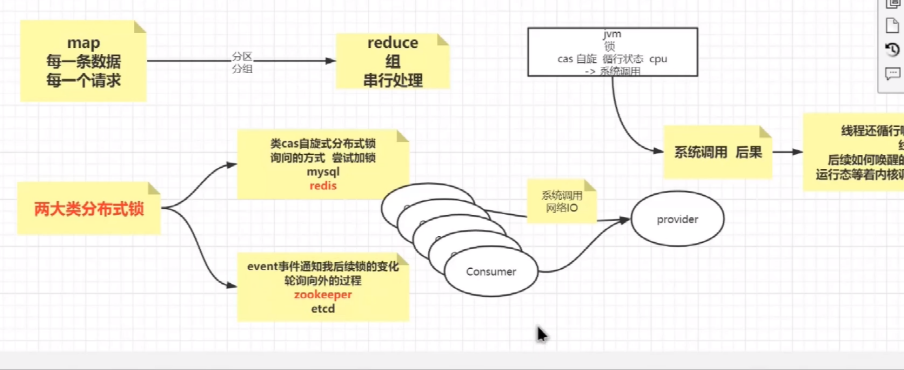
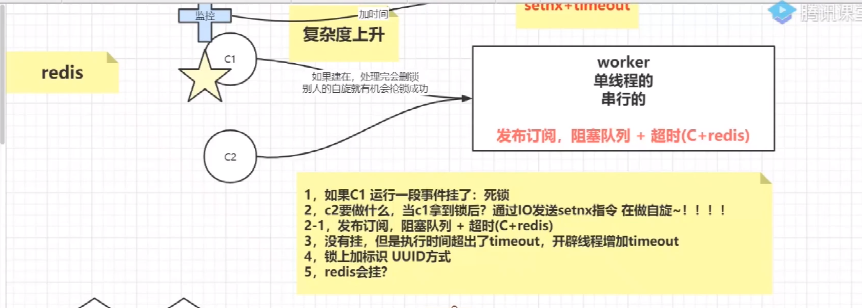
1. 两大分布式锁

单机情况：CAS可能比syn快一些，直接CPU轮询

分布式情况：由于分布式锁不在系统内部，是在第三方，是需要通过系统调用进行IO通信，如果在分布式情况下使用CAS，就可能会使得系统“疯狂”向第三方发生数据，就会使得第三方IO压力很大。但是只有一个请求可以得到锁，很多无效请求，造成网卡、交换机资源的浪费，所以性能没有比event时间好。



1. Redis为什么可以用做分布式锁？
2. Redis工作线程是单线程的，处理请求的时候是串行化处理，无需同步机制。
3. Redis的Vlaue有本地方法，获取锁只需要调用它的本地方法（常用setnx + timeout）。如果set成功，则表示抢占到锁。
4. 问题



1. 线程抢占到锁：

可能出现死锁。因为线程抢占到分布式锁之后，可能运行一段时间挂了，此时锁没有释放，成了死锁。整个业务就停宕了。

解决方法：可以在加锁的时候设置一个timeout过期时间，时间到了锁就消失。

1. 线程未抢占到锁：

可能出现资源浪费、IO吞吐压力变大。因为没有抢占到锁，会通过IO不断发送setnx指令做自旋，但是只有一个请求可以拿到锁，就会造成网卡资源浪费、IO吞吐压力变大。

解决方法：发布订阅（观察者模式）、阻塞队列、超时

1. 锁不一致性：

可能出现抢占到锁的线程未挂的时候锁失效，其他自旋抢占到锁，就会造成锁不一致。

解决方法：加一个监控线程，如果抢占到锁的线程还在执行没有任何异常，则监控线程需要在一定时间给分布式锁加时间。

1. 释放锁：

为了防止释放错锁，需要在锁上加标识（UUID方式）。

1. Redis宕机：

解决方法：Redis集群（主从复制、Cluster）、RedLock

1. Redis故障解决方法一：Redis集群

CAP：一致性、高可用性、分区容错性

哨兵：高可用下自动化运维的实现方式，它是做分布式协调工作，它发现主出故障之后，把从升为主。没有哨兵，人工运维也可以。

1. 主从复制

特点：追求AP，同步不精准，一致性不够。

场景：解决单点故障问题，利用哨兵进行主从切换。

1. cluster模式

场景：分治、分片，解决容量、压力、瓶颈问题。

1. 主从复制问题

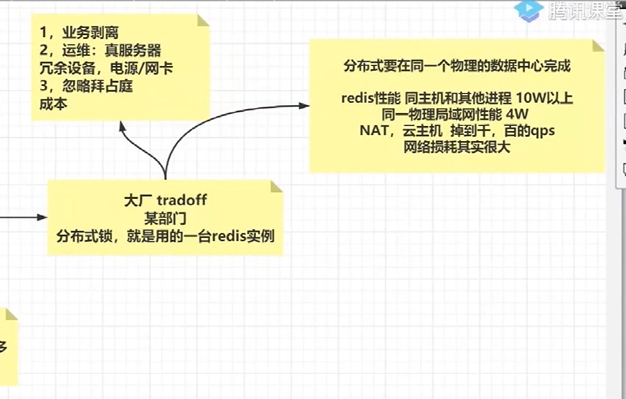
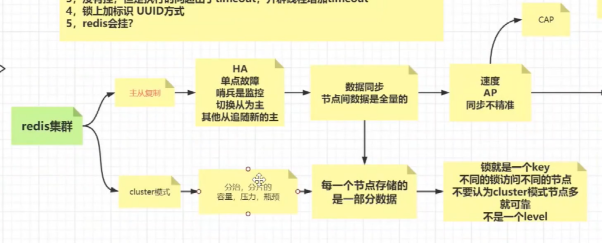
问题：一致性不够，线程抢占到锁之后，主出现故障，此时切换到从，而从没有这把锁，其他线程很有可能通过自旋获得锁，就出现锁并行执行。

解决方法：

1. 业务剥离：多部署Redis，用于缓存、限流、用户信息、抽奖等等，一个业务一个Redis，分布式锁也用一个Redis服务器
2. 运维：冗余设备，多个电源、网卡等等，防止服务器出故障。
3. 两者区别

主从复制数据丢失：主从之间数据是一样的，一个出现故障，其他没有出故障，那么数据不会丢失。

Cluster数据丢失：每个节点存储一部分数据，若某一个节点出故障后，那么它的数据就会丢失。



1. Redis故障解决方法二：redlock
2. 原理

redlock是客户端实现的，不是Redis实现的，获得一个Redis锁则计数加1。如果线程里面的锁计数只要超过一半，那么此线程就可以获得锁，并且还要设置timeout等（问题1-4还是会出现）。如果没有则进行随机尝试获取锁。

通过多台Redis，降低故障的影响。只要出现故障得Redis数过半，就不会有问题。