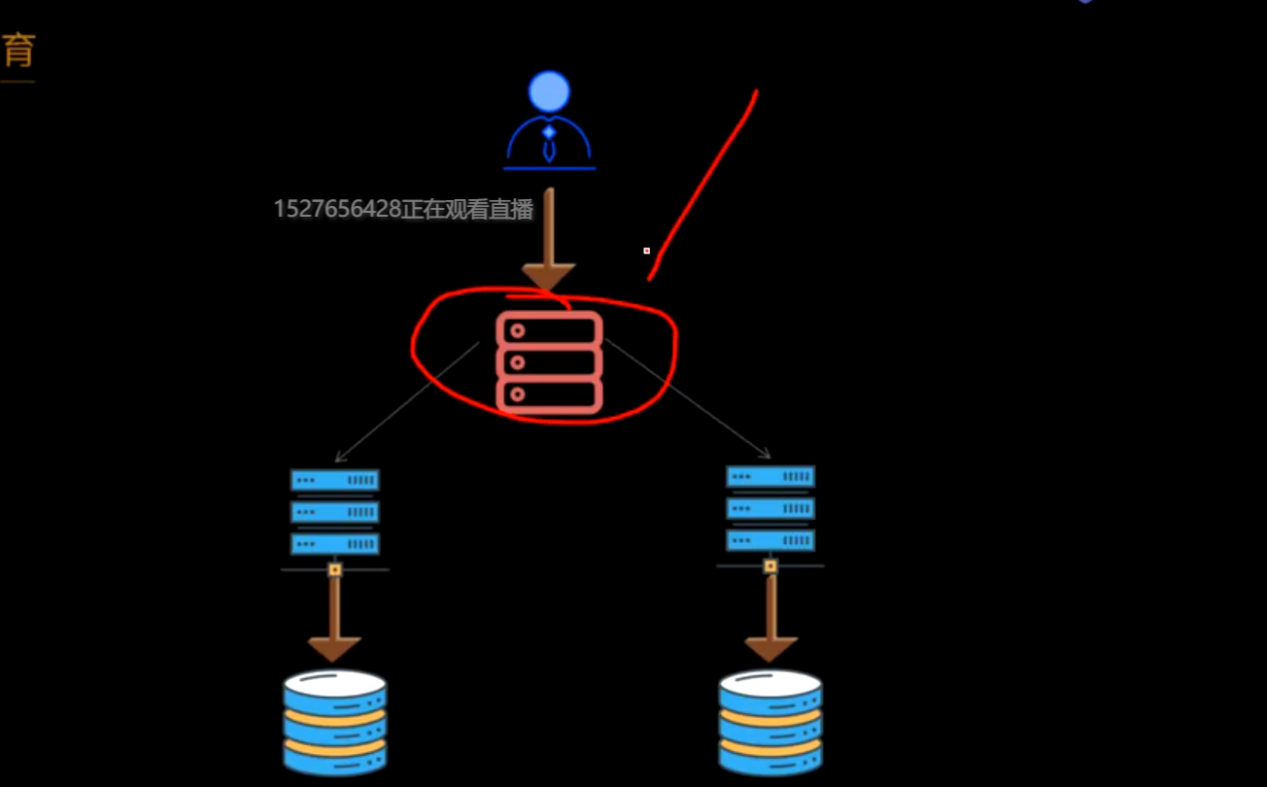
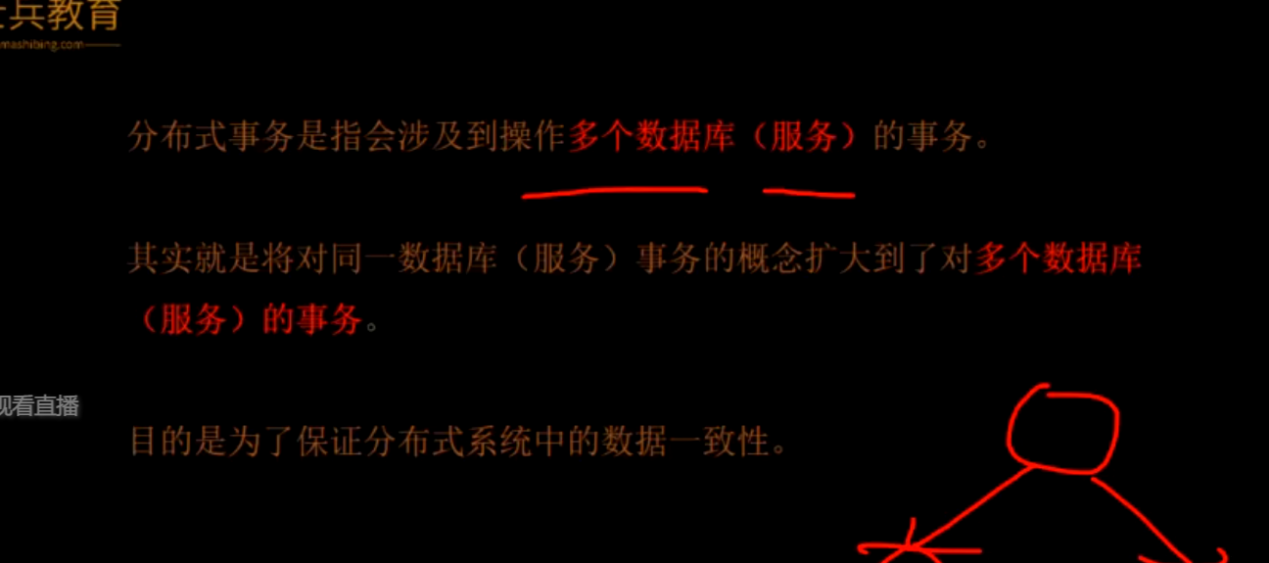
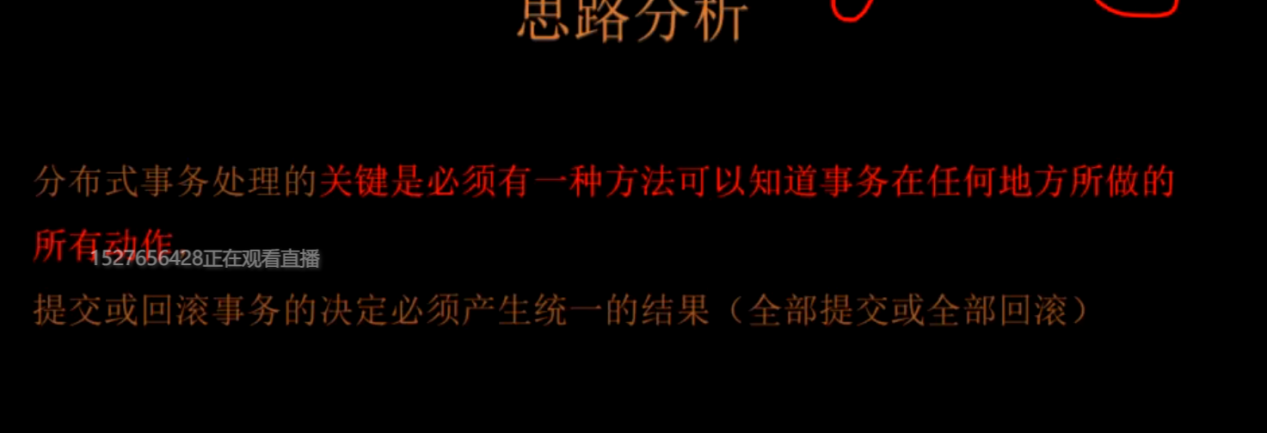
日志先行 单体的一致性

