多活之路

饿了么多活的数据通路

个人介绍

李双涛

饿了么中间件团队首席架构师

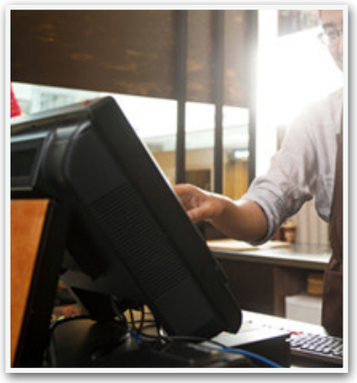
异地多活项目总架构师

DRC项目开发

饿了么简介

- 愿景：以创新科技打造全球领先的本地生活平台
- 主营外卖、商超、即时配送、餐饮供应链等业务
- 外卖覆盖城市数1400+，餐厅数100万+，日订单900万+，日运单450万+，骑手300万+用户浏览、下单、支付骑手取餐、送餐商家接单、备餐、呼叫配送

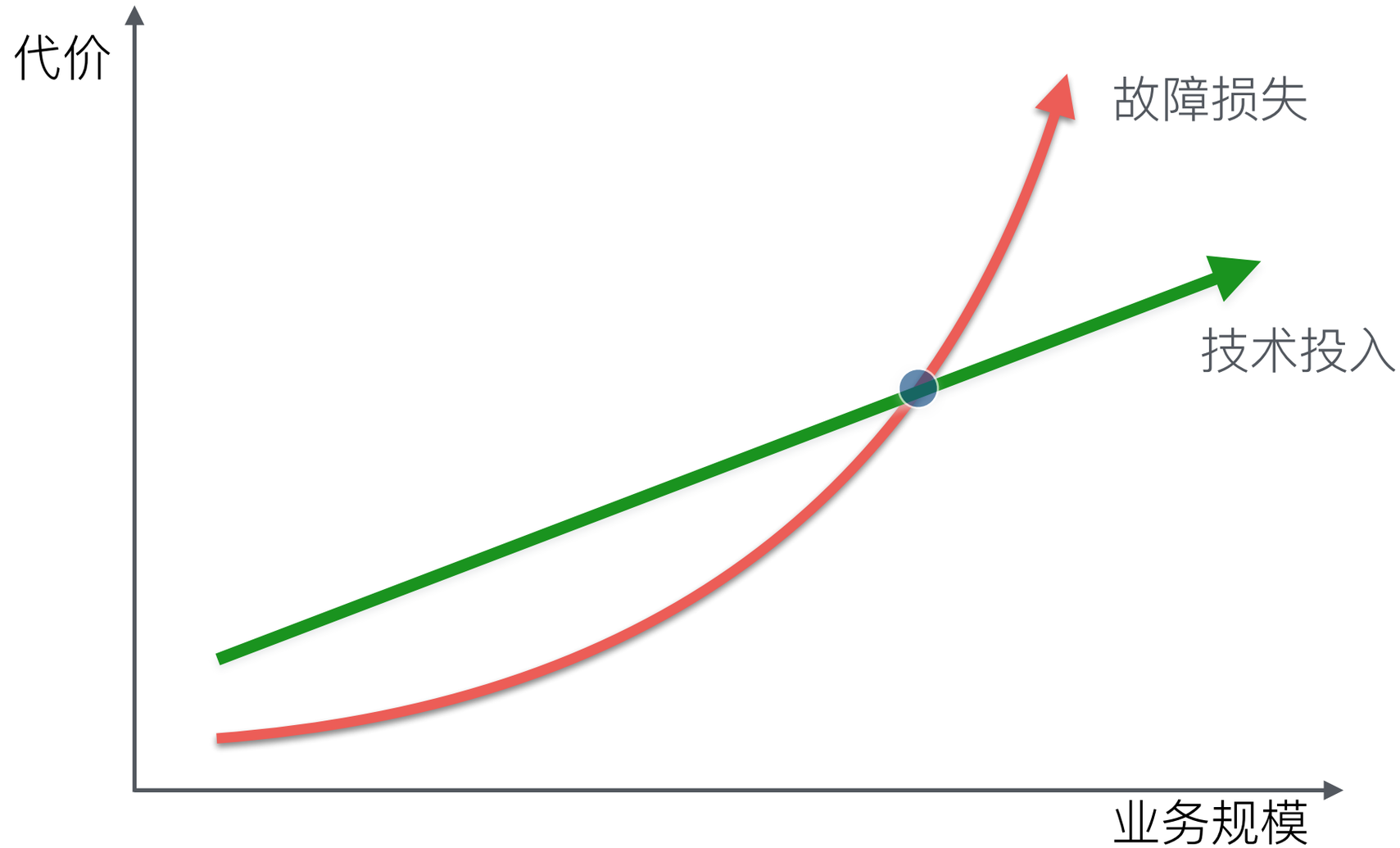
用户浏览、下单、支付



商家接单、备餐、呼叫配送

骑手取餐、送餐

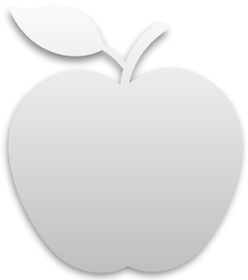
饿了么做多活的意义



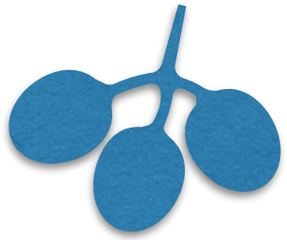
做多活，归根结底是个成本问题

1. 随着业务的增长，会达到一个临界点，故障的损失超过多活的投入
2. 单个机房的容量已经无法承载业务发展，需要突破单机房物理限制
3. 多活提升了服务品质，可用性提高，负载分担，用户就近接入

