微信高并发资金交易系统设计方案——百亿红包背后的技术支撑

原创 2017-02-15 方乐明 InfoQ

作者｜方乐明
编辑｜尾尾

每年节假日，微信红包的收发数量都会暴涨，尤以除夕为最。如此大规模、高峰值的业务需要，背后需要怎样的技术支撑？百亿级别的红包规模，如何保证并发性能与资金安全？

背景介绍

2017年1月28日，正月初一，微信公布了用户在除夕当天收发微信红包的数量——142亿个，而其收发峰值也已达到76万每秒。百亿级别的红包，如何保障并发性能与资金安全？这给微信带来了超级挑战。面对挑战，微信红包在分析了业界“秒杀”系统解决方案的基础上，采用了SET化、请求排队串行化、双维度分库表等设计，形成了独特的高并发、资金安全系统解决方案。实践证明，该方案表现稳定，且实现了除夕夜系统零故障运行。

本文将为读者介绍百亿级别红包背后的系统高并发设计方案，包括微信红包的两大业务特点、微信红包系统的技术难点、解决高并发问题通常使用的方案，以及微信红包系统的高并发解决方案。

微信红包的两大业务特点

微信红包（尤其是发在微信群里的红包，即群红包）业务形态上很类似网上的普通商品“秒杀”活动。

用户在微信群里发一个红包，等同于是普通商品“秒杀”活动的商品上架；微信群里的所有用户抢红包的动作，等同于“秒杀”活动中的查询库存；用户抢到红包后拆红包的动作，则对应“秒杀”活动中用户的“秒杀”动作。

不过除了上面的相同点之外，微信红包在业务形态上与普通商品“秒杀”活动相比，还具备自身的特点：

首先，微信红包业务比普通商品“秒杀”有更海量的并发要求。

微信红包用户在微信群里发一个红包，等同于在网上发布一次商品“秒杀”活动。假设同一时间有10万个群里的用户同时在发红包，那就相当于同一时间有10万个“秒杀”活动发布出去。10万个微信群里的用户同时抢红包，将产生海量的并发请求。