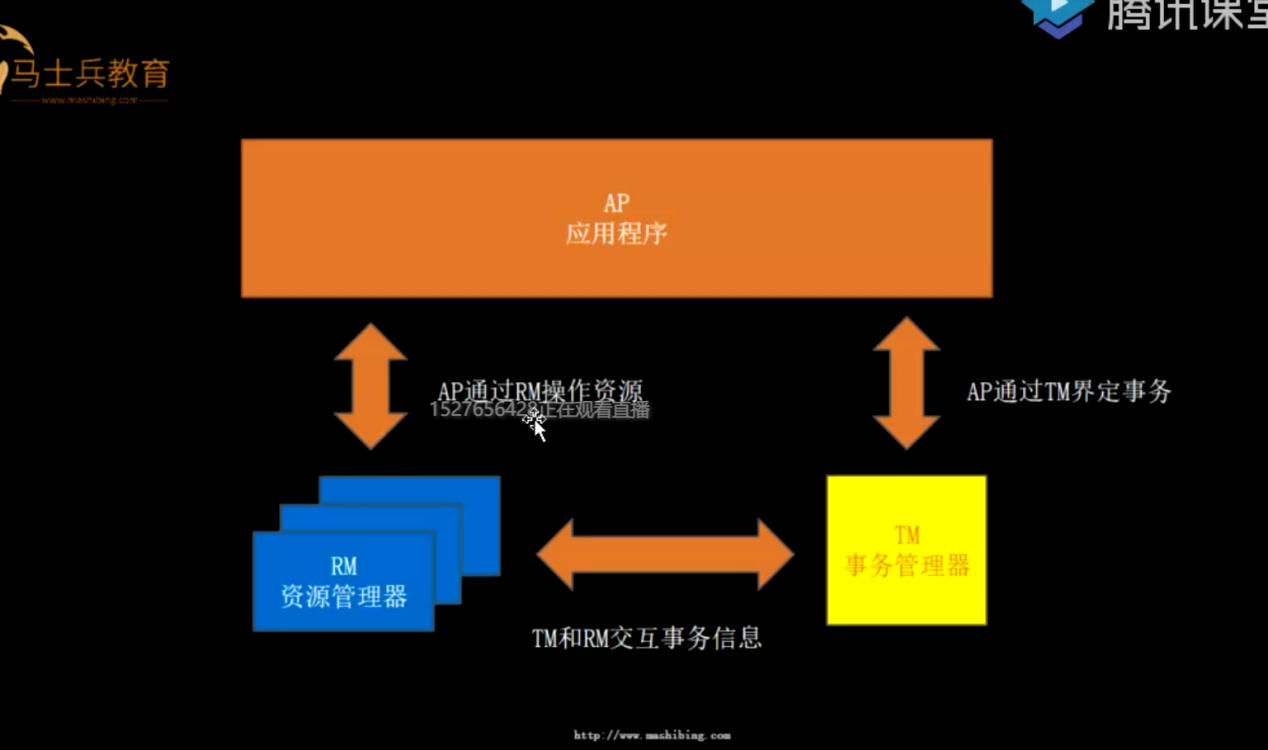


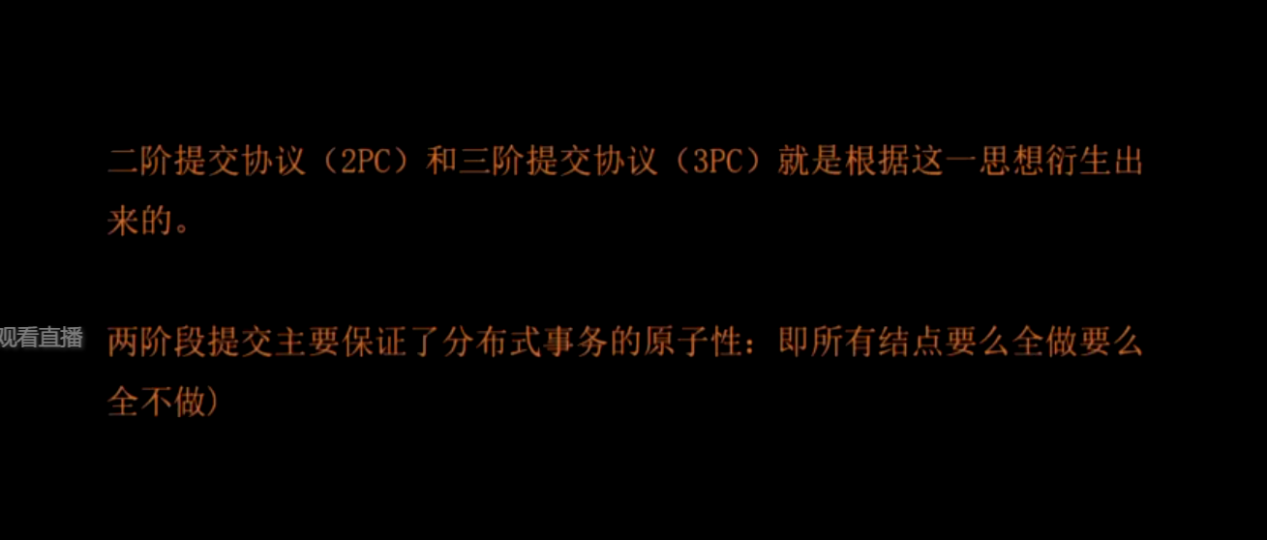
、

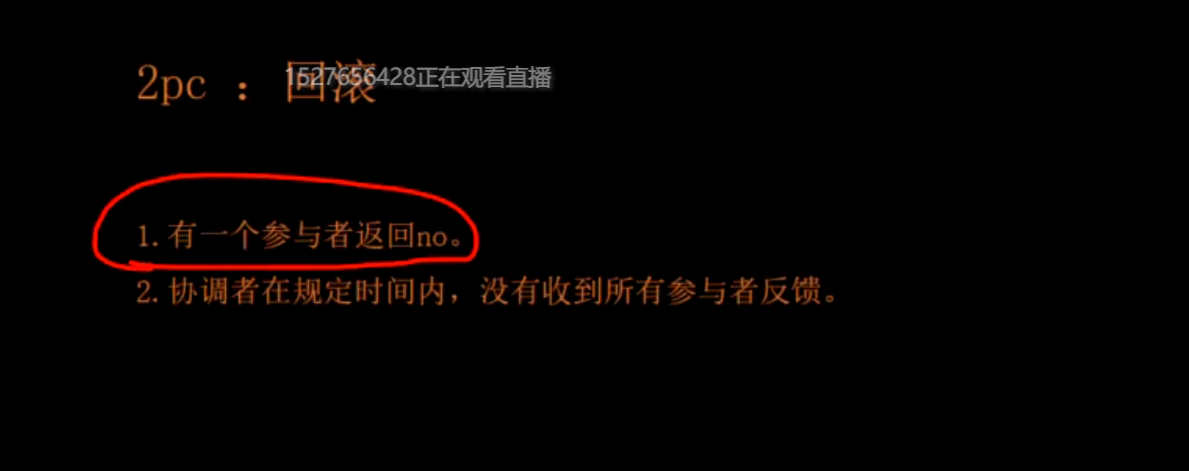


增加一个协调器











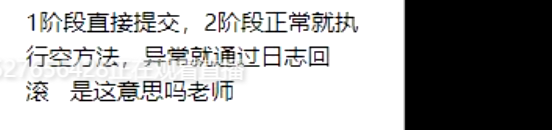