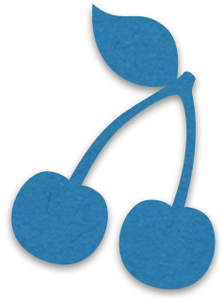
多活背景



多活投入很高



千万级日订单，故障代价极高

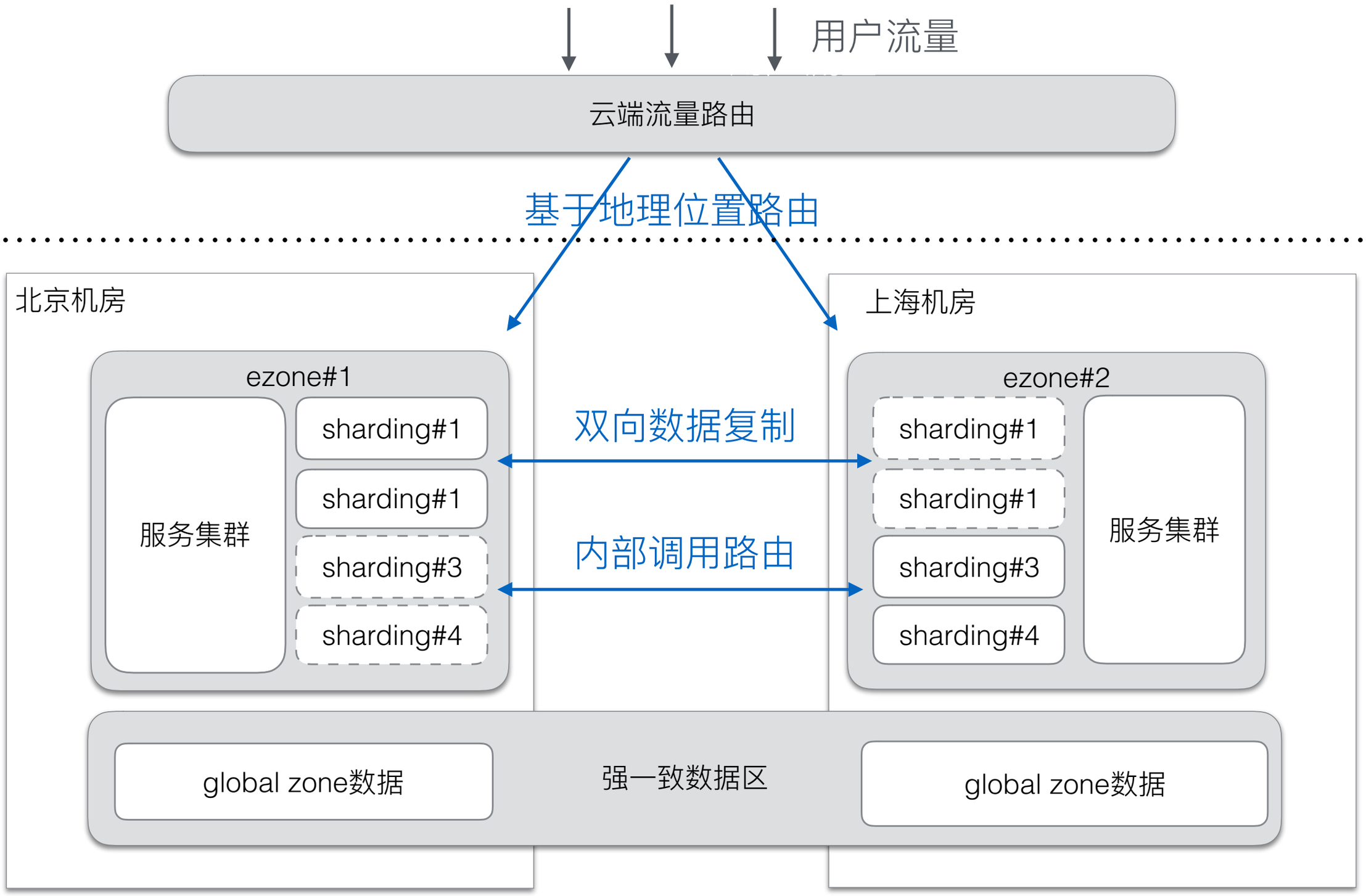


单IDC容量达到上限

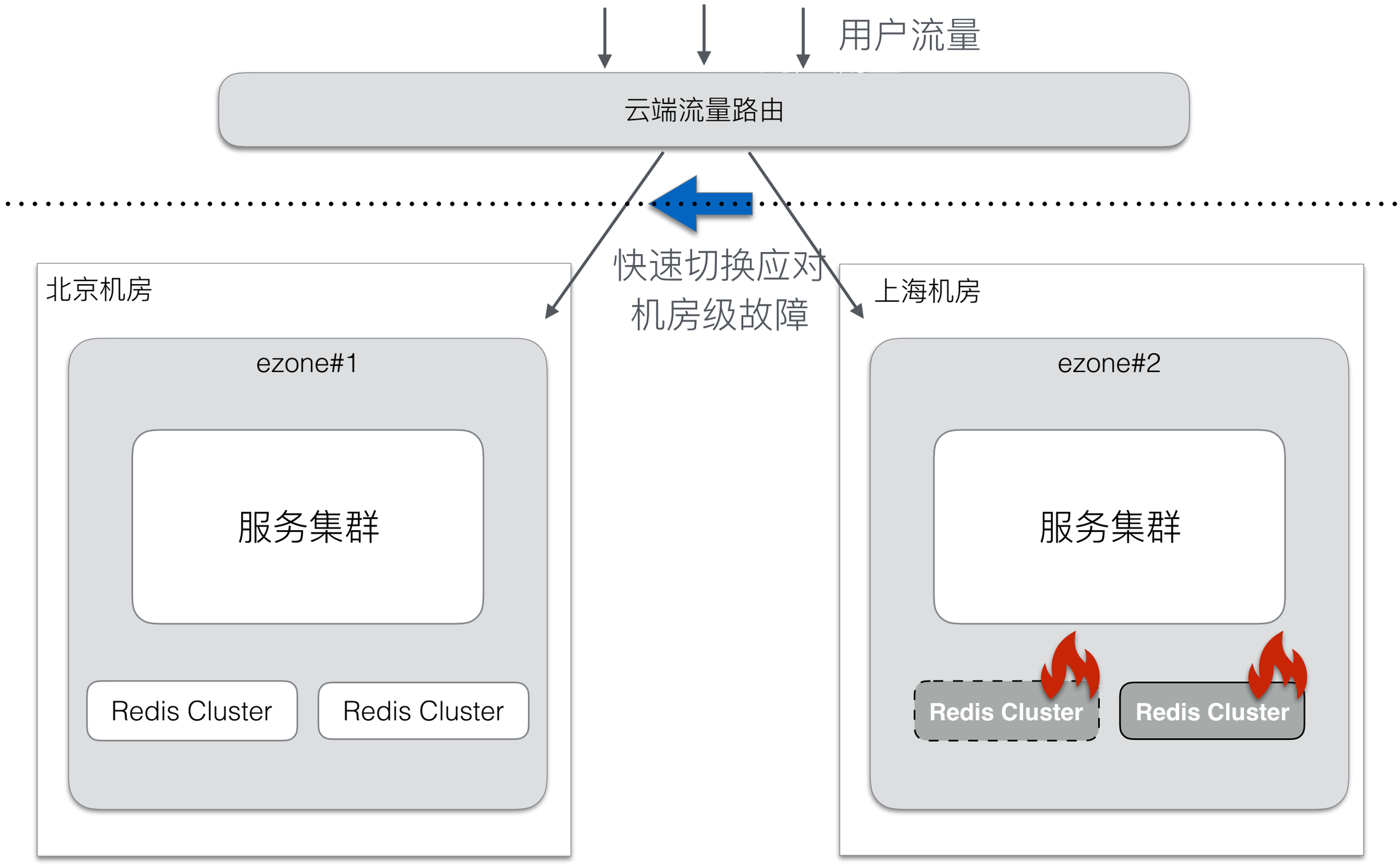
4000+服务器，3000+容灾

各类大促，导致的流量爆增

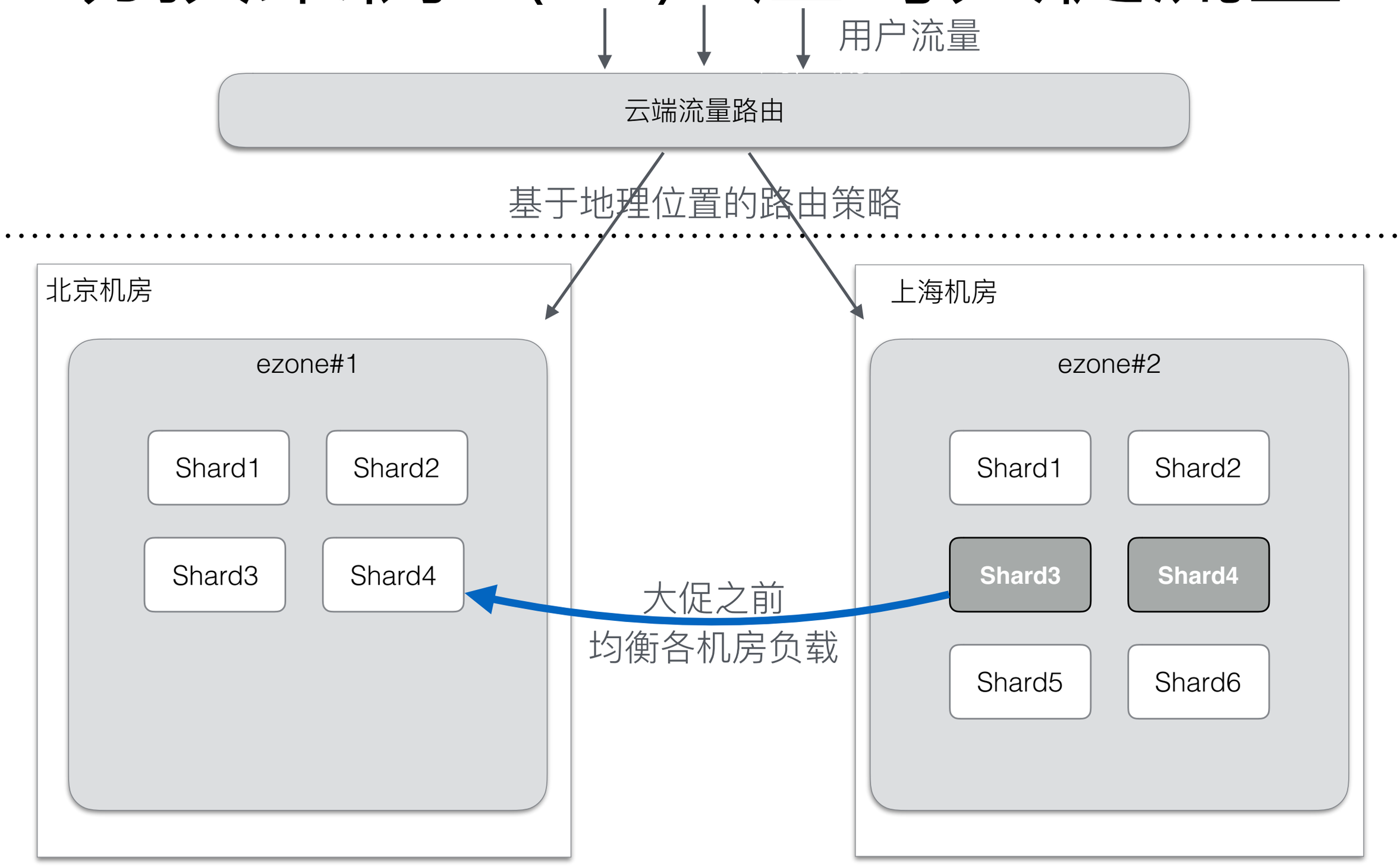
饿了么多活整体架构


用户流量
云端流量路由
基于地理位置路由
北京机房
上海机房
ezone#1
服务集群
sharding#1
sharding#1
sharding#3
sharding#4
双向数据复制
内部调用路由
ezone#2
sharding#1
sharding#1
sharding#3
sharding#4
服务集群
global zone数据
强一致数据区
global zone数据

切换案例（一） 基础组件Fail



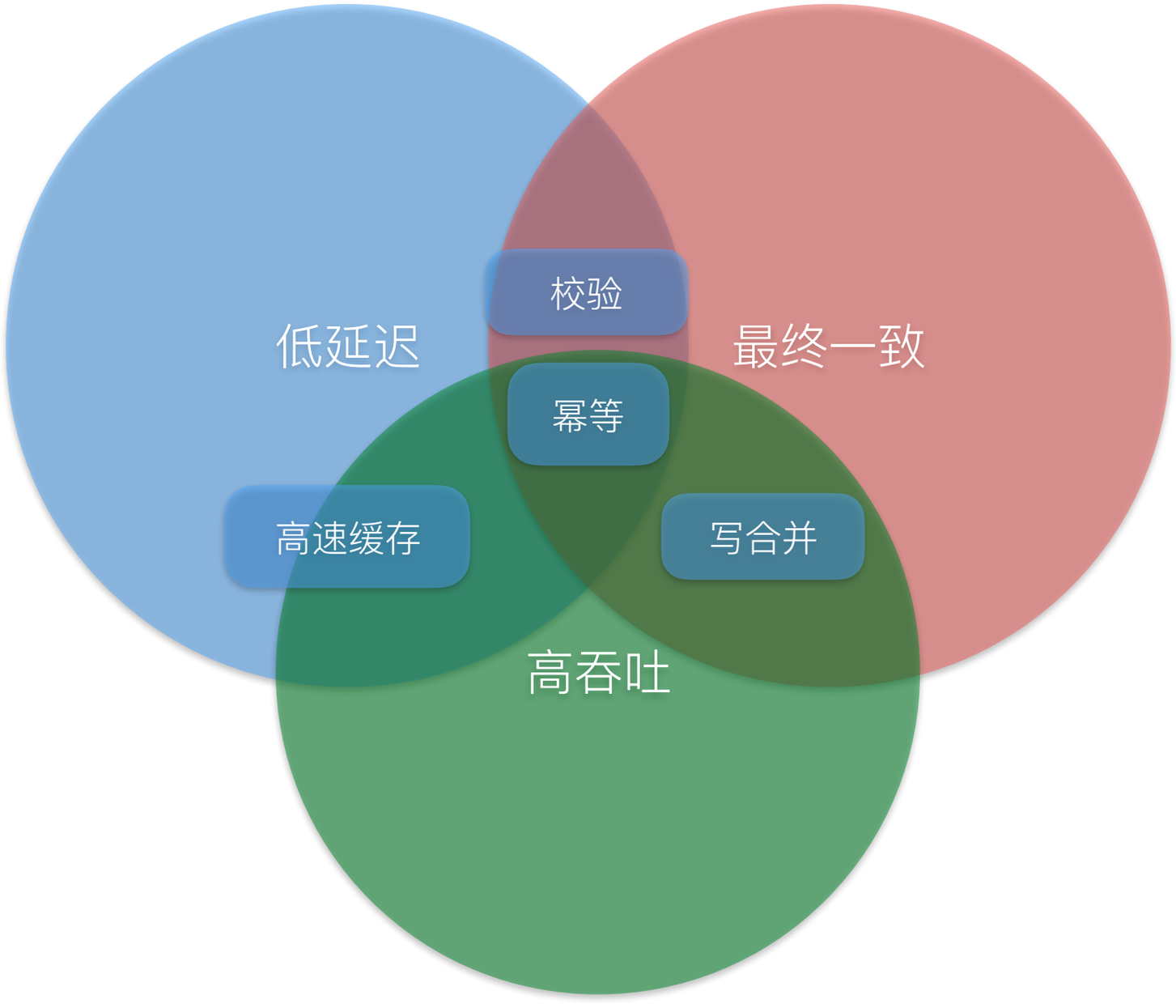
切换案例（二）应对大促流量



多活的数据通路


北京机房
ezone db instance
sharding#1
sharding#1
sharding#3
sharding#4
DB双向复制
DRC 复制工具
上海机房
ezone db instance
sharding#1
sharding#1
sharding#3
sharding#4
global zone数据
强一致数据区
global zone数据
其他落地数据
zookeeper
Message Queue
Redis
ZK复制
消息复制
Action复制
其他落地数据
zookeeper
Message Queue
Redis