其次，微信红包业务要求更严格的安全级别。

微信红包业务本质上是资金交易。微信红包是微信支付的一个商户，提供资金流转服务。

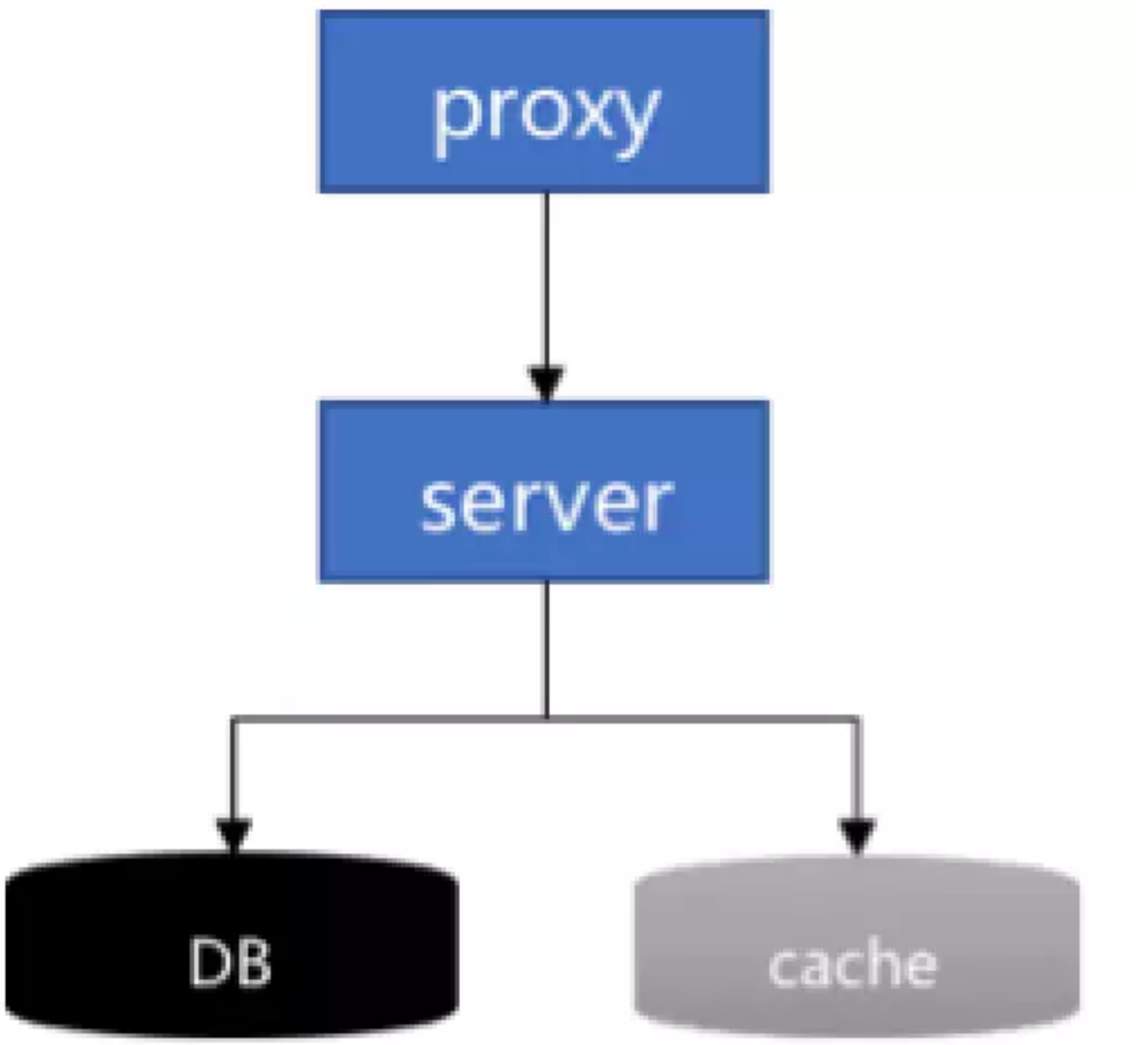
用户发红包时，相当于在微信红包这个商户上使用微信支付购买一笔“钱”，并且收货地址是微信群。当用户支付成功后，红包“发货”到微信群里，群里的用户拆开红包后，微信红包提供了将“钱”转入折红包用户微信零钱的服务。

资金交易业务比普通商品“秒杀”活动有更高的安全级别要求。普通的商品“秒杀”商品由商户提供，库存是商户预设的，“秒杀”时可以允许存在“超卖”（即实际被抢的商品数量比计划的库存多）、“少卖”（即实际被抢的商户数量比计划的库存少）的情况。但是对于微信红包，用户发100元的红包绝对不可以被拆出101元；用户发100元只被领取99元时，剩下的1元在24小时过期后要精确地退还给发红包用户，不能多也不能少。

以上是微信红包业务模型上的两大特点。

微信红包系统的技术难点

在介绍微信红包系统的技术难点之前，先介绍下简单的、典型的商品“秒杀”系统的架构设计，如下图所示。



该系统由接入层、逻辑服务层、存储层与缓存构成。Proxy处理请求接入，Server承载主要的业务逻辑，Cache用于缓存库存数量、DB则用于数据持久化。

一个“秒杀”活动，对应DB中的一条库存记录。当用户进行商品“秒杀”时，系统的主要逻辑在于DB中库存的操作上。一般来说，对DB的操作流程有以下三步：

1. 锁库存

2. 插入“秒杀”记录
3. 更新库存

其中，锁库存是为了避免并发请求时出现“超卖”情况。同时要求这三步操作需要在一个事务中完成（所谓的事务，是指作为单个逻辑工作单元执行的一系列操作，要么完全地执行，要么完全地不执行）。

“秒杀”系统的设计难点就在这个事务操作上。商品库存在DB中记为一行，大量用户同时“秒杀”同一商品时，第一个到达DB的请求锁住了这行库存记录。在第一个事务完成提交之前这个锁一直被第一个请求占用，后面的所有请求需要排队等待。同时参与“秒杀”的用户越多，并发进DB的请求越多，请求排队越严重。因此，并发请求抢锁，是典型的商品“秒杀”系统的设计难点。

