以下内容来源于QCon某高可用架构群聊天记录整理 背景：有某个朋友咨询微信红包的架构，在官方或非官方同学的解释和讨论中得出以下讨论内容，在此期间有多个同学发红包做现网算法测试。

抢红包过程

当有人在群里发了一个N人的红包，总金额M元，后台大概发生的事情如下：

一、发红包后台操作：

1. 在数据库中增加一条红包记录，存储到CKV，设置过期时间；
2. 在Cache（可能是腾讯内部kv数据库，基于内存，有落地，有内核态网络处理模块，以内核模块形式提供服务））中增加一条记录，存储抢红包的人数N

二、抢红包后台操作：

1. 抢红包分为抢和拆，抢操作在Cache层完成，通过原子减操作进行红包数递减，到0就说明抢光了，最终实际进入后台拆操作的量不大，通过操作的分离将无效请求直接挡在Cache层外面。这里的原子减操作并不是真正意义上的原子减操作，是其Cache层提供的CAS，通过比较版本号不断尝试，存在一定程度上的冲突，冲突的用户会放行，让其进入下一步拆的操作，这也解释了为啥有用户抢到了拆开发现领完了的情况。
2. 拆红包在数据库完成，通过数据库的事务操作累加已经领取的个数和金额，插入一条领取流水，入账为异步操作，这也解释了为啥在春节期间红包领取后在余额中看不到。拆的时候会实时计算金额，其金额为1分到剩余平均值2倍之间随机数，一个总金额为M元的红包，最大的红包为 M * 2 /N（且不会超过M），当拆了红包后会更新剩余金额和个数。财付通按20万笔每秒入账准备，实际只到8万每秒。

FAQ

1. 既然在抢的时候有原子减了就不应该出现抢到了拆开没有的情况？
   这里的原子减并不是真正意义上的原子操作，是Cache层提供的CAS，通过比较版本号不断尝试。
2. cache和db挂了怎么办？
   主备 +对账
3. 有没有红包个数没了，但余额还有情况？
   没有，程序最后会有一个take all操作以及一个异步对账保障。
4. 为什么要分离抢和拆？
   总思路是设置多层过滤网，层层筛选，层层减少流量和压力。这个设计最初是因为抢操作是业务层，拆是入账操作，一个操作太重了，而且中断率高。 从接口层面看，第一个接口纯缓存操作，搞压能力强，一个简单查询Cache挡住了绝大部分用户，做了第一

道筛选，所以大部分人会看到已经抢完了的提示。

5. 抢到红包后再发红包或者提现，这里有什么策略吗？

   大额优先入账策略

6. 有没有从数据上证明每个红包的概率是不是均等？

   不是绝对均等，就是一个简单的拍脑袋算法。

7. 拍脑袋算法，会不会出现两个最佳？

   会出现金额一样的，但是手气最佳只有一个，先抢到的那个最佳。

8. 发红包人的钱会不会冻结？

   是直接实时扣掉，不是冻结。

9. 采用实时算出金额是出于什么考虑？

   实时效率更高，预算才效率低下。预算还要占额外存储。因为红包只占一条记录而且有效期就几天，所以不需要多大空间。就算压力大时，水平扩展机器是。

> 微信红包的算法实现探讨（基于PHP）