Seata 的at模式

多准备几个协调者 协调者是独立的服务

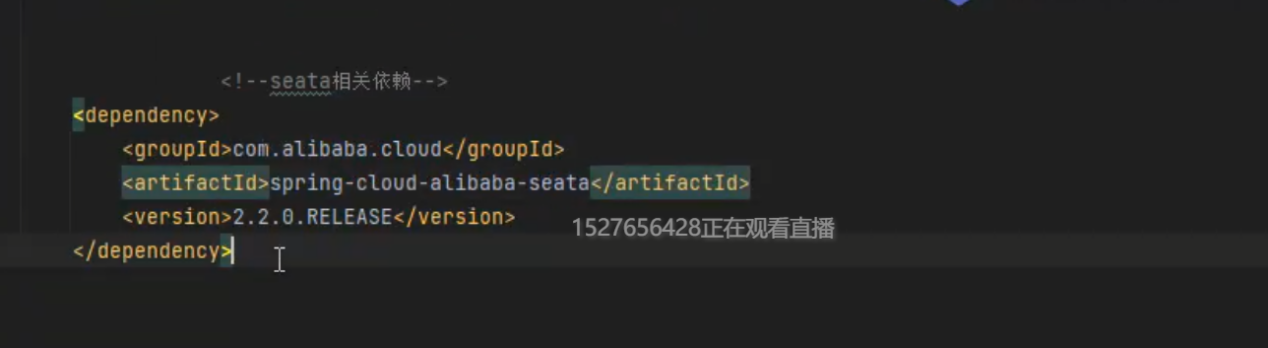
数据不一致 人工介入 脚本修改

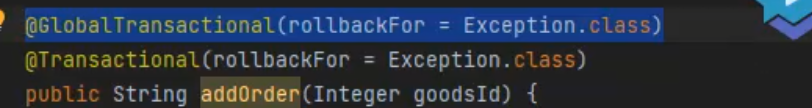
占用资源：阶段1直接调教 释放连接资源 镜像回滚 极端2

用undolog 进行回滚