微信红包业务相比普通商品“秒杀”活动，具有海量并发、高安全级别要求的特点。在微信红包系统的设计上，除了并发请求抢锁之外，还有以下两个突出难点：

首先，事务级操作量级大。上文介绍微信红包业务特点时提到，普遍情况下同时会有数以万计的微信群在发红包。这个业务特点映射到微信红包系统设计上，就是有数以万计的“并发请求抢锁”同时在进行。这使得DB的压力比普通单个商品“库存”被锁要大很多倍。

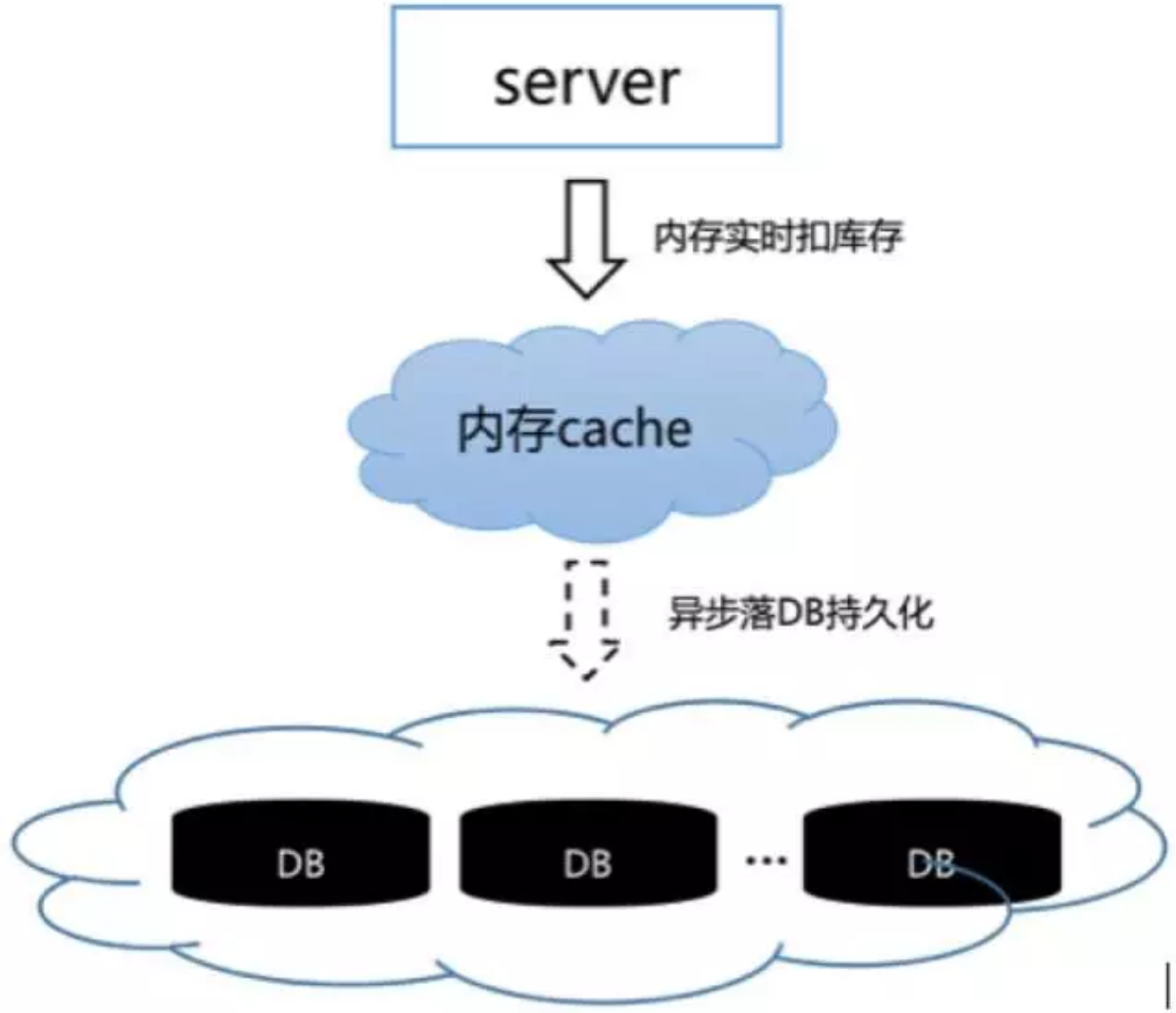
其次，事务性要求严格。微信红包系统本质上是一个资金交易系统，相比普通商品“秒杀”系统有更高的事务级别要求。