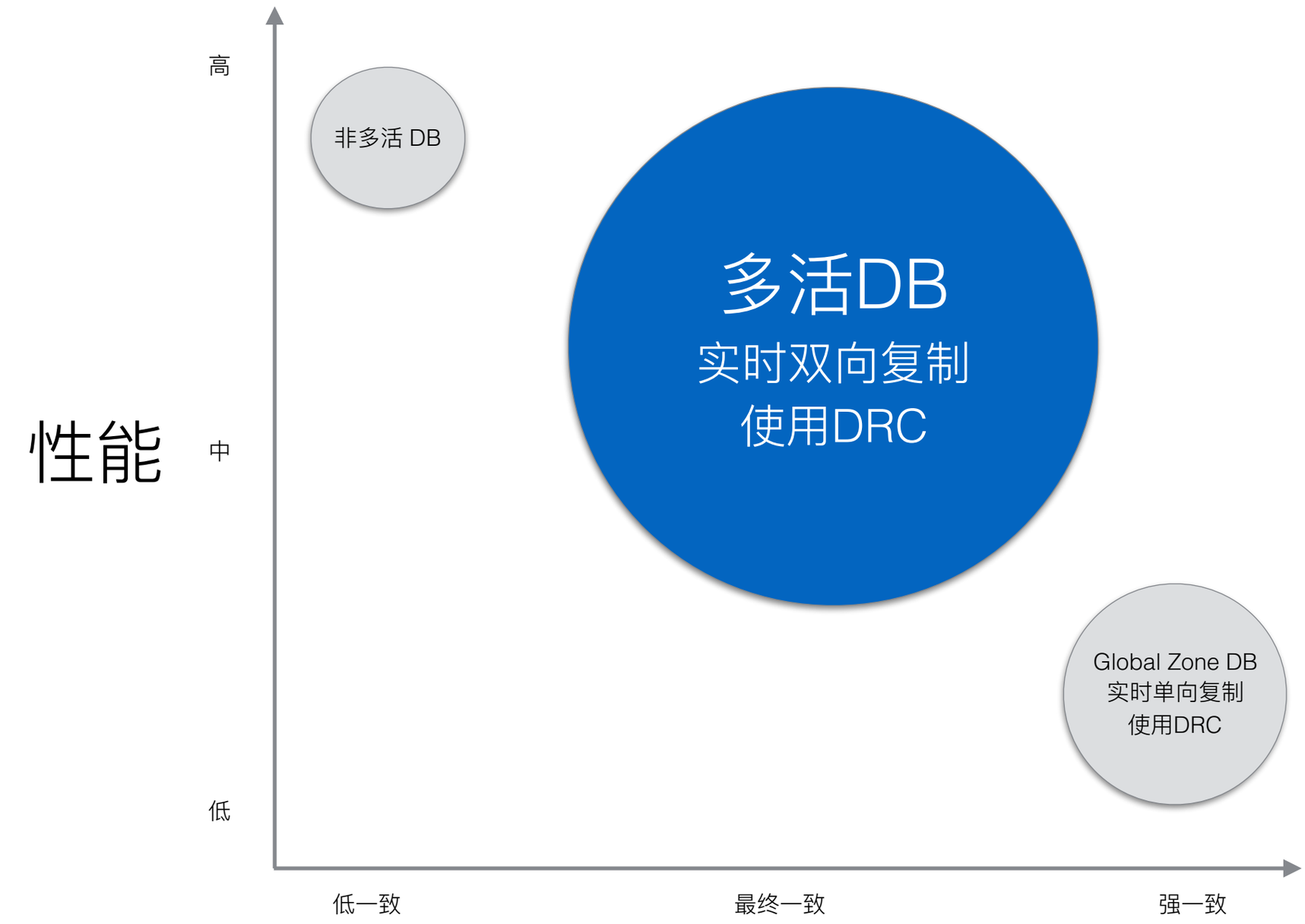
对DB双向复制的要求



数据集群规模(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+

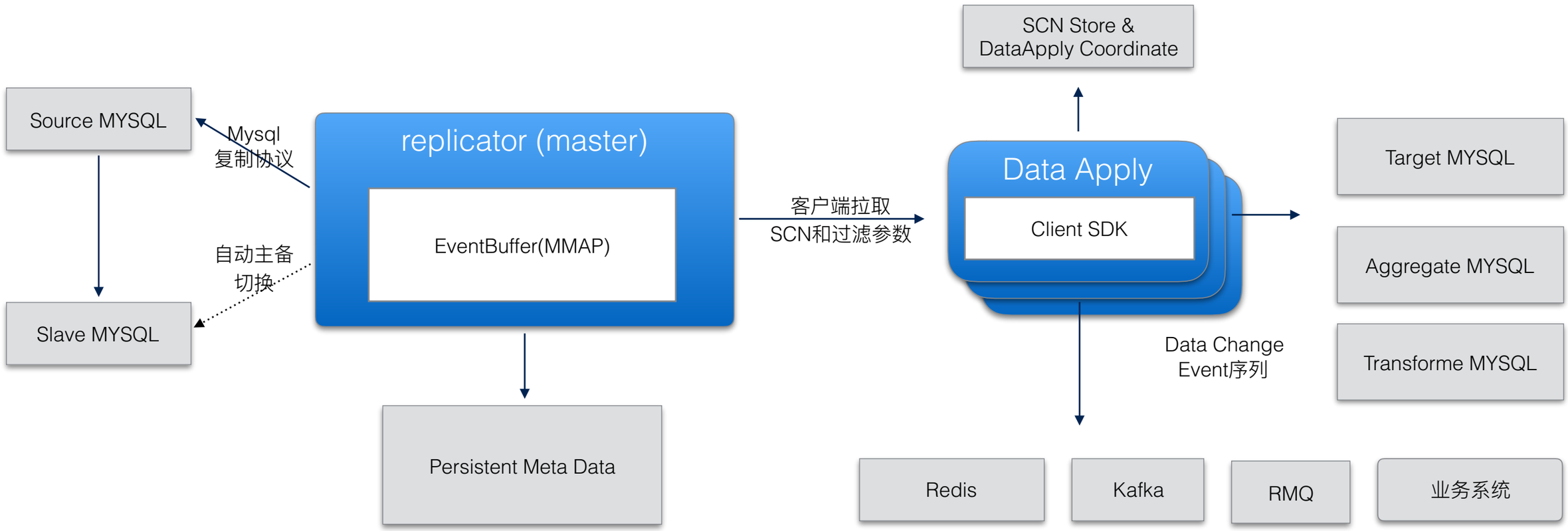
数据分类


高
中
低
性能
非多活 DB
多活DB
实时双向复制
使用DRC
Global Zone DB
实时单向复制
使用DRC
低一致
最终一致
强一致
一致性要求

数据分类

Global DB	多活DB	非多活DB
强事务要求服务	**多活服务**	**非多活服务**
单IDC可写、其他IDC只读	App多写 & 多度	不支持多活服务
IDC间单向复制	DRC双向复制	原生主从复制
MySQL强事务一致	DRC保障数据一致	IDC挂掉服务不可用
IDC挂掉会导致服务有损	IDC挂掉不影响服务	

总体架构



要点:

0.对于任何唯一的数据源，数据修改事件（DataChangeEvent）需要有一个唯一的单调递增的SCN。

1.Replicator实现Mysql Binary Log Dump协议，从Source Mysql中取得DataChangeEvent。

2.Replicator接收到DataChangeEvent数据后使用一个 Heap 外的环状内存结构（MMAP），减少GC负荷，每个 DBInstance 只需要一个Replicator抽取数据