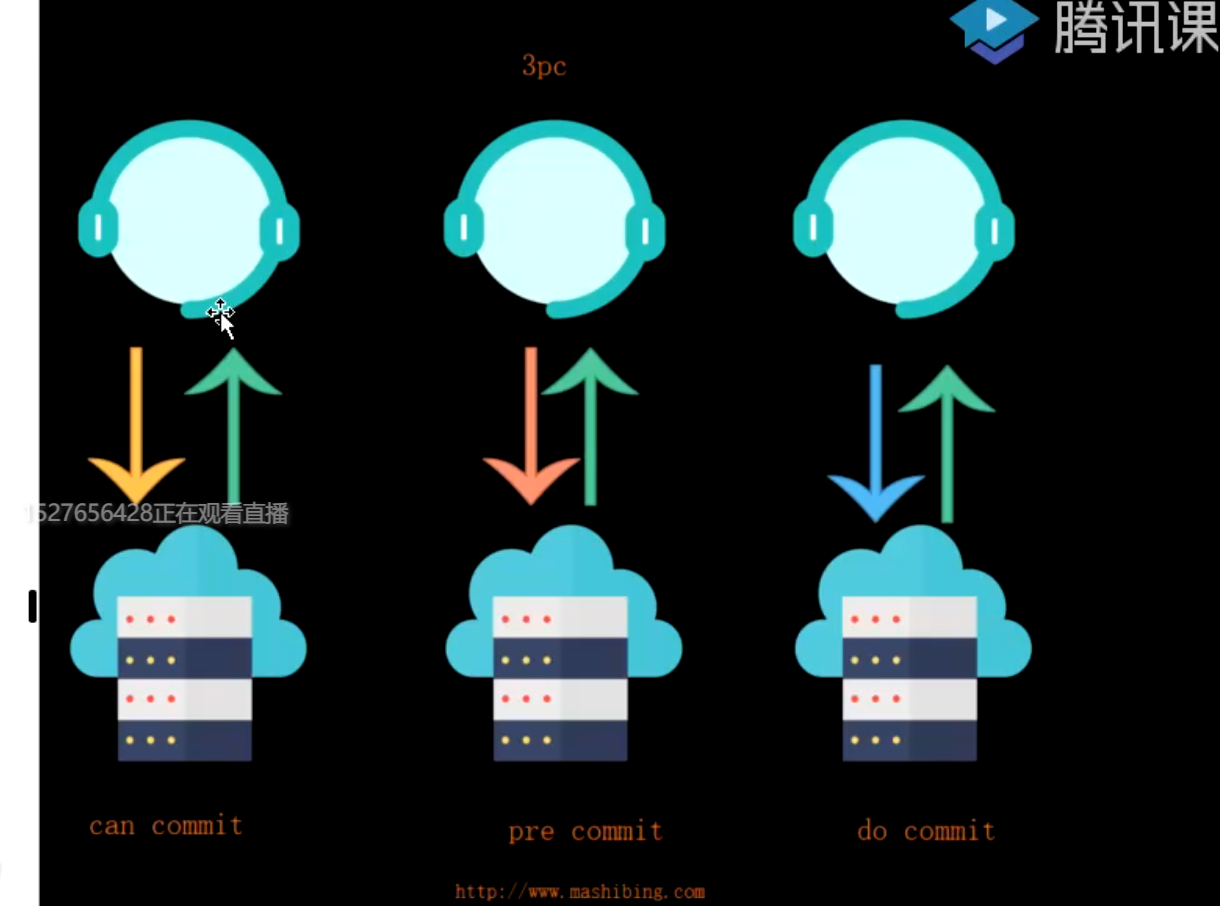
Seata server连接了两个库





三阶段

第三阶段 超时confirm 牧师 返回到男士这里 返回路上失败



Cap定理 追求一致性就牺牲了 可用性

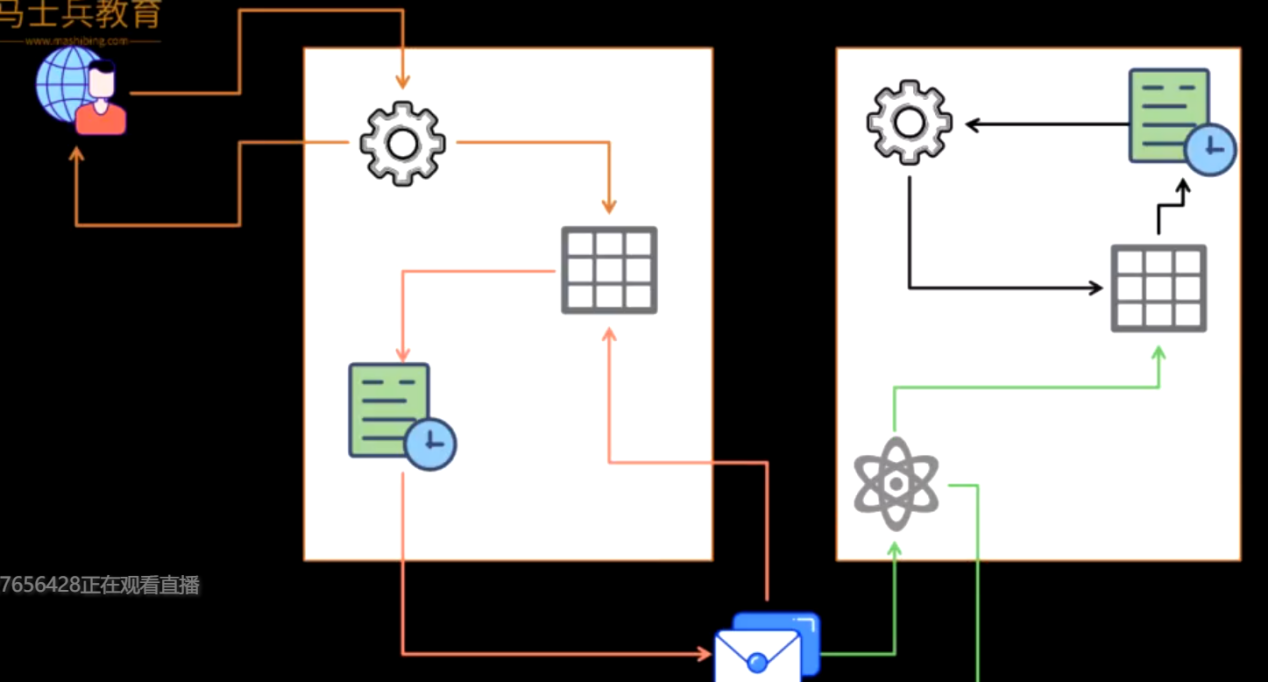
追求可用性的同时 zhui