解决高并发问题常用方案

普通商品“秒杀”活动系统，解决高并发问题的方案，大体有以下几种：

方案一，使用内存操作替代实时的DB事务操作。

如图2所示，将“实时扣库存”的行为上移到内存Cache中操作，内存Cache操作成功直接给Server返回成功，然后异步落DB持久化。



这个方案的优点是用内存操作替代磁盘操作，提高了并发性能。

但是缺点也很明显，在内存操作成功但DB持久化失败，或者内存Cache故障的情况下，DB持久化会丢数据，不适合微信红包这种资金交易系统。

方案二，使用乐观锁替代悲观锁。

所谓悲观锁，是关系数据库管理系统里的一种并发控制的方法。它可以阻止一个事务以影响其他用户的方式来修改数据。如果一个事务执行的操作对某行数据应用了锁，那只有当这个事务把锁释放，其他事务才能够执行与该锁冲突的操作。对应于上文分析中的“并发请求抢锁”行为。

所谓乐观锁，它假设多用户并发的事务在处理时不会彼此互相影响，各事务能够在不产生锁的情况下处理各自影响的那部分数据。在提交数据更新之前，每个事务会先检查在该事务读取数据后，有没有其他事务又修改了该数据。如果其他事务有更新的话，正在提交的事务会进行回滚。

商品“秒杀”系统中，乐观锁的具体应用方法，是在DB的“库存”记录中维护一个版本号。在更新“库存”的操作进行前，先去DB获取当前版本号。在更新库存的事务提交时，检查该版本号是否已被其他事务修改。如果版本没被修改，则提交事务，且版本号加1；如果版本号已经被其他事务修改，则回滚事务，并给上层报错。

这个方案解决了“并发请求抢锁”的问题，可以提高DB的并发处理能力。

但是如果应用于微信红包系统，则会存在下面三个问题：

1. 如果拆红包采用乐观锁，那么在并发抢到相同版本号的拆红包请求中，只有一个能拆红包成功，其他的请求将事务回滚并返回失败，给用户报错，用户体验完全不可接受。
2. 如果采用乐观锁，将会导致第一时间同时拆红包的用户有一部分直接返回失败，反而那些“手慢”的用户，有可能因为并发减小后拆红包成功，这会带来用户体验上的负面影响。
3. 如果采用乐观锁的方式，会带来大数量的无效更新请求、事务回滚，给DB造成不必要的额外压力。