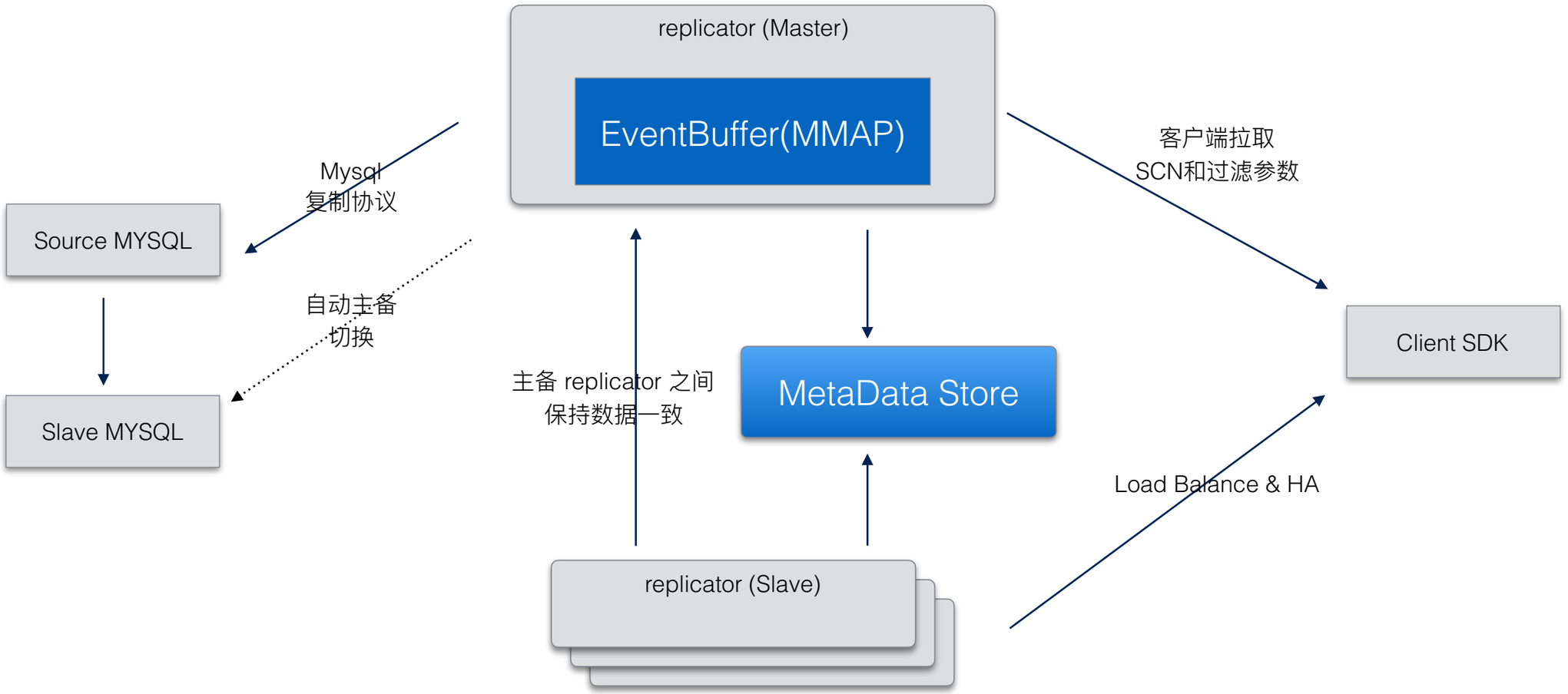
5.Replicator中保存当前SCN Number，有一个Master和任意多个Slave，当Master实效后，Slave Replicator可以选举新的Master

6.Client SDK从Replicator中拉取DataChangeEvent序列，拉取时需要提供当前位置和过滤条件。

7.Replicator中不保存任何Client SDK相关的状态信息（SCN），该信息由Client SDK调用方维护。

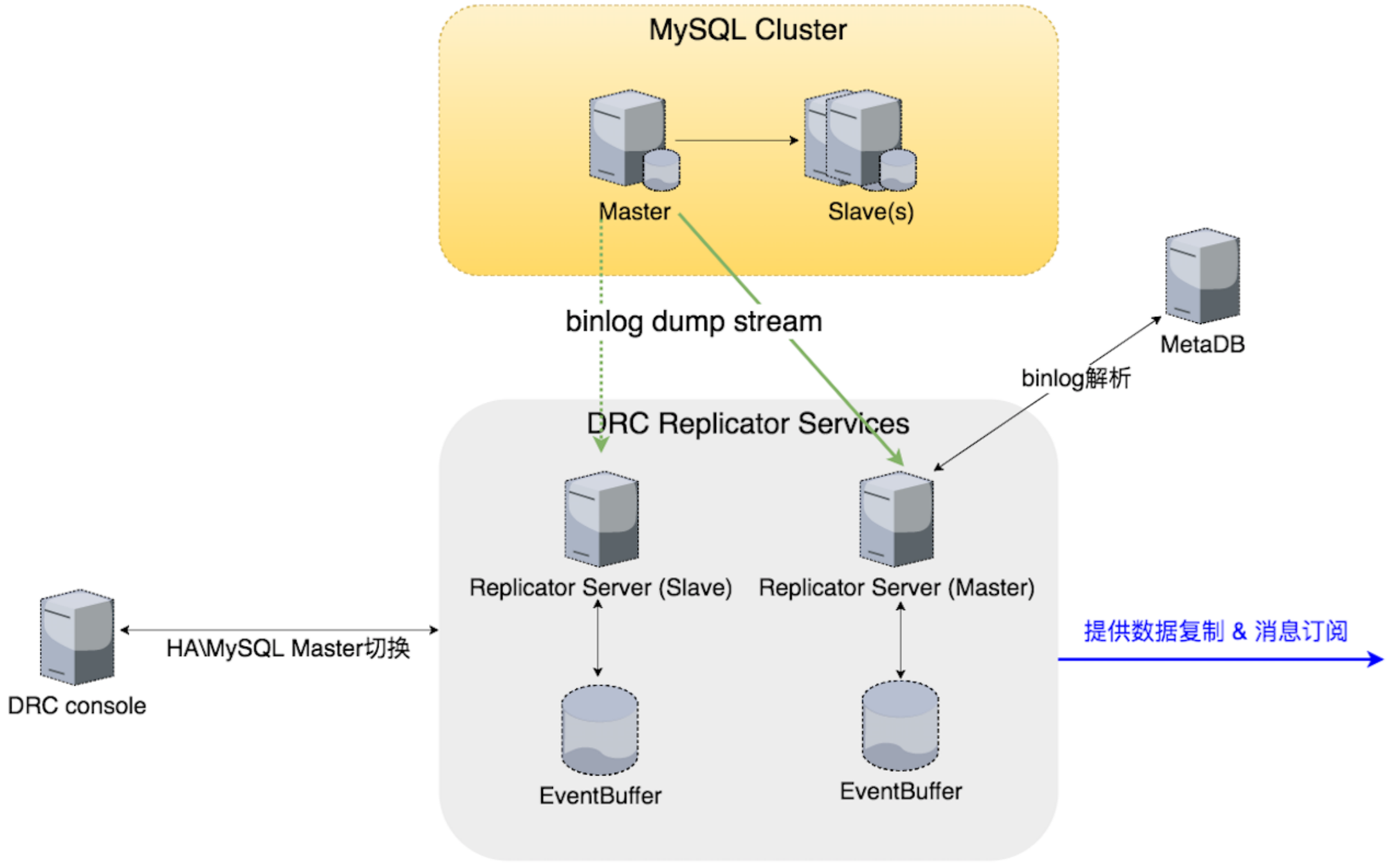
8.Data Apply负责把ChangeEvent按照需要的格式应用到数据库中，需要保持事务一致性，按照SCN的顺序。

Replicator

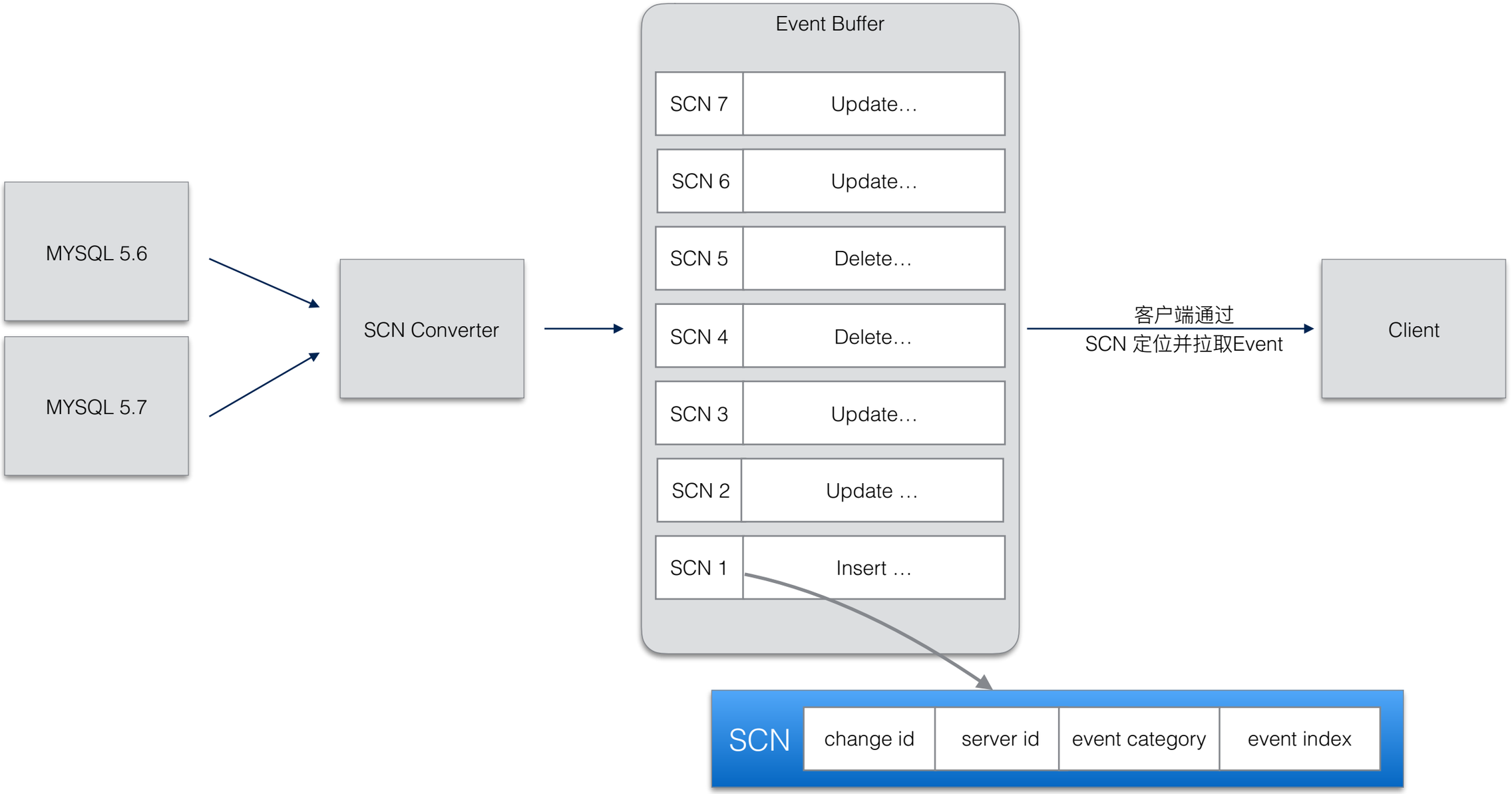


1. 部署时可以一个master多个slave，也可以多个master都连接DB，前者对DB压力小。
2. SCN在各个Replicator中是一致的，当master实效后会选举出新的master（zk），并从该master的当前位置开始复制。
3. Replicator中不保存客户端，Client SDK Pull数据的时候需要制定开始SCN位置，所以可以随时切换到任何一个Replicator拉取数据。
4. 如果客户端提交的SCN超过当前Replicator的最老数据，SDK会重定向到Bootstrap Service拉取更久之前的数据。
5. Bootstrap Service也是一个Client SDK，持续从master拉取数据，并保存到本地文件系统的EventLogStore File中，保存的时间可以很长，并且维护一个简单的索引系统，用于快速查找指定SCN的位置。
6. Bootstrap Service还可以维护一个数据SnapShot服务，在需要的时候方便Client数据库快速基于Snapshot + 回放EventLog建立初始数据。

