> 你知道的越多，你不知道的越多

> 点赞再看，养成习惯

> **GitHub**上已经开源 https://github.com/JavaFamily 有一线大厂面试点脑图和个人联系方式，欢迎Star和指教

# 前言

**Redis**在互联网技术存储方面使用如此广泛，几乎所有的后端技术面试官都要在**Redis**的使用和原理方面对小伙伴们进行360°的刁难。

作为一个在互联网公司面一次拿一次Offer的面霸，打败了无数竞争对手，每次都只能看到无数落寞的身影失望的离开，略感愧疚（**请允许我使用一下夸张的修辞手法**）。

于是在一个寂寞难耐的夜晚，我痛定思痛，决定开始写**《吊打面试官》**系列，希望能帮助各位读者以后面试势如破竹，对面试官进行360°的反击，吊打问你的面试官，让一同面试的同僚瞠目结舌，疯狂收割大厂Offer！

# 絮叨

之前写了很多**Redis**相关的知识点，我又大概回头看了下，除了比较底层的东西没写很深之外，我基本上的点都提到过了，我相信如果只是为了应付面试应该是够了的，但是如果你想把它们真正的吸收纳为己用，还是需要**大量的知识积累**，和**很多实际操作**的。

就我自己而言**Redis**在开发过程中实在用得太普遍了，热点数据的存储啊，整体性能的提升啊都会用到，但是就像我说的**技术就是一把双刃剑**，使用它们随之而来的问题也会很多的，我在老东家双十二就遇到**缓存雪崩**问题让整体服务宕机3分钟，相必大家都知道阿里今年的双十一数据了，那三分钟在这种时候到底值多少钱？真的不敢想象。

**Redis**的普遍我就拿掘金我自己的认知举例，不知道对不对，但是目测是对的。


**大家看到问题所在了么？**是的热门的赞的数据不是最新的，我盲猜一波上面的热门文章是缓存。失效时间应该是几十分钟的，为啥这么做呢？

热门文章是大家共同都会看到的，也就是**热点数据**，在那做缓存，他是不需要那么高的**实时性**的，那下面的文章列表是最新发布的文章，有高实时性的特点，大家访问多的放在缓存还可以给DB减少压力，我也不知道掘金是不是这么做的哈，反正道理是这么个道理了。

那什么场景是使用**Redis比较复杂的场景**，而且需要**大量中间件**和**业务逻辑**去配合的呢？

**秒杀**！是的就是今天的主题秒杀，我就用我自己的思路带大家一起看一下，设计一个秒杀从前到后，从内到外到底要技术人员做多少准备。

## 捞一下

上一期吊打系列我们提到了**Redis**相关的一些知识，还没看的小伙伴可以回顾一下，这**对于这期的阅读很有帮助**，涉及到主从同步、读写分离、持久化这样的知识点。

- **《吊打面试官》系列-Redis基础**
- **《吊打面试官》系列-缓存雪崩、击穿、穿透**
- **《吊打面试官》系列-Redis哨兵、持久化、主从、手撕LRU**
- **《吊打面试官》系列-Redis终章凛冬将至、FPX新王登基**
- **《吊打面试官》系列-Redis常见面试题（带答案）**

打好基础才可以写出更好的代码哟！不然就等着产品测试怼你吧。

# 正文

首先设计一个系统之前，我们需要先确认我们的业务场景是怎么样子的，我就**带着大家一起假设一个场景**好吧。

## 场景

我们现场要卖100件下面这个**婴儿纸尿裤**，然后我们根据以往这样秒杀活动的数据经验来看，目测来抢这100件纸尿裤的人足足有10万人。（南极人打钱！）


你一听，完了呀，这我们的服务器哪里顶得住啊！说真的直接打DB肯定挂。但是别急嘛，有**暖男**敖丙在，我们在开始之前应该先思考下**会出现哪些问题**？

问题

**高并发：**

是的**高并发**这个是我们想都不用想的一个点，一瞬间这么多人进来这不是高并发什么时候是呢？

是吧，秒杀的特点就是这样**时间极短、 瞬间用户量大。**

正常的店铺营销都是用极低的价格配合上短信、APP的精准推送，吸引特别多的用户来参与这场秒杀，**爽了商家苦了开发呀。**

秒杀大家都知道如果真的营销到位，价格诱人，几十万的流量我觉得完全不是问题，那单机的**Redis**我感觉3-4W的QPS还是能顶得住的，但是再高了就没办法了，那这个数据随便搞个热销商品的秒杀可能都不止了。