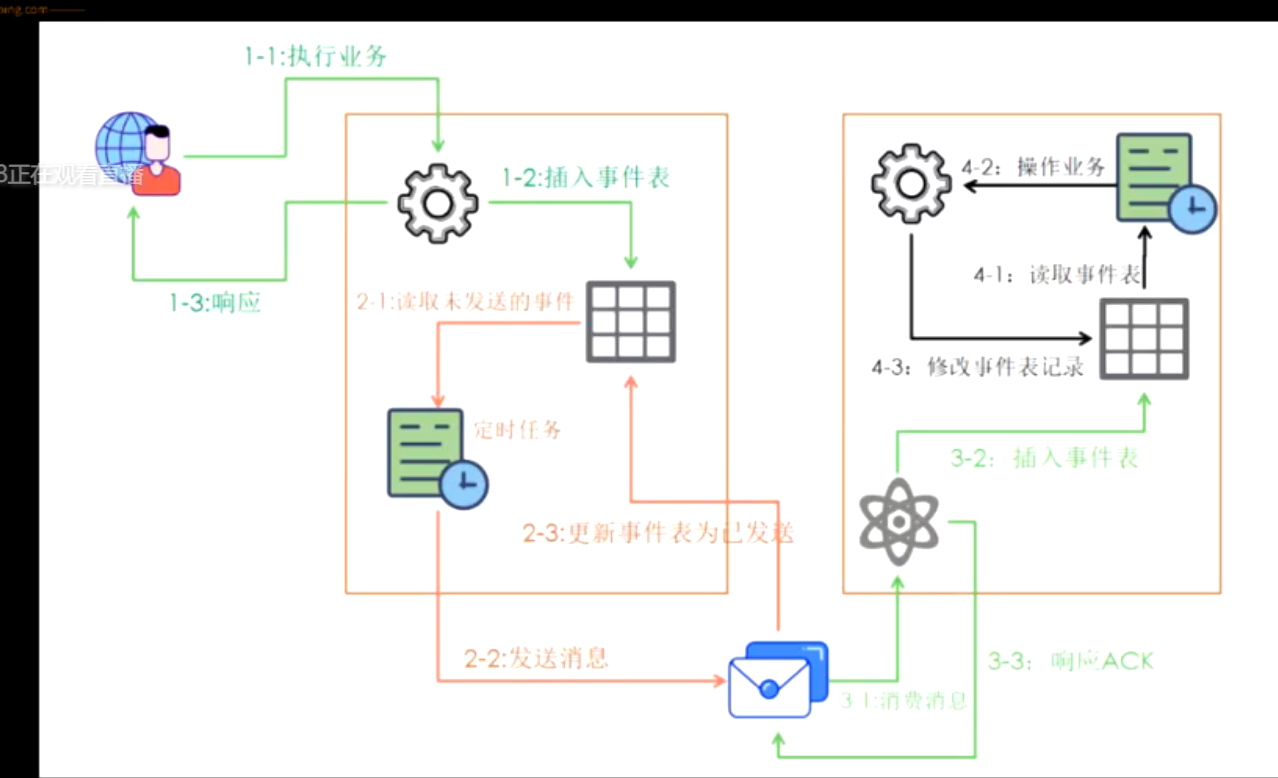
Tcc try confirm cancel

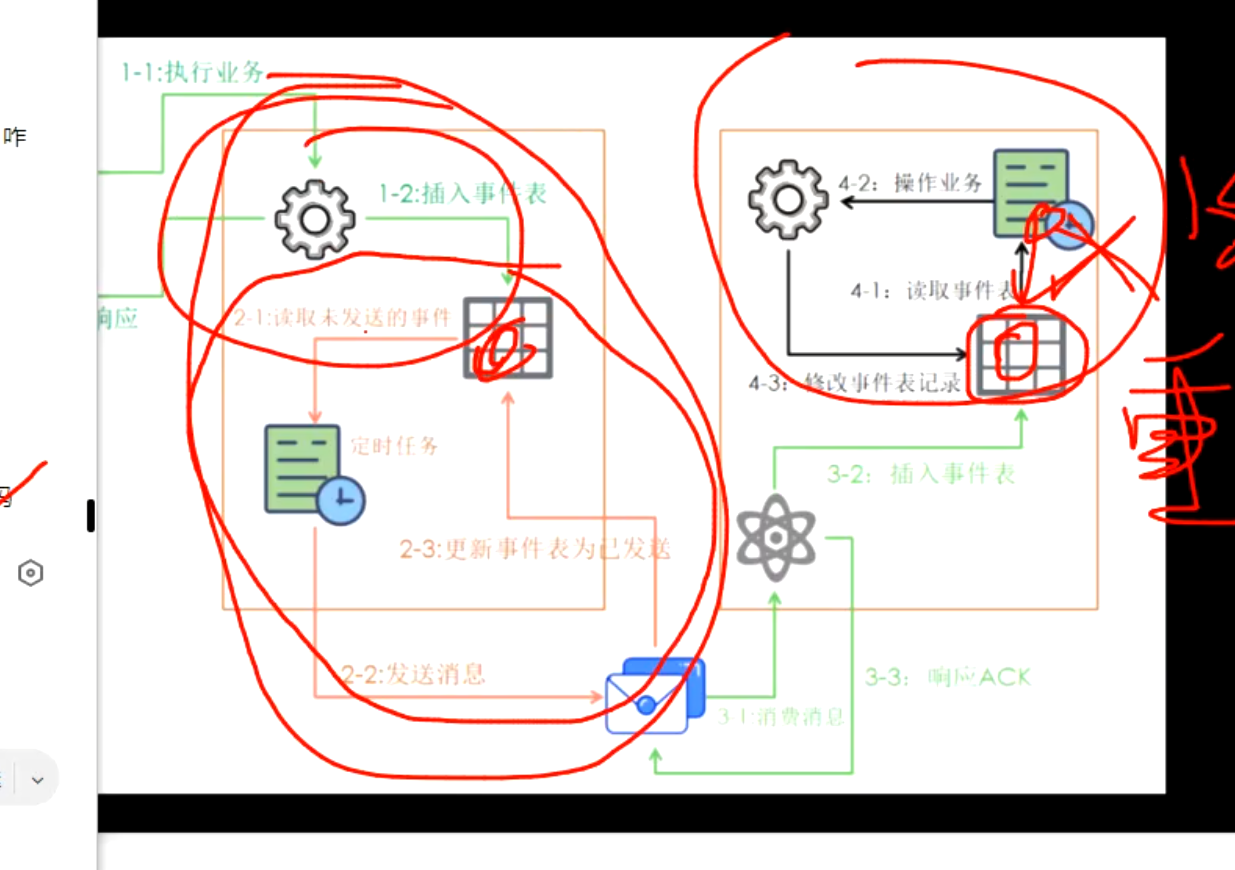
Redis mysql 使用tcc

t阶段 order inventor try

消息队列 加本地时间表