Replicator 物理部署

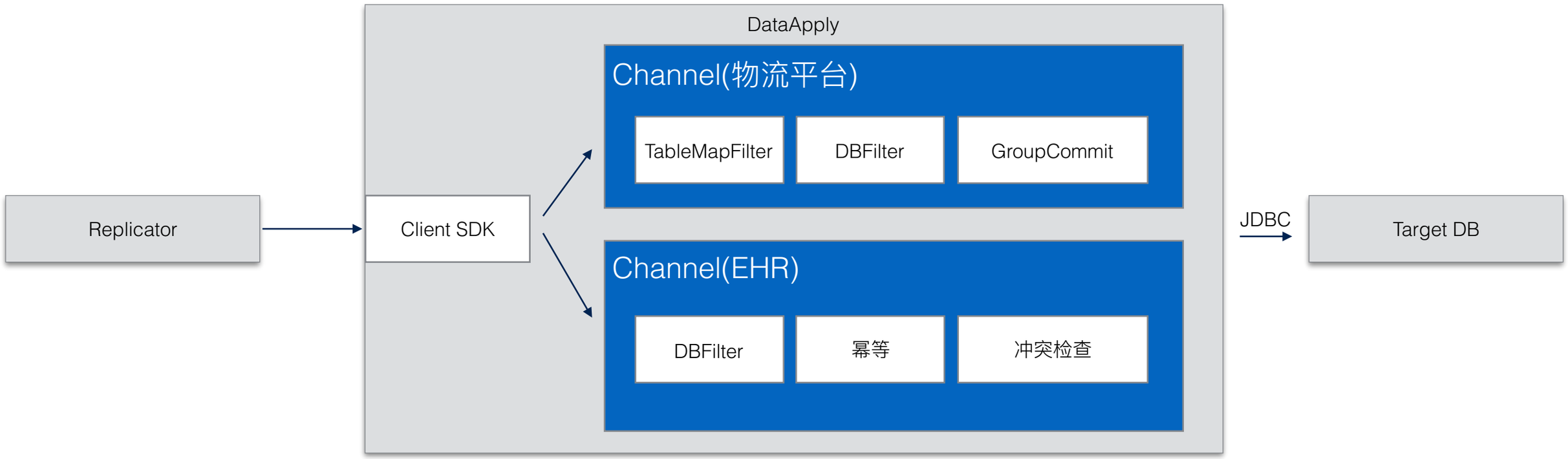


SCN（Series Change Number）



1. SCN单调有续，并全局唯一，绑定到唯一的事件上
2. 所有 Replicator 中 SCN 一致，SCN 的产生逻辑需要绑定到数据源的唯一逻辑上

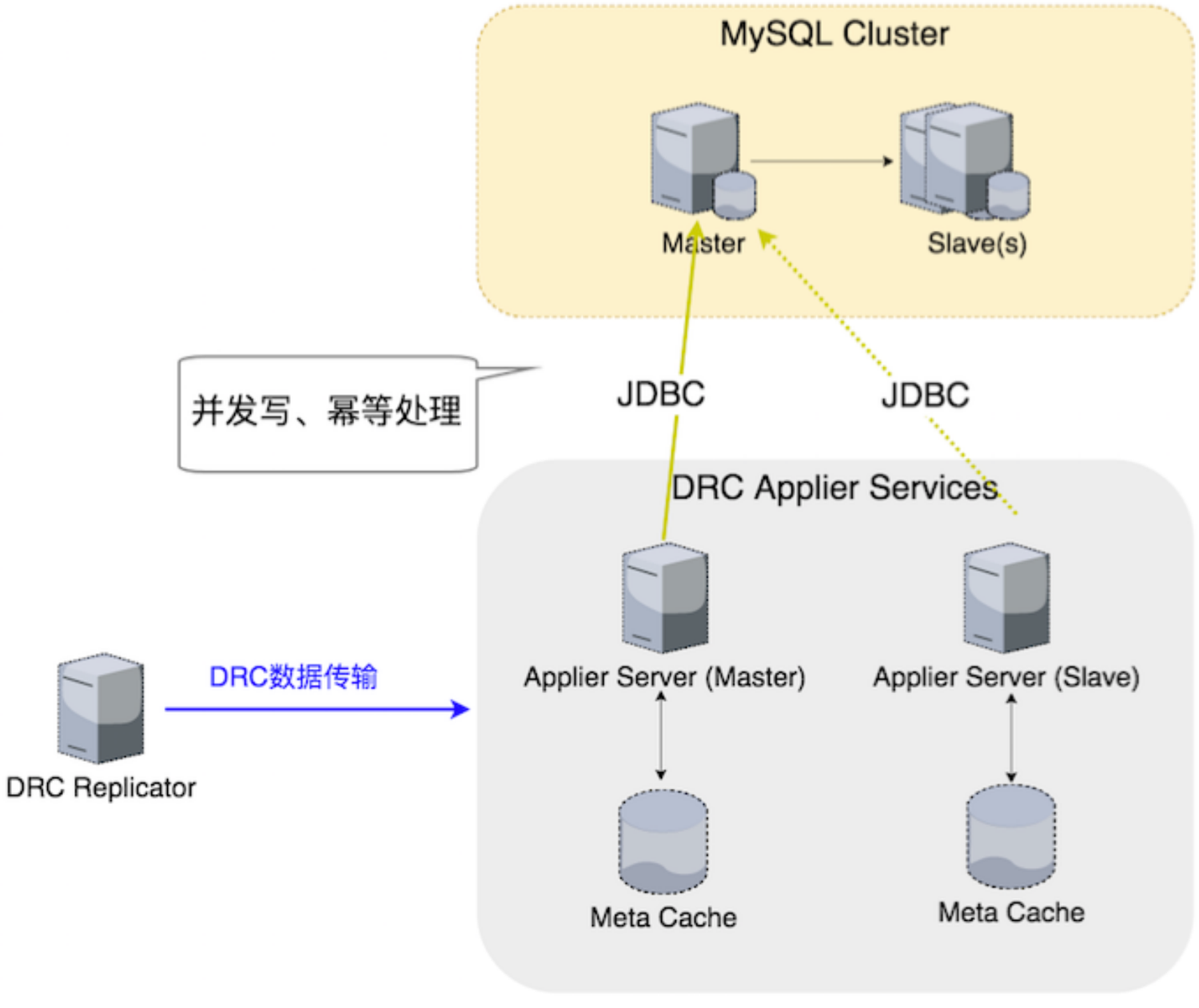
Data Apply



要点：

1. 以Channel的形式在DataApply Server上组织复制单元
2. Channel是一张表或者一组表
3. Channel内部逻辑通过串联的Filter实现
4. Channel支持运行时动态配置（Hascar）

