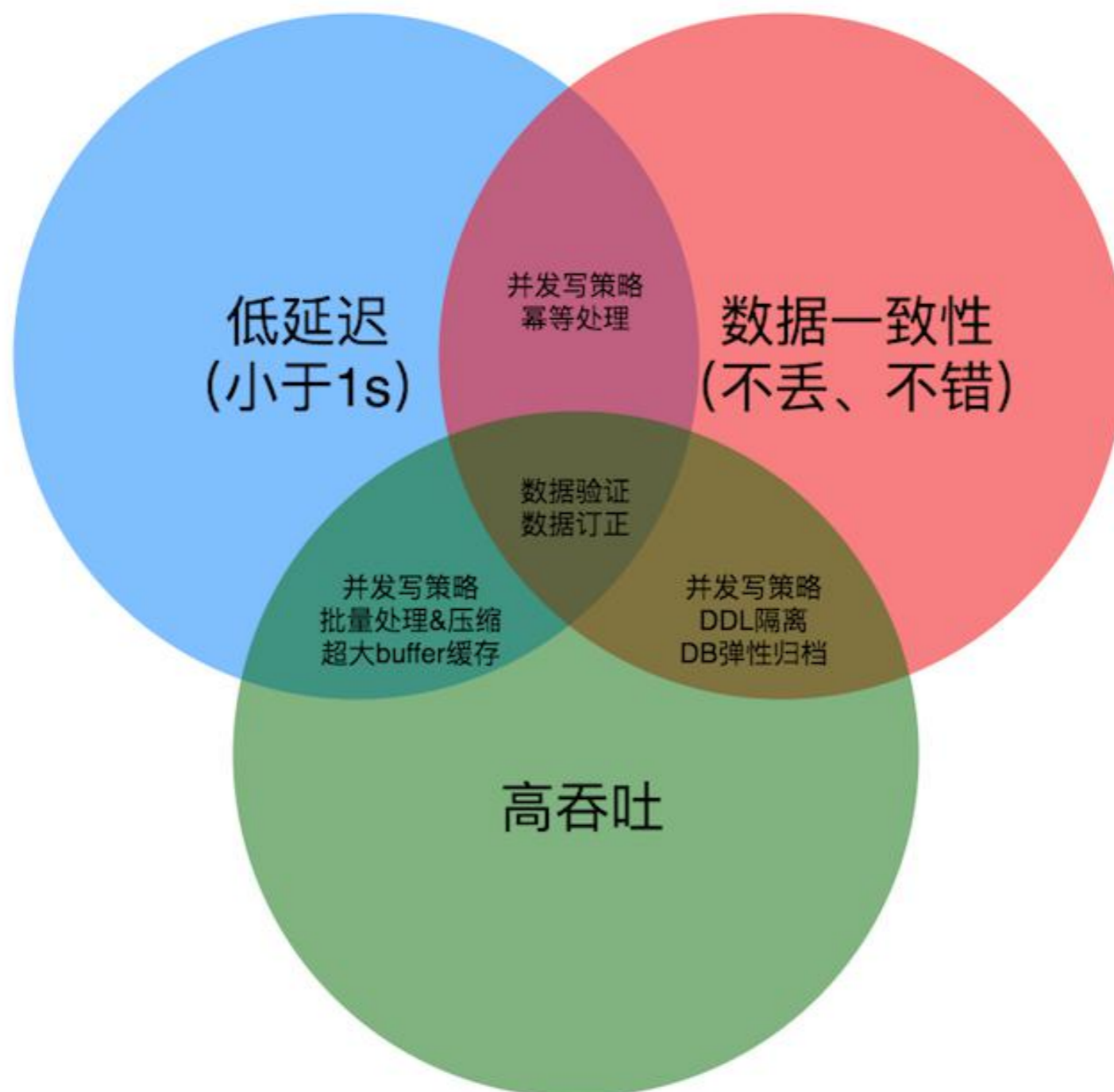
异地多活背景

- 饿了么异地双活
 - 北京IDC：4000+服务器；灾备IDC：3000+ vm
 - 业务状况：千万级日订单，北京IDC无法扩容，单IDC容量无法满足业务增长要求
 - 上海IDC正在建设，2017.4月份结束
 - 多活项目：2016.9月份开始调研和设计，2017.2启动全员参与开发，5月份前需要生产灰度

异地多活的底层数据同步实施



异地多活对底层数据的要求



数据集规模(多活改造前)