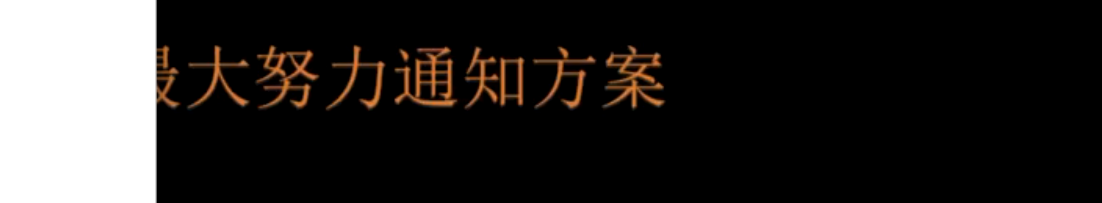




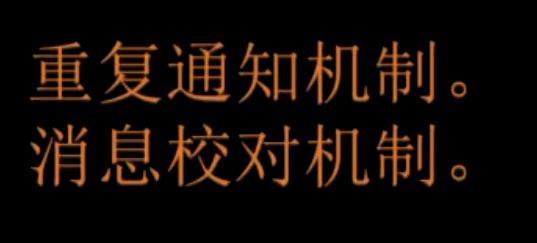


4 出错 人工介入

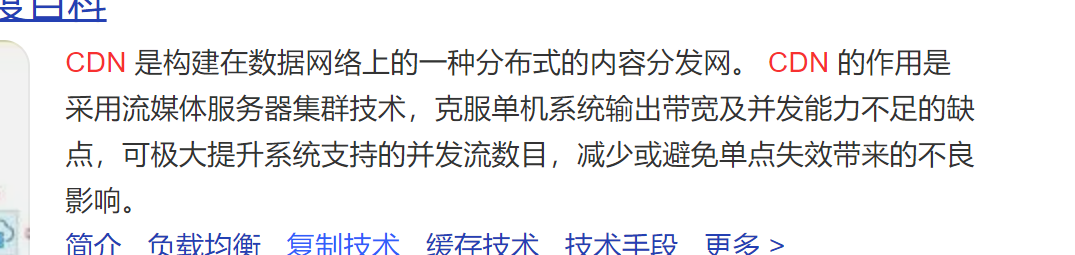
最大努力通知方案



不停通知 如果没有反应 就来查



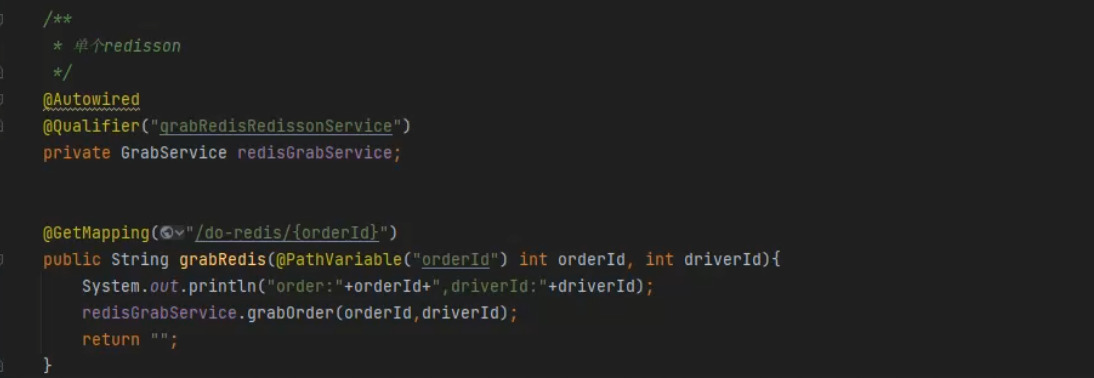