DRC Apply Server

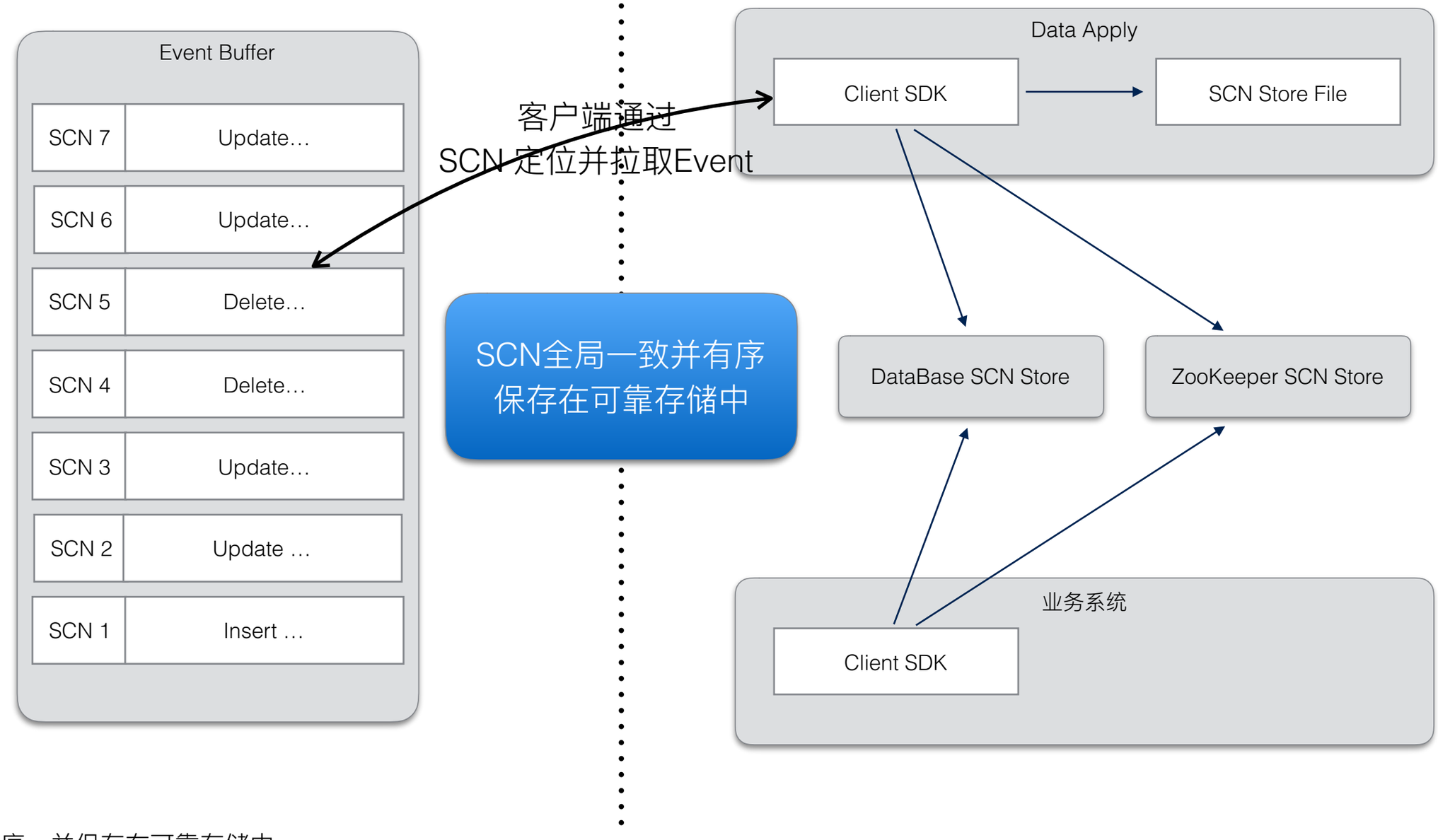


DRC数据一致性保障

- 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）

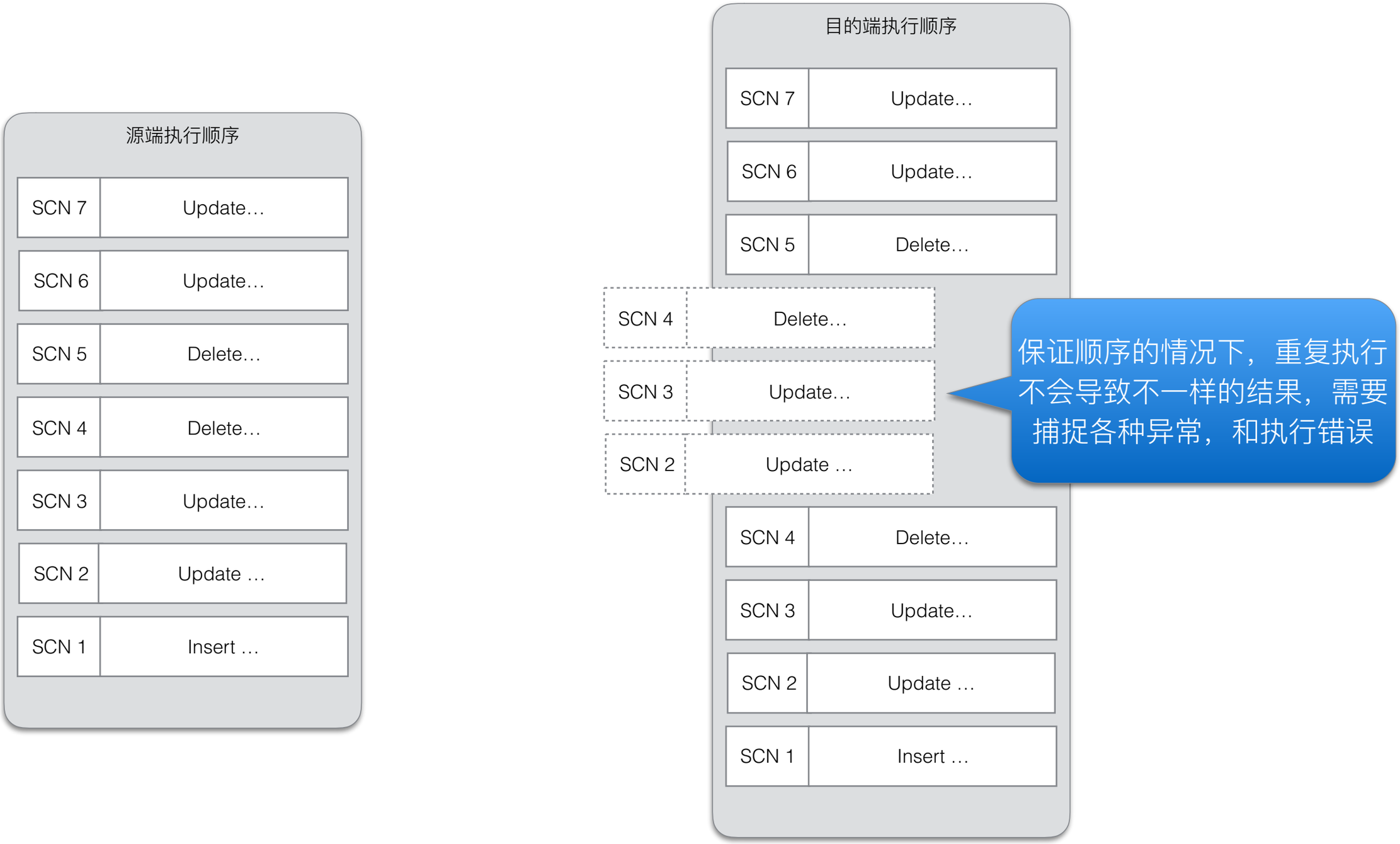
一致性保障

数据一致性

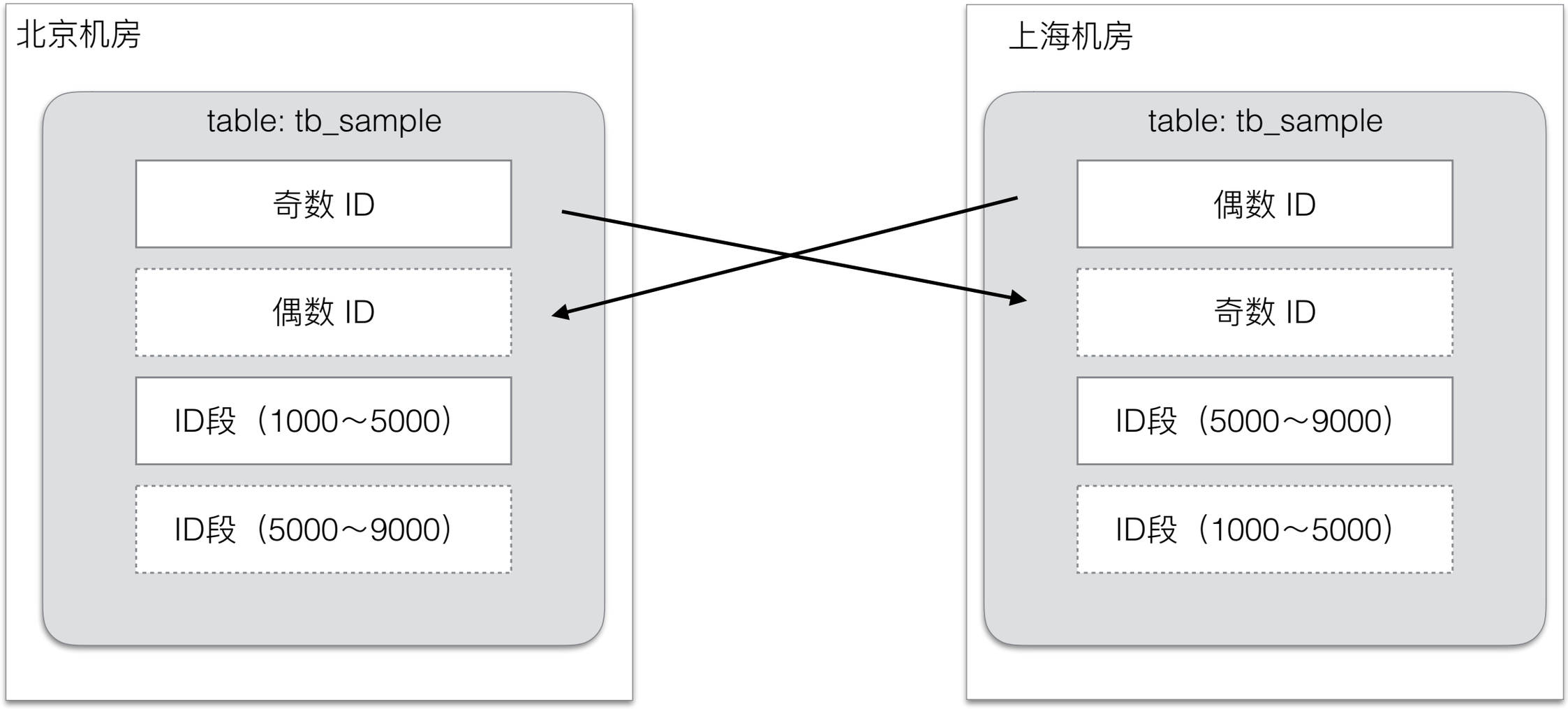


1. SCN有序，并保存在可靠存储中
2. 任何节点的失效，都可以通过SCN位点来恢复

复制幂等

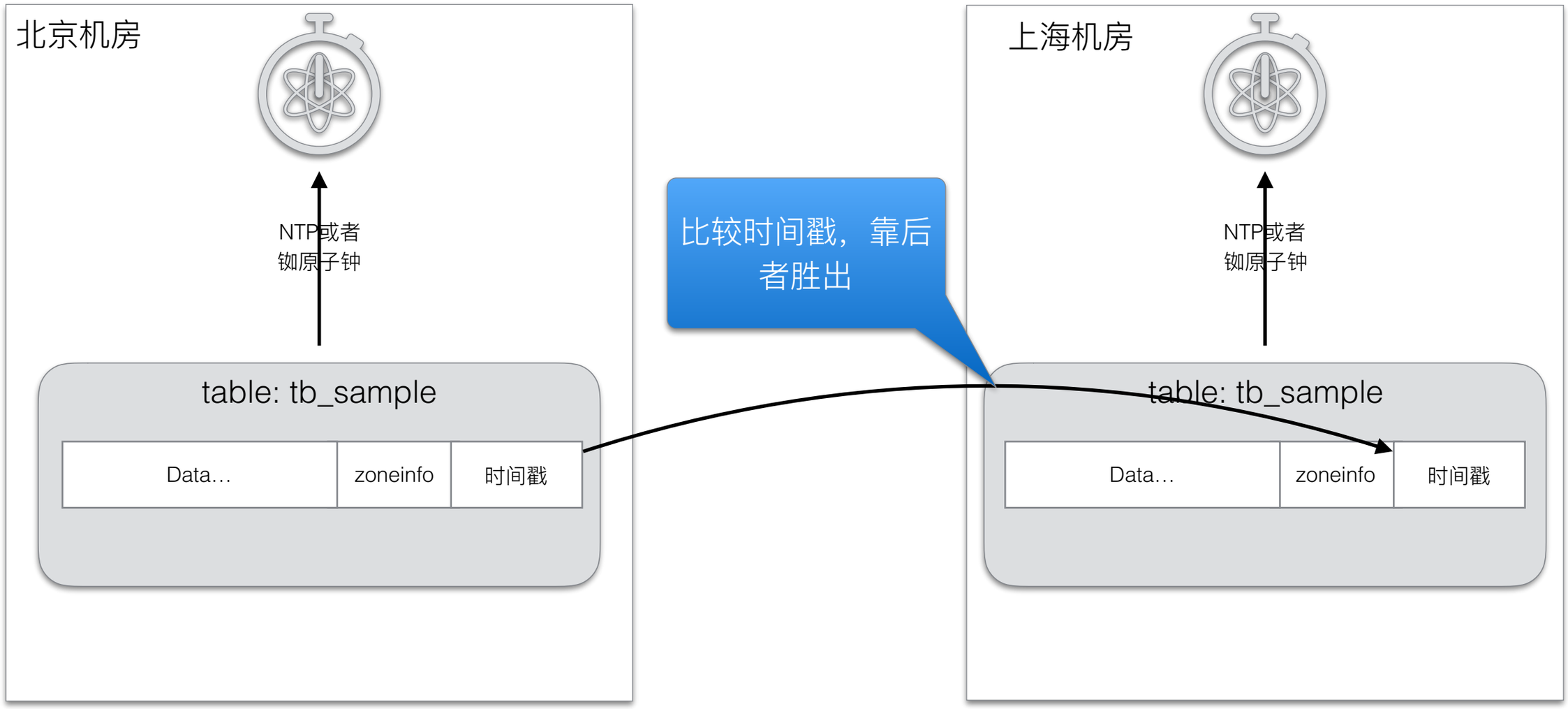

源端执行顺序
SCN 7
Update…
SCN 6
Update…
SCN 5
Delete…
SCN 4
Delete…
SCN 3
Update…
SCN 2
Update …
SCN 1
Insert …
目的端执行顺序
SCN 7