基于以上原因，微信红包系统不能采用乐观锁的方式解决并发抢锁问题。

微信红包系统的高并发解决方案

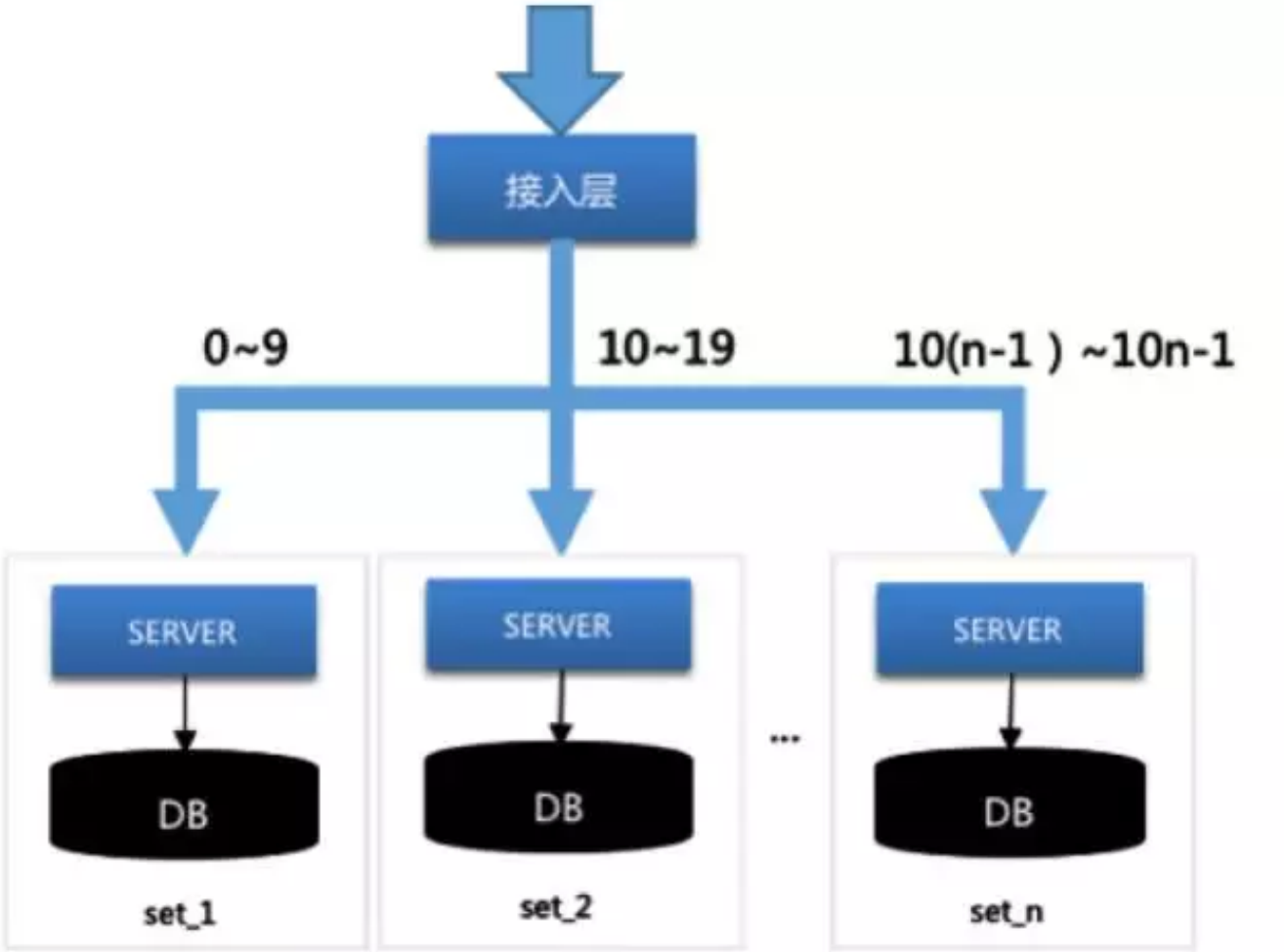
综合上面的分析，微信红包系统针对相应的技术难点，采用了下面几个方案，解决高并发问题。

1.系统垂直SET化，分而治之。

微信红包用户发一个红包时，微信红包系统生成一个ID作为这个红包的唯一标识。接下来这个红包的所有发红包、抢红包、拆红包、查询红包详情等操作，都根据这个ID关联。

红包系统根据这个红包ID，按一定的规则（如按ID尾号取模等），垂直上下切分。切分后，一个垂直链条上的逻辑Server服务器、DB统称为一个SET。

各个SET之间相互独立，互相解耦。并且同一个红包ID的所有请求，包括发红包、抢红包、拆红包、查详情详情等，垂直stick到同一个SET内处理，高度内聚。通过这样的方式，系统将所有红包请求这个巨大的洪流分散为多股小流，互不影响，分而治之，如下图所示。