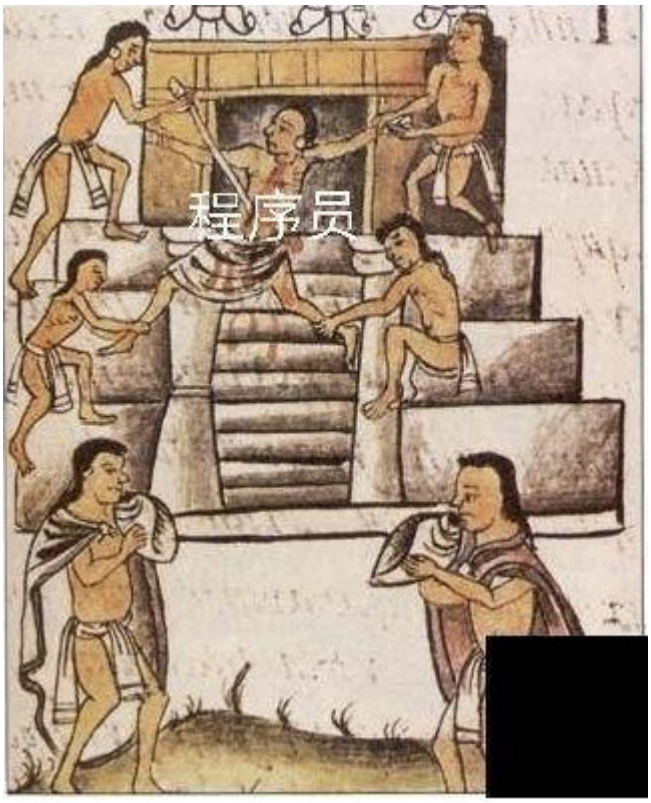
大量的请求进来，我们需要考虑的点就很多了，**缓存雪崩**，**缓存击穿**，**缓存穿透**这些我之前提到的点都是有可能发生的，出现问题打挂DB那就很难受了，活动失败用户体验差，活动人气没了，最后背锅的还是**开发。**


出来背锅了

**超卖：**

但凡是个秒杀，都怕**超卖**，我这里举例的只是尿不湿，要是换成100个华为MatePro30，商家的预算经费卖100个可以赚点还可以造势，结果你写错程序多卖出去200个，你不发货用户**投诉你**，平台**封你店**，你发货就**血亏**，你怎么办？ （没事看了敖丙的文章直接不怕）

那最后只能**杀个开发祭天**解气了，秒杀的价格本来就低了，基本上都是不怎么赚钱的，超卖了就恐怖了呀，所以超卖也是很关键的一个点。




**恶意请求：**

你这么低的价格，假如我抢到了，我转手卖掉我不是**血赚**？就算我不卖我也不亏啊，那用户知道，你知道，别的别有用心的人（黑客、黄牛...）肯定也知道的。

那简单啊，我知道你什么时候抢，我搞个几十台机器搞点脚本，我也模拟出来十几万个人左右的请求，那我是不是意味着我基本上有80%的成功率了。

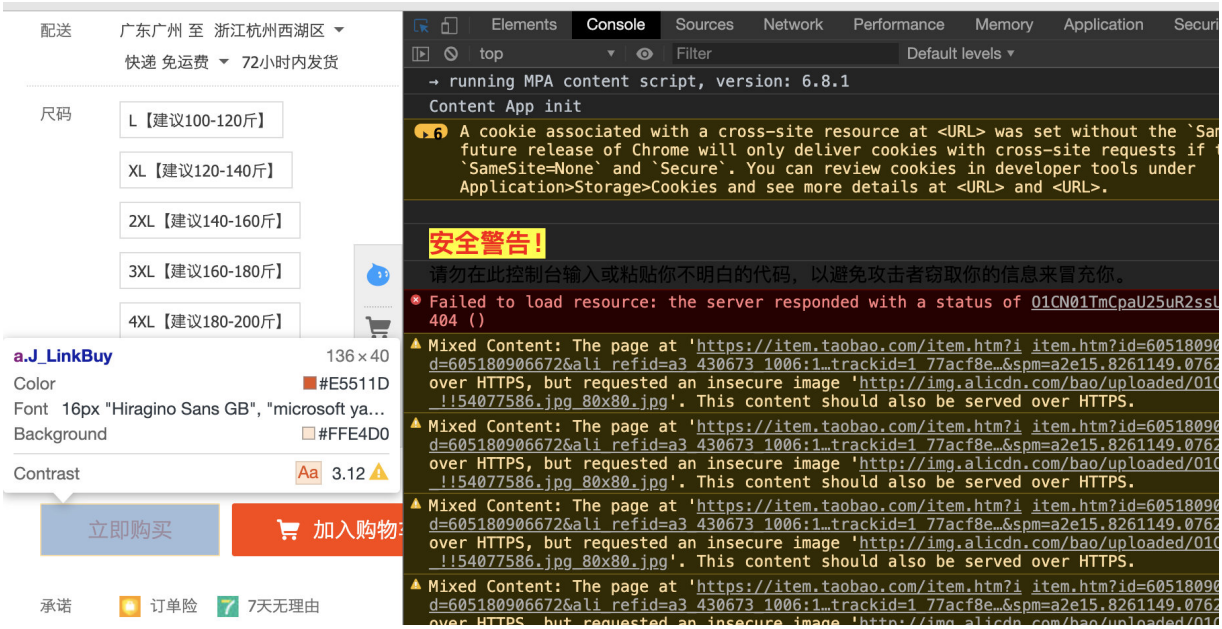
真实情况可能远远不止，因为机器请求的速度比人的手速往往快太多了，在贵州的敖丙我每年回家抢高铁票都是**秒光**的，我也不知道有没有黄牛的功劳，我要Diss你，黄牛。杰伦演唱会门票抢不到，我也Diss你。