- 数据中心数量： 1
- MySQL集群： 250+（master: 250+实例， slave: 1000+实例）
- Redis集群： 400+

多活下MySQL的用途分类

Global DB

强事务要求服务

单IDC可写、其他IDC只读
IDC间单向复制
MySQL强事务一致
IDC挂掉会导致服务有损

多活DB

多活服务

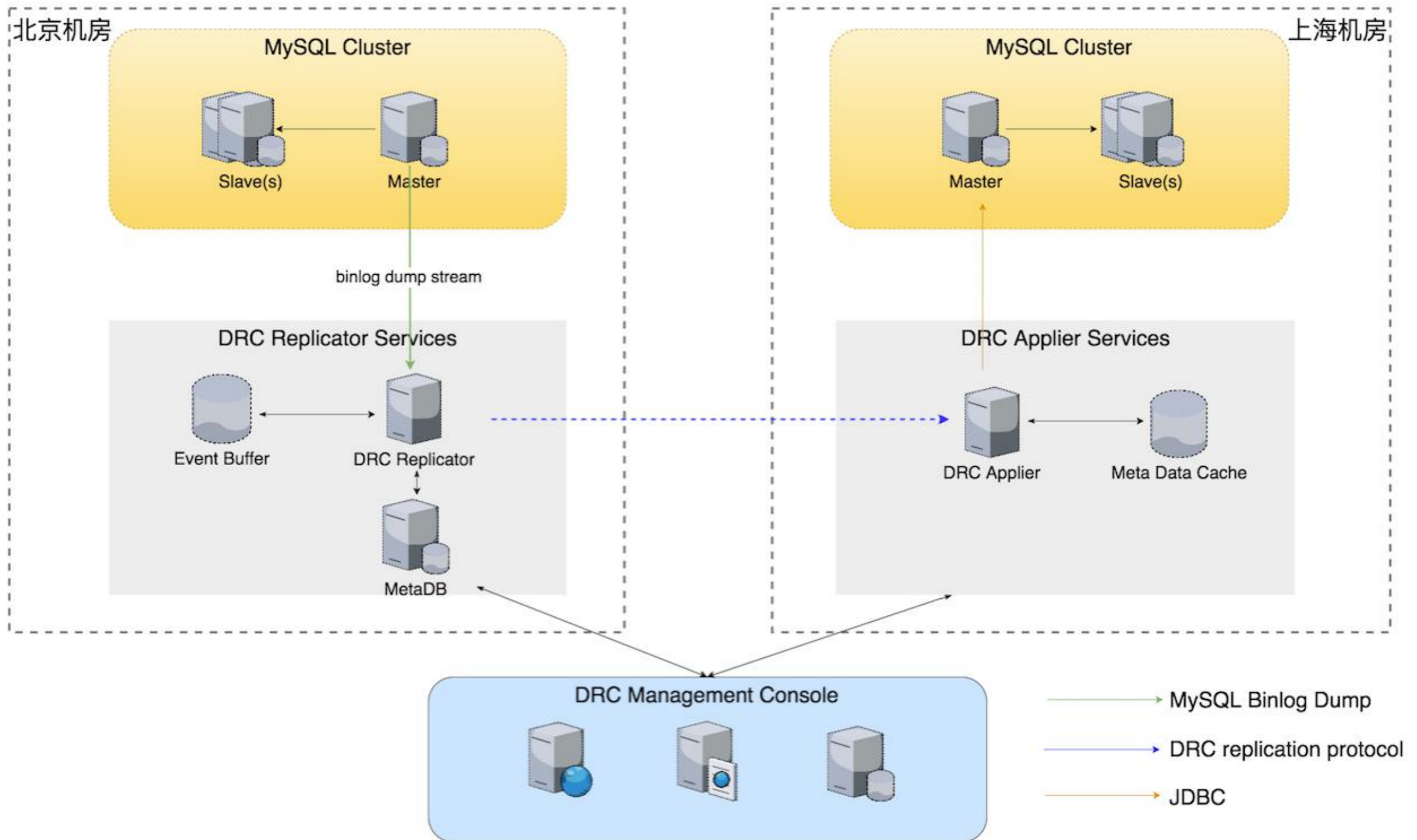
App多写 & 多度
DRC双向复制
DRC保障数据一致
IDC挂掉不影响服务

非多活DB

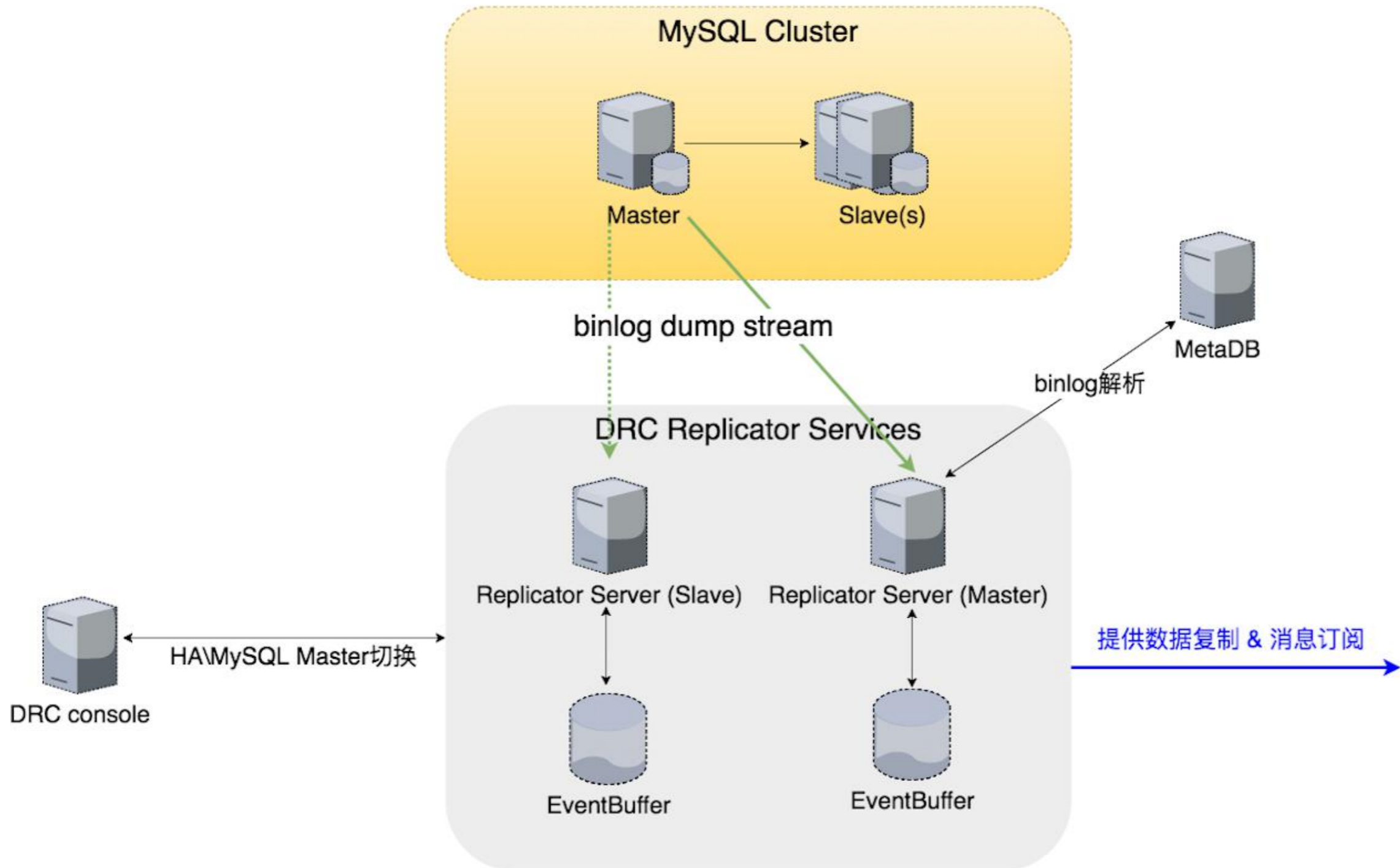
非多活服务

不支持多活服务
原生主从复制
IDC挂掉服务不可用

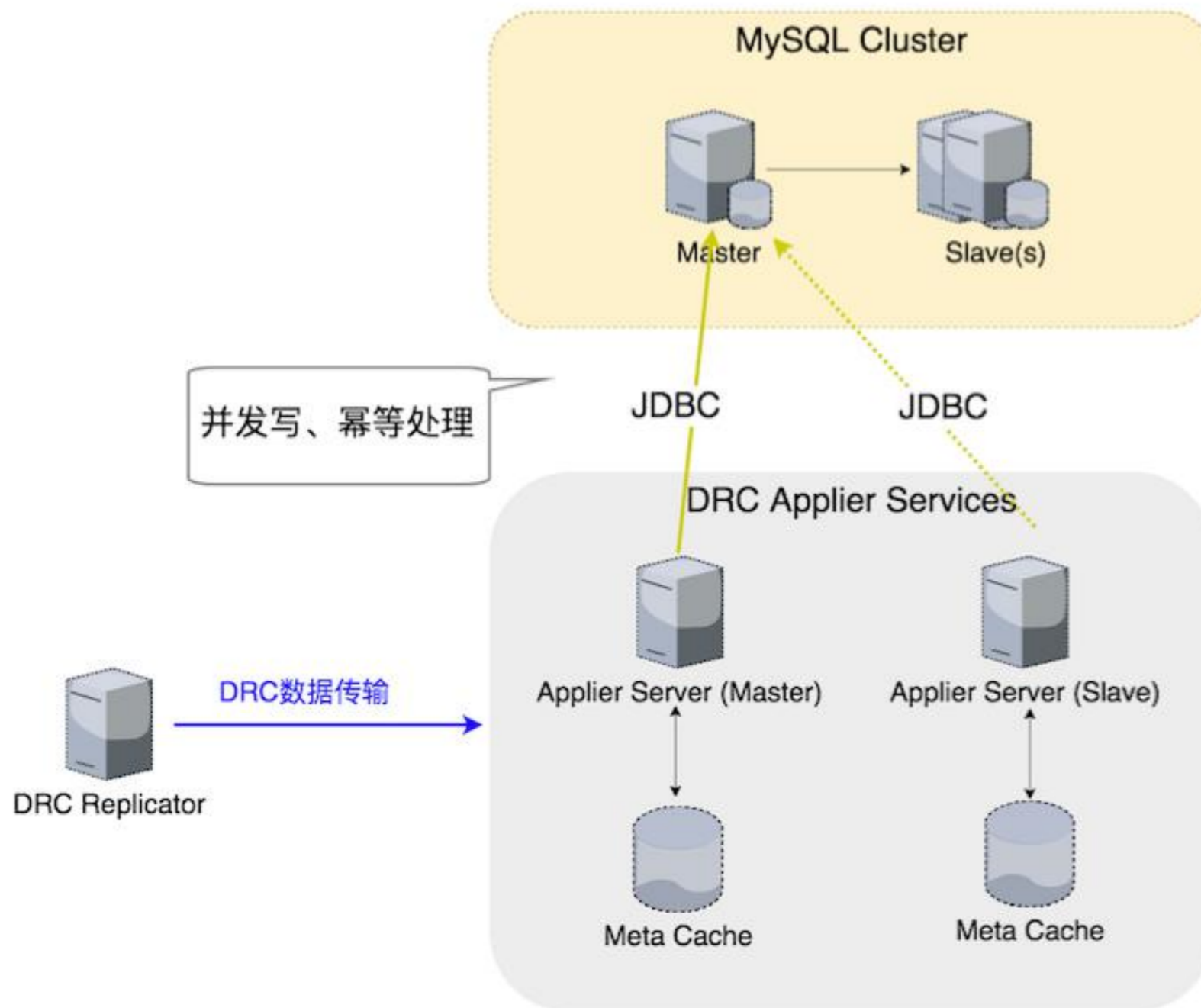
DRC总体架构设计



DRC Replicator Server



DRC Applier Server



DRC防止循环复制

- MySQL核内解决
 - ~~binlog改造~~
- DRC自身解决
 - ~~session disable binlog~~
 - checkpoint table