这个方案解决了同时存在海量事务级操作的问题，将海量化为小量。

2.逻辑Server层将请求排队，解决DB并发问题。

红包系统是资金交易系统，DB操作的事务性无法避免，所以会存在“并发抢锁”问题。但是如果到达DB的事务操作（也即拆红包行为）不是并发的，而是串行的，就不会存在“并发抢锁”的问题了。

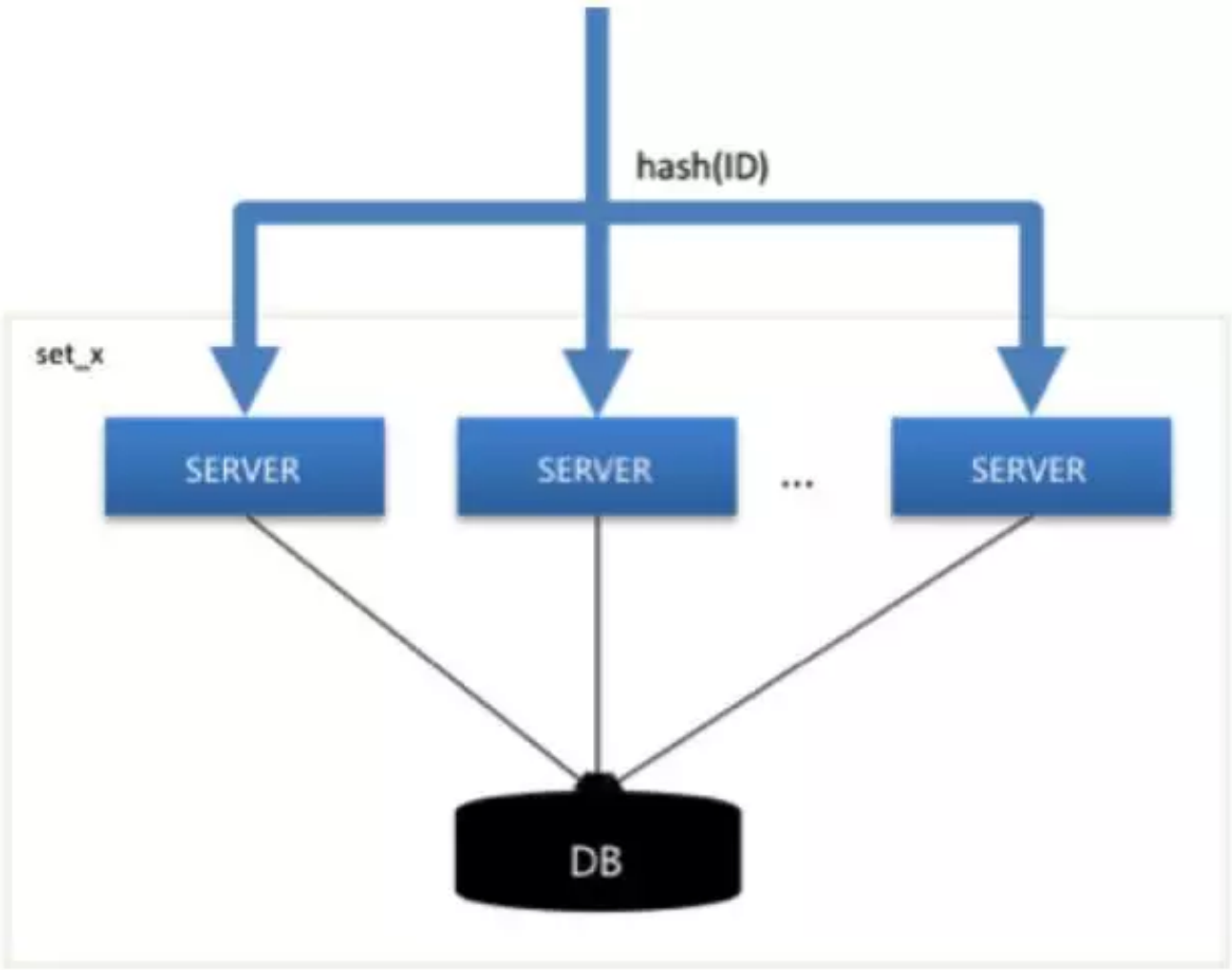
按这个思路，为了使拆红包的事务操作串行地进入DB，只需要将请求在Server层以FIFO（先进先出）的方式排队，就可以达到这个效果。从而问题就集中到Server的FIFO队列设计上。