Tip：科普下，小道消息了解到的，黄牛的抢票系统，比国内很多小公司的系统还吊很多，架构设计都是顶级的，我用**顶配的服务**加上**顶配的架构设计**，你还想看演唱会？还想回家？

不过不用黄牛我回家都难，我们云贵川跟我一样要回家过年的仔太多了555！

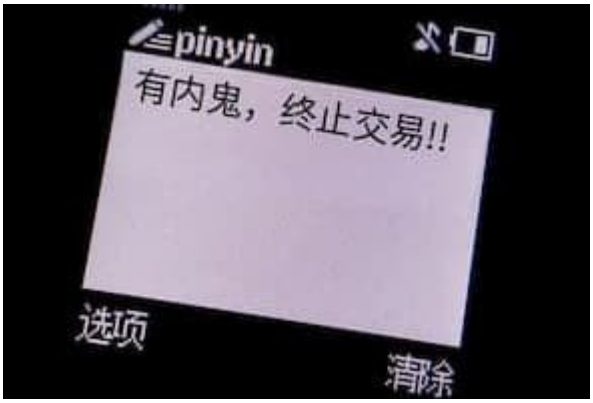
**链接暴露：**

前面几个问题大家可能都很好理解，一看到这个有的小伙伴可能会比较疑惑，啥是**链接暴露**呀？




相信是个开发同学都对这个画面一点都不陌生吧，懂点行的仔都可以打开谷歌的**开发者模式**，然后看看你的网页代码，有的就有URL，但是我写VUE的时候是事件触发然后去调用文件里面的接口看源码看不到，但是我可以点击一下**查看你的请求地址**啊，不过你好像可以对按钮在秒杀前置灰。

不管怎么样子都有危险，撇开外面的所有的东西你都挡住了，你卖这个东西实在便宜得过分，有诱惑力，你能保证**开发不动心**？开发知道地址，在秒杀的时候自己提前请求。。。（开发：怎么TM又是我）




**数据库：**

每秒上万甚至十几万的**QPS**（每秒请求数）直接打到**数据库**，基本上都要把库打挂掉，而且你服务不单单是做秒杀的还涉及其他的业务，你没做**降级、限流、熔断**啥的，别的一起挂，小公司的话可能**全站崩溃404。**

反正不管你秒杀怎么挂，你别把别的搞挂了对吧，搞挂了就不是杀一个程序员能搞定的。

程序员：我TM好难啊！

问题都列出来了，那怎么设计，怎么解决这些问题就是接下去要考虑的了，我们对症下药。

**服务单一职责：**

设计个能抗住高并发的系统，我觉得还是得**单一职责。**

什么意思呢，大家都知道现在设计都是**微服务的设计思想**，然后再用**分布式的部署方式**

也就是我们下单是有个订单服务，用户登录管理等有个用户服务等等，那为啥我们不给秒杀也开个服务，我们把秒杀的代码业务逻辑放一起。

单独给他建立一个数据库，现在的互联网架构部署都是**分库**的，一样的就是订单服务对应订单库，秒杀我们也给他建立自己的秒杀库。

至于表就看大家怎么设计了，该设置索引的地方还是要设置索引的，建完后记得用**explain**看看**SQL**的执行计划。（不了解的小伙伴也没事，MySQL章节我会说的）