Update…
SCN 6
Update…
SCN 5
Delete…
SCN 4
Delete…
SCN 3
Update…
SCN 2
Update …
SCN 4
Delete…
SCN 3
Update…
SCN 2
Update …
SCN 1
Insert …
保证顺序的情况下，重复执行不会导致不一样的结果，需要捕捉各种异常，和执行错误

多点写入的一致性保证



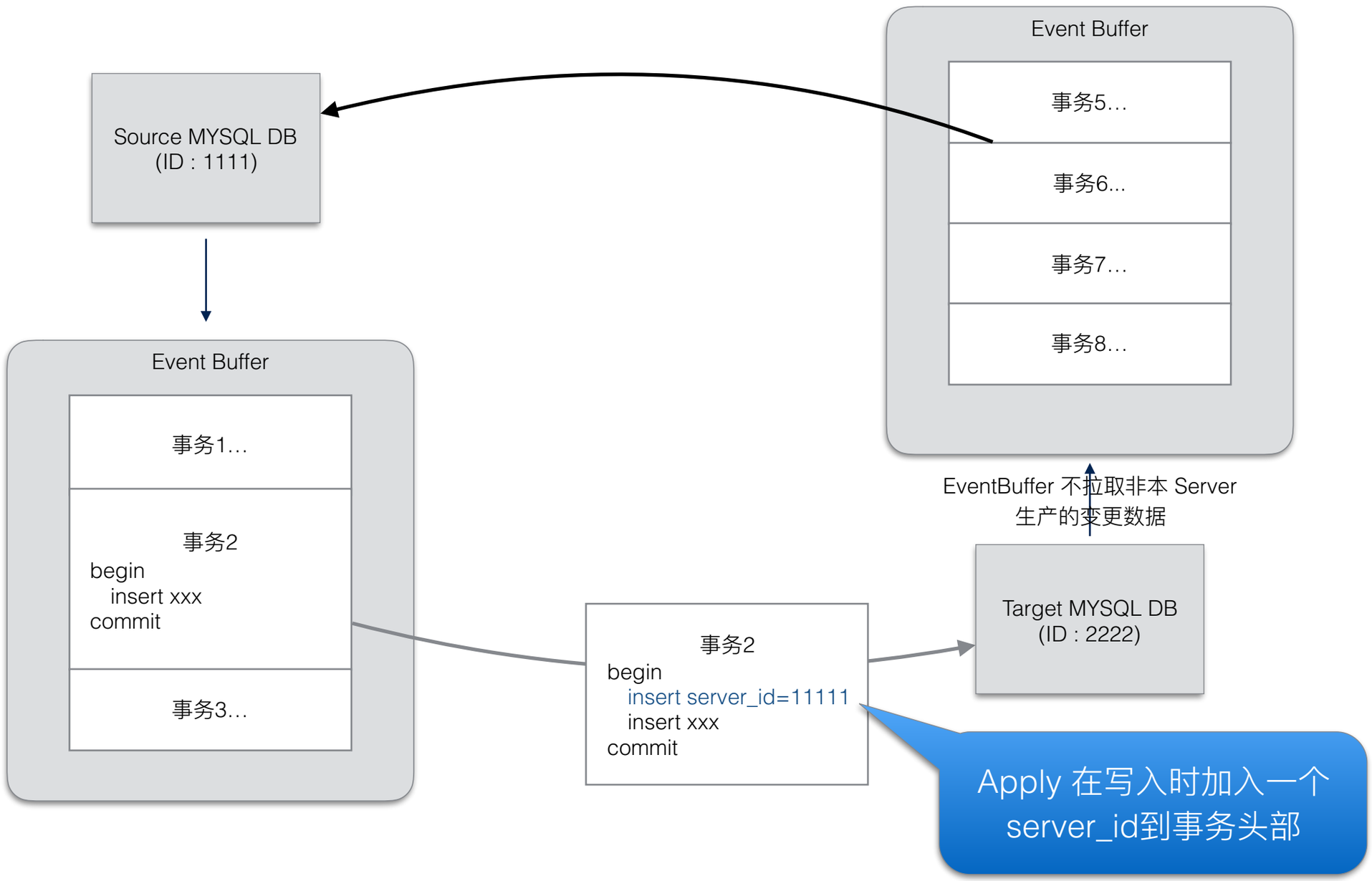
1. 不同机房产生不同的数据，全量数据不会冲突
2. 数据访问中间件（DAL）会拒绝掉不正确的写入请求

冲突处理



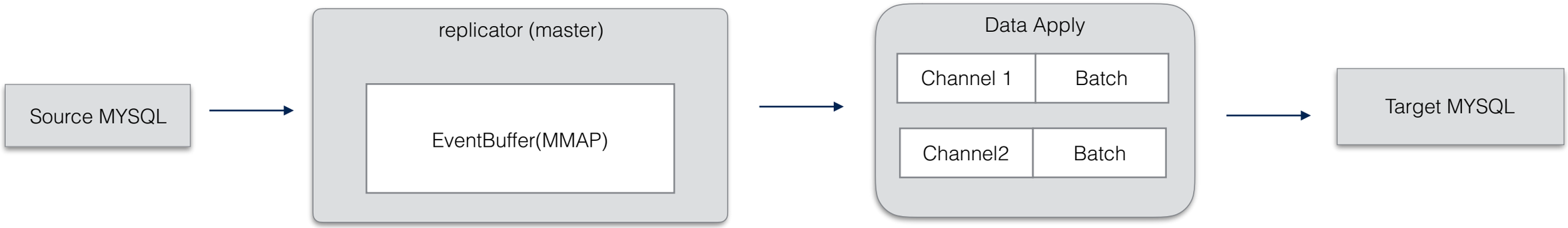
1. 万一发生同一笔数据，在两个机房同时修改，则引入冲突处理
2. 基本的冲突处理，通过时间戳完成，如果时间戳不能满足需要，通过调用业务提供的冲突解决方案解决
3. 冲突解决时，为业务方提供了原始数据和最新数据