微信红包系统设计了分布式的、轻巧的、灵活的FIFO队列方案。其具体实现如下：

首先，将同一个红包ID的所有请求stick到同一台Server。

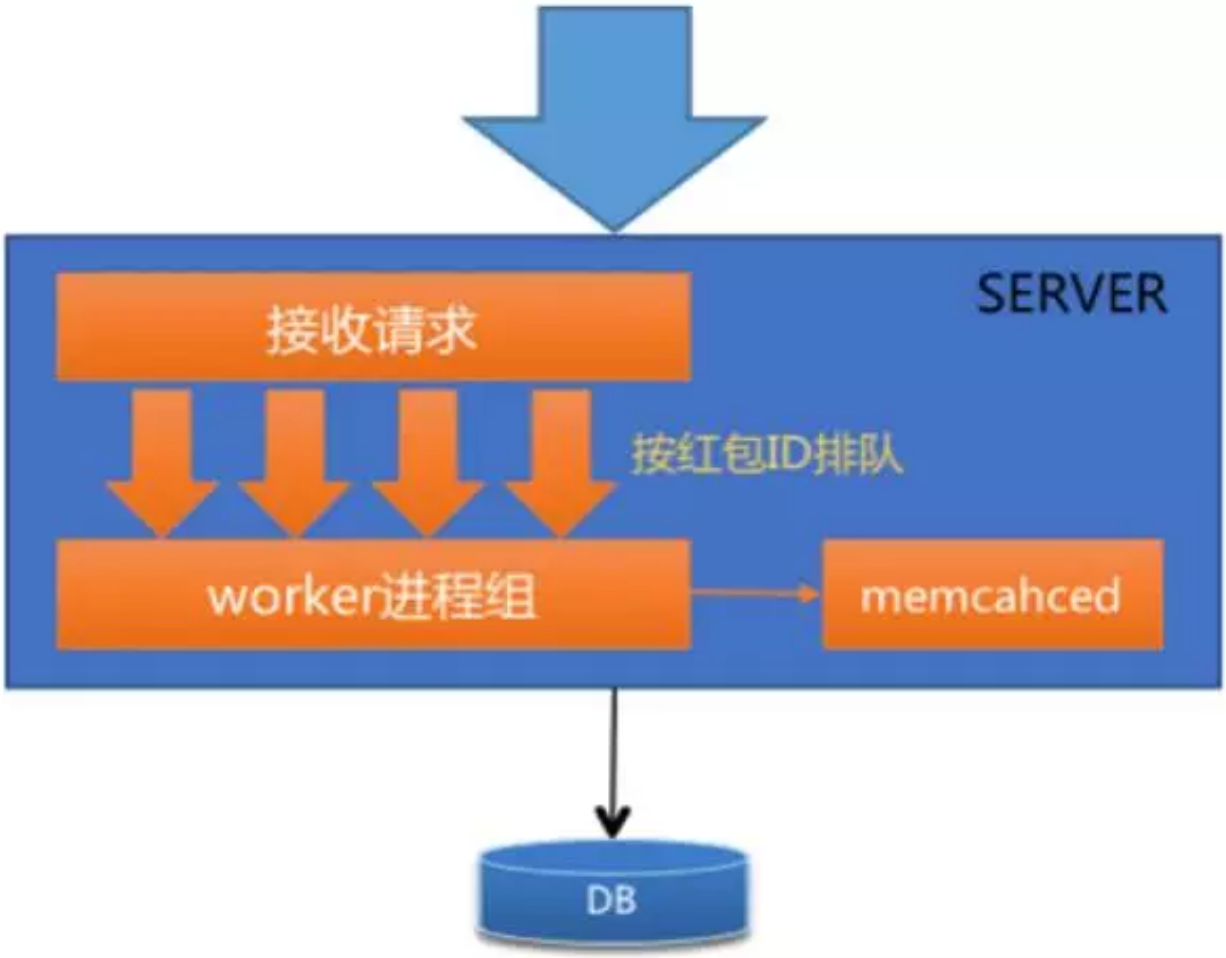
上面SET化方案已经介绍，同个红包ID的所有请求，按红包ID stick到同个SET中。不过在同个SET中，会存在多台Server服务器同时连接同一台DB（基于容灾、性能考虑，需要多台Server互备、均衡压力）。