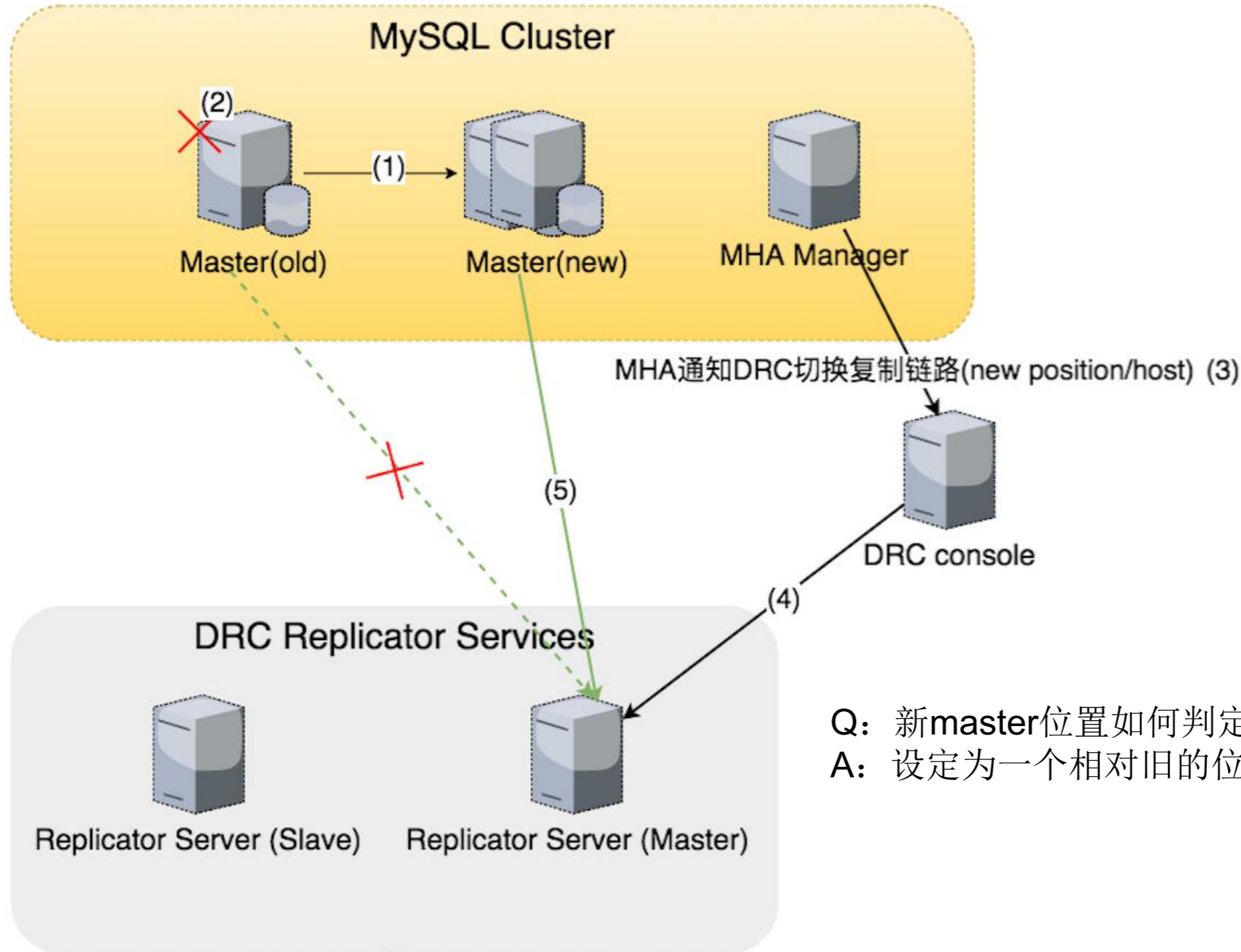
DRC数据一致性保障

- 多活不等于多写
 - IDC流量切分(uid\loc)
 - 订单在同一个IDC中完成流转、避免IDC多写
 - 避免了数据冲突的发生
- 数据幂等处理
 - 为什么需要幂等、幂等的重要性
 - 应对重复数据处理（HA/运维场景）
 - 避免了数据丢失
- 数据更新冲突（数据最新优先原则）

DRC数据复制低延迟保障

- **Applier**并发写
 - 守住原则：事务的时序性和数据幂等处理
 - 基于表/行级别的并发**apply**策略
 - **group commit**: 合并多个小事务，一次**commit**，减少**checkpoint table**操作次数
- 批量压缩数据传输
 - 只传输必要的**event logs**，减少数据传输
 - 批量**log**组包压缩，有效使用IDC间的专线带宽

DRC & MySQL Master切换



Q: 新master位置如何判定? 保证数据不丢。
A: 设定为一个相对旧的位置, DRC容忍重复数据处理

DRC线上运行状况（规模）

- 复制链路：400+
- 消息订阅
 - 目前共17个业务方接入订阅DRC消息
 - 1亿+/天 DRC消息

DRC线上运行状况（性能）



谢谢！

email: yongting.chen@ele.me