DRC防止循环复制



1. 通过在复制事务中加入Hack SQL，标记数据来源 DB
2. 未来可以通过修改 Mysql Binlog 机制实现，减少资源消耗

低延迟高吞吐保障



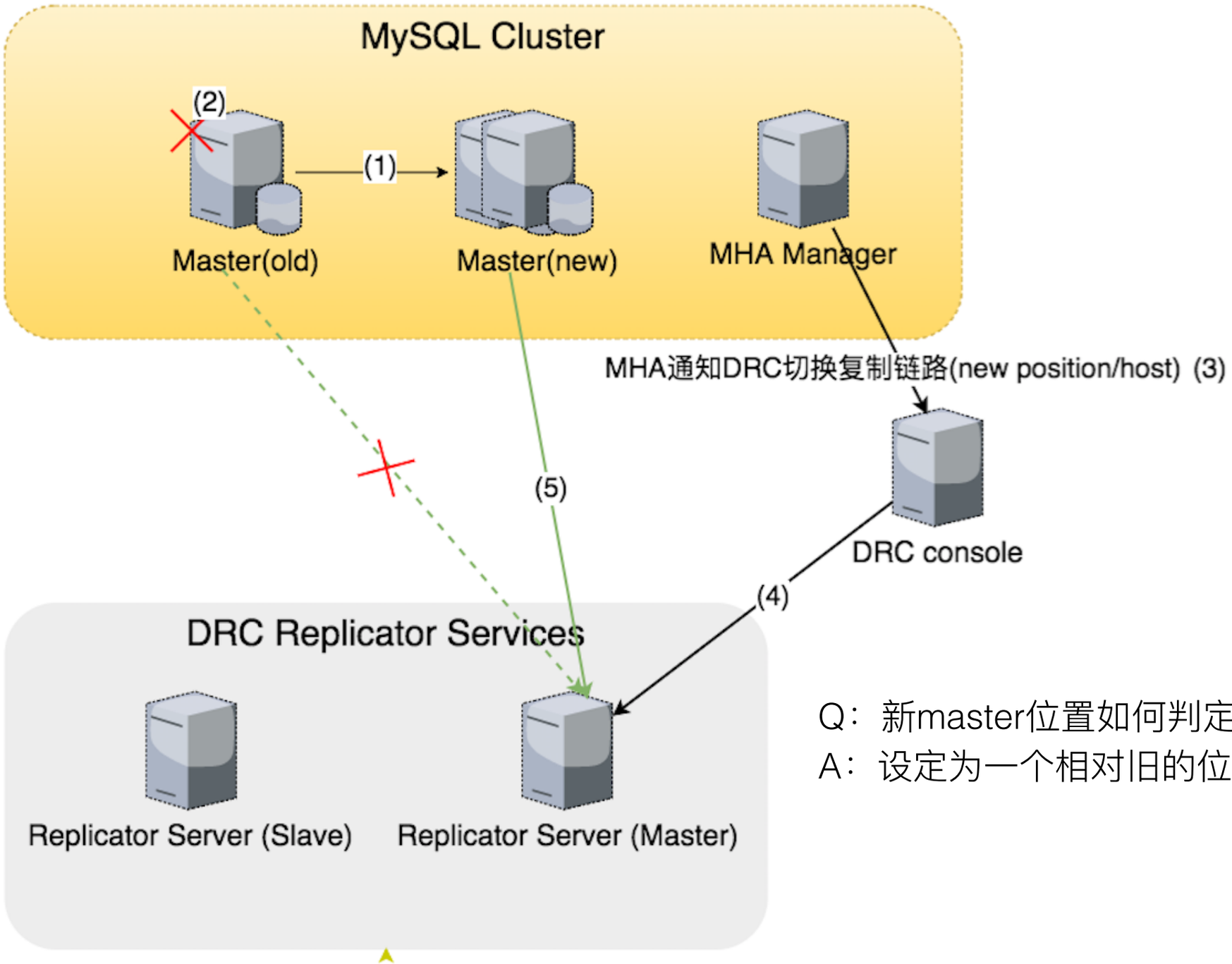
低延迟

- 大容量，高性能 RingBuffer
- 避免重复 Mysql 拉取
- 过滤重复 Event
- 压缩数据
- 传输时过滤，只传递必要的Event
- 压缩传输，减少网络开销

高吞吐

- 多通道并发写入
- 基于表 / 行的并发策略
- 合并事务 Group Commit
- 参数化执行优化

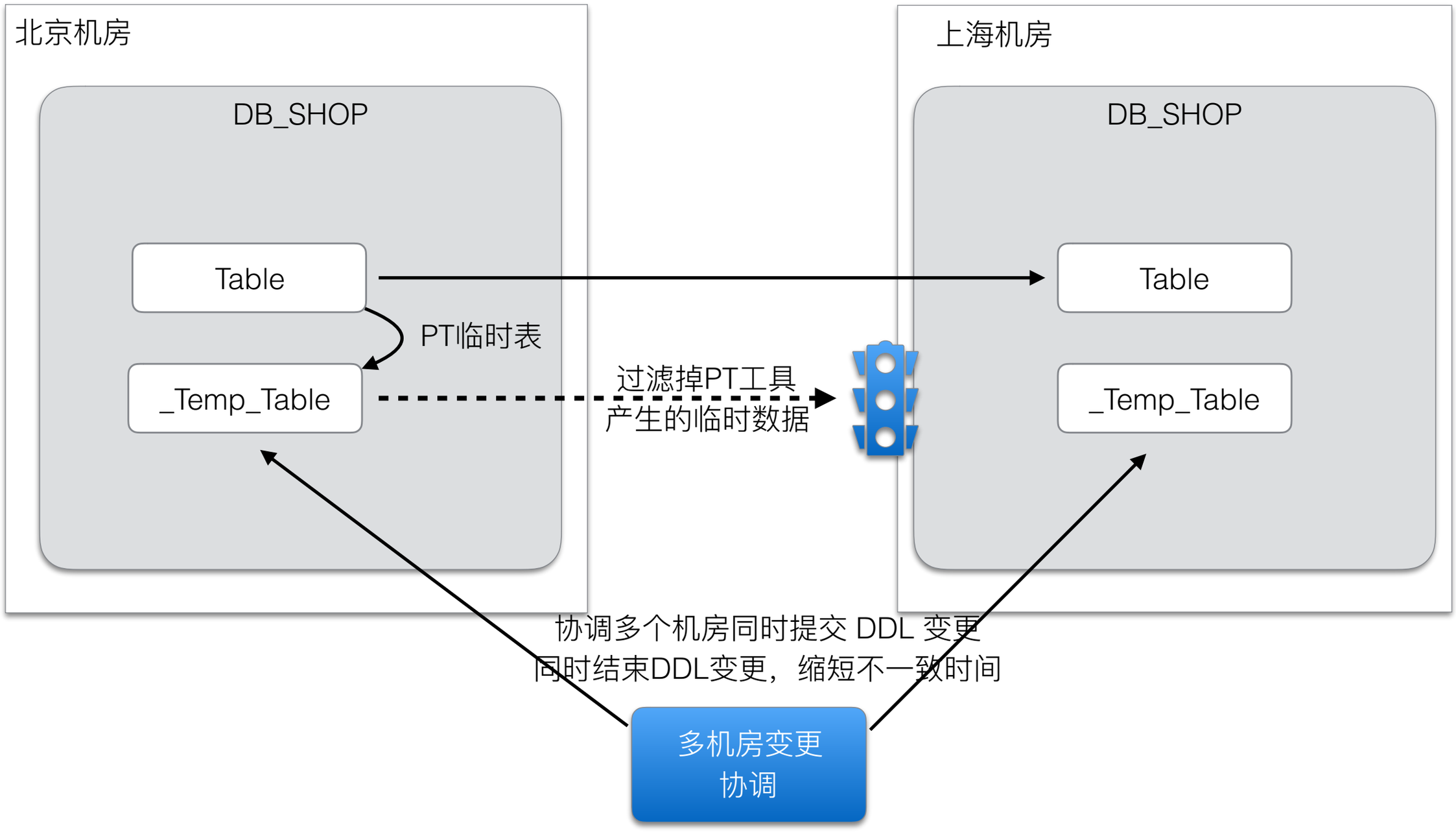
DRC & MySQL Master切换



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

DDL 变更改造



1. 大表DDL变更导致大量的 DB_EVENT，照成复制堵塞
2. DRC 和 DBA 协作，开发了自己的DDL跨机房协调工具，解决了这个问题

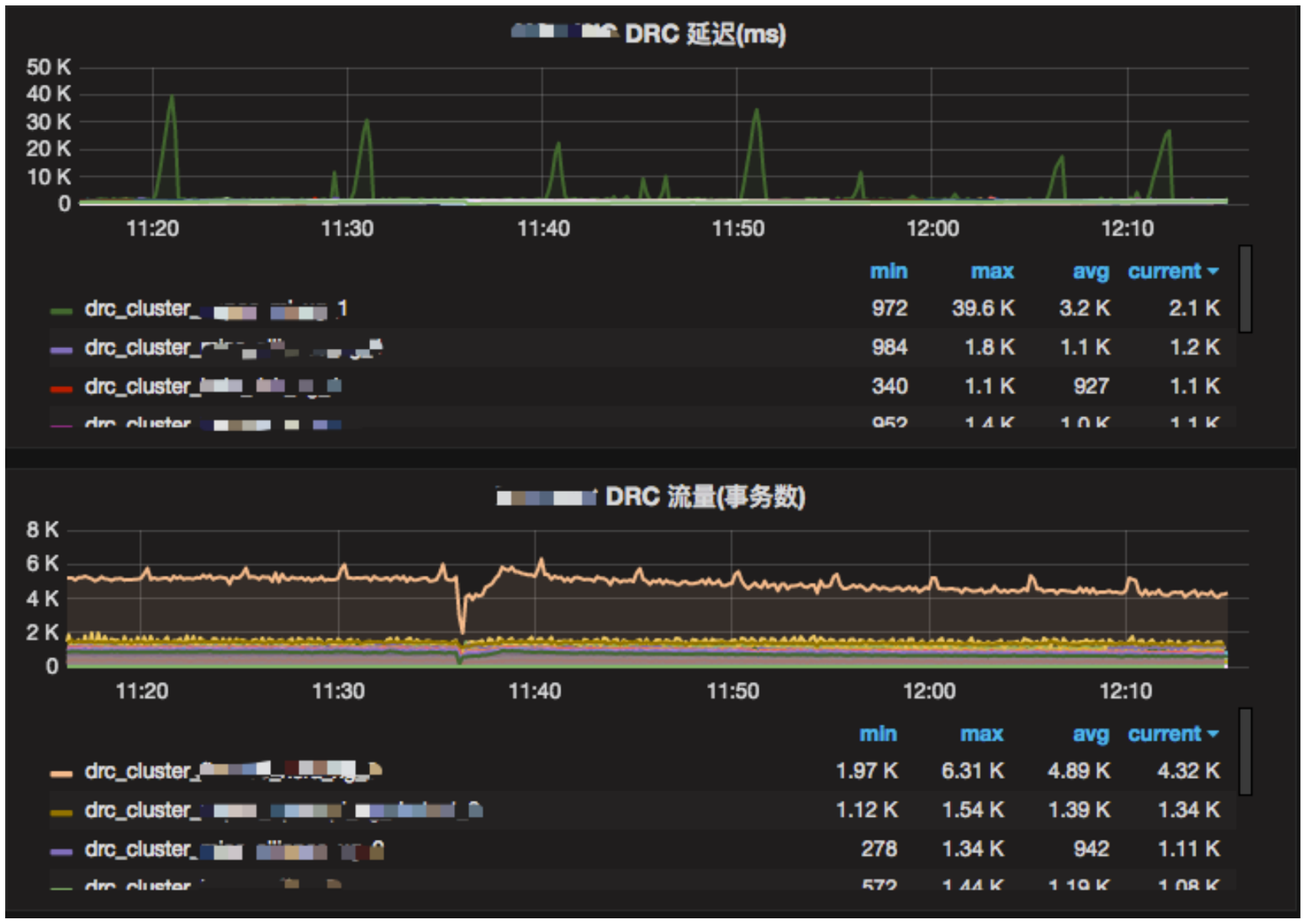
DRC线上运行状况（规模）

400+复制通道

消息订阅接入20+个业务方
1亿+条消息 / 天

所有业务的
跨机房cache刷新

DRC线上运行状况（性能）



支持 90% 饿了么数据流量

平均延时 < 1S，峰值吞吐 > 15K

谢谢！

email: shuangtao.li@ele.me