为了使同一个红包ID的所有请求，stick到同一台Server服务器上，在SET化的设计之外，微信红包系统添加了一层基于红包ID hash值的分流，如下图所示。



其次，设计单机请求排队方案。

将stick到同一台Server上的所有请求在被接收进程接收后，按红包ID进行排队。然后串行地进入worker进程（执行业务逻辑）进行处理，从而达到排队的效果，如下图所示。



最后，增加memcached控制并发。

为了防止Server中的请求队列过载导致队列被降级，从而所有请求拥进DB，系统增加了与Server服务器同机部署的memcached，用于控制拆同一个红包的请求并发数。

具体来说，利用memcached的CAS原子累增操作，控制同时进入DB执行拆红包事务的请求数，超过预先设定数值则直接拒绝服务。用于DB负载升高时的降级体验。

通过以上三个措施，系统有效地控制了DB的“并发抢锁”情况。

3.双维度库表设计，保障系统性能稳定

红包系统的分库表规则，初期是根据红包ID的hash值分为多库多表。随着红包数据量逐渐增大，单表数据量也逐渐增加。而DB的性能与单表数据量有一定相关性。当单表数据量达到一定程度时，DB性能会有大幅度下降，影响系统性能稳定性。采用冷热分离，将历史冷数据与当前热数据分开存储，可以解决这个问题。

处理微信红包数据的冷热分离时，系统在以红包ID维度分库表的基础上，增加了以循环天分表的维度，形成了双维度分库表的特色。

具体来说，就是分库表规则像db_xx.t_y_dd设计，其中，xx/y是红包ID的hash值后三位，dd的取值范围在01~31，代表一个月天数最多31天。

通过这种双维度分库表方式，解决了DB单表数据量膨胀导致性能下降的问题，保障了系统性能的稳定性。同时，在热冷分离的问题上，又使得数据搬迁变得简单而优雅。

综上所述，微信红包系统在解决高并发问题上的设计，主要采用了SET化分治、请求排队、双维度分库表等方案，使得单组DB的并发性能提升了8倍左右，取得了很好的效果。

最后总结

微信红包系统是一个高并发的资金交易系统，最大的技术挑战是保障并发性能与资金安全。这种全新的技术挑战，传统的“秒杀”系统设计方案已不能完全解决。在分析了业界“秒杀”系统解决方案的基础上，微信红包采用了SET化、请求排队串行化、双维度分库表等设计，形成了独特的高并发、资金安全系统解决方案，并在平时节假日、2015和2016春节实践中充分证明了可行性，取得了显著的效果。在刚刚过去的2017鸡年除夕夜，微信红包收发峰值达到76万每秒，收发微信红包142亿个，微信红包系统的表现稳定，实现了除夕夜系统零故障。

作者介绍

方乐明，现任微信支付应用产品系统负责人，主要从事微信红包、微信转账、微信群收款等支付应用产品的系统设计、可用性提升、高性能解决方案设计等，曾负责2015、2016和2017年春节微信红包系统的性能优化与稳定性提升，取得良好的效果。

今日荐号

InfoQ推出的IT教育平台——斯达克学院（StuQ ）

为技术人提供系统实战课程

学习微服务，机器学习，iOS开发最潮流技术

回复“课程”获得热门课程介绍和优惠码

微信ID:stuq2015

今日荐文

点击下方图片即可阅读


前端之巅
ID:frontshow
紧跟前端发展
共享一线技术
关注前端之巅公众号
InfoQ