# 分布式 高并发 乙级流量

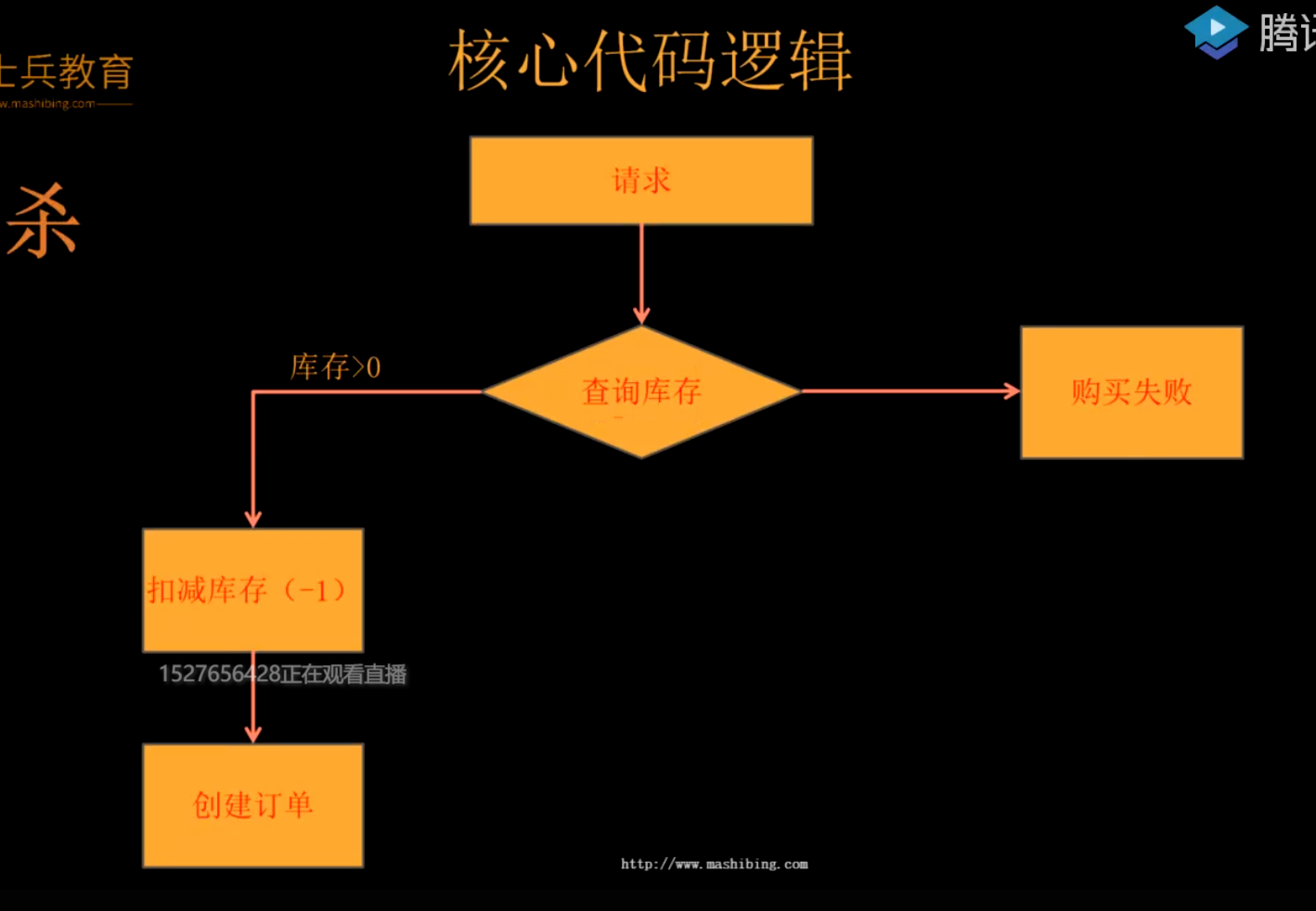
1. 客户端减少请求次数
2. Dns 分流 一个dns 对应多个ip address dns cname(域名到域名映射） 买别人的服务器 cdn 服务商 （chinacache)
3. 就近原则 地理位置 ip 判断
4. 

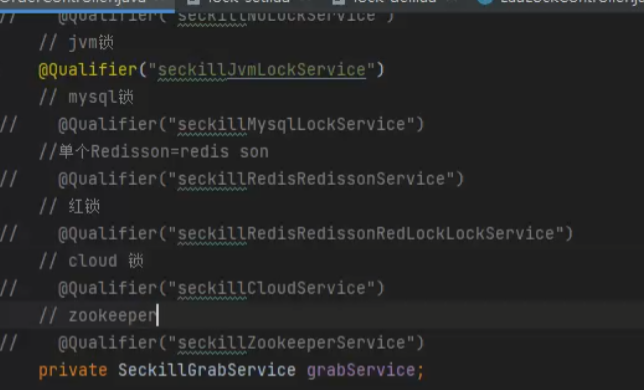
cdn根据用户来源，将用户进行分流, content deliver network

动态内容 静态内容 动静分离 动态进入到网关 黑白名单

1. 负载均衡
2. 缓存 ，队列MQ，削峰，
3. （不要改代码）数据库做分区 partition，不需要改代码 10行 两个存储文件一个表 一亿条数据， 两个文件存储
4. 分表--分库分表
5. 读写分离





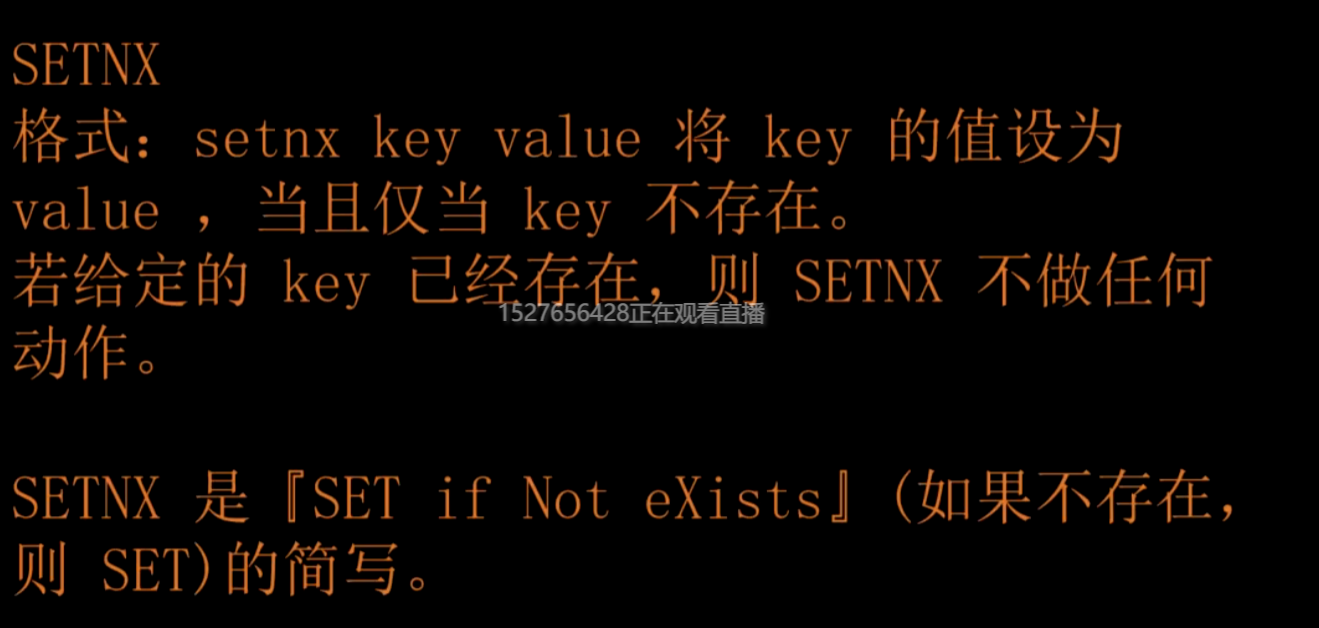


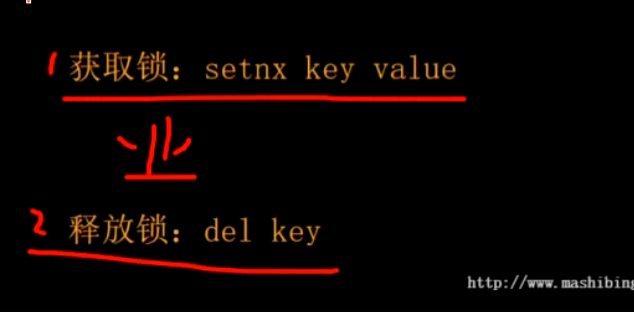
单体可用 集群就挂 高可用 多台服务 jvm 锁 数据不对

服务之间 线程不安全

需要在公共的部分 加锁 第三方锁(能存东西就行） 磁盘(主键 唯一索引） redis zookeeper







如果释放锁挂了 需要在中间设置过期时间 ex 原子命令