单一职责的好处就是就算秒杀没抗住，秒杀库崩了，服务挂了，也不会影响到其他的服务。（强行高可用）

**秒杀链接加盐：**

我们上面说了链接要是提前暴露出去可能有人直接访问url就提前秒杀了，那又有小伙伴要说了我做个时间的校验就好了呀，那我告诉你，知道链接的地址比起页面人工点击的还是有**很大优势**。

我知道url了，那我通过程序不断获取最新的北京时间，可以达到**毫秒级别**的，我就在00毫秒的时候请求，我敢说绝对比你人工点的成功率大太多了，而且我可以一毫秒发送N次请求，搞不好你卖100个产品我全拿了。


**那这种情况怎么避免？**

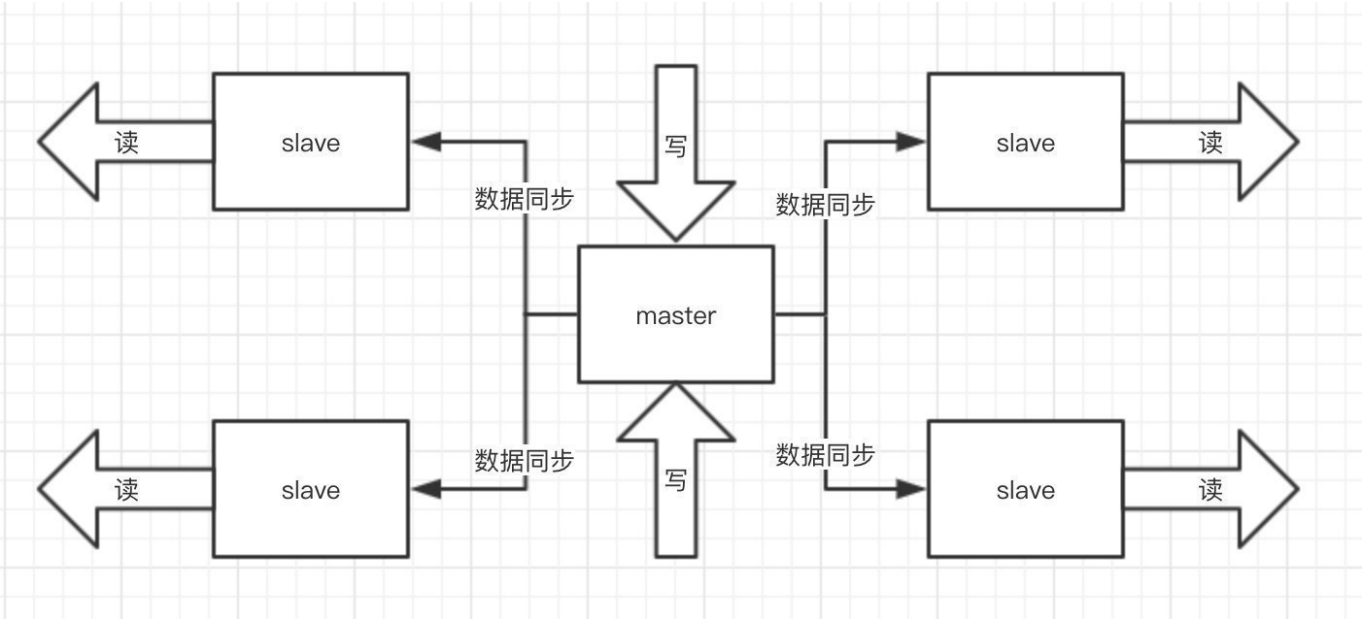
简单，把**URL动态化**，就连写代码的人都不知道，你就通过MD5之类的加密算法加密随机的字符串去做url，然后通过前端代码获取url后台校验才能通过。

暖男我呢，又准备了一个简单的url加密给大家尝尝鲜，还不**点个赞**？

```java
/**
 * Url md5加密
 * @param   url
 * @return  url
 * @author  敖丙
 * @公众号   JavaFamily
 */
public static String getKuaishouSign(String url) {
    try {
        MessageDigest md5 = MessageDigest.getInstance("MD5");
        md5.update((url).getBytes("UTF-8"));
        byte[] b = md5.digest();

        int i;
        StringBuffer buf = new StringBuffer();
        for (int offset = 0; offset < b.length; offset++) {
            i = b[offset];
            if (i < 0) {
                i += 256;
            }
            if (i < 16) {
                buf.append("0");
            }
            buf.append(Integer.toHexString(i));
        }

        url = buf.toString();
        System.out.println("result = " + url);

    } catch (Exception e) {
        log.error("error", e);
    }
    return url;
}
```

**Redis集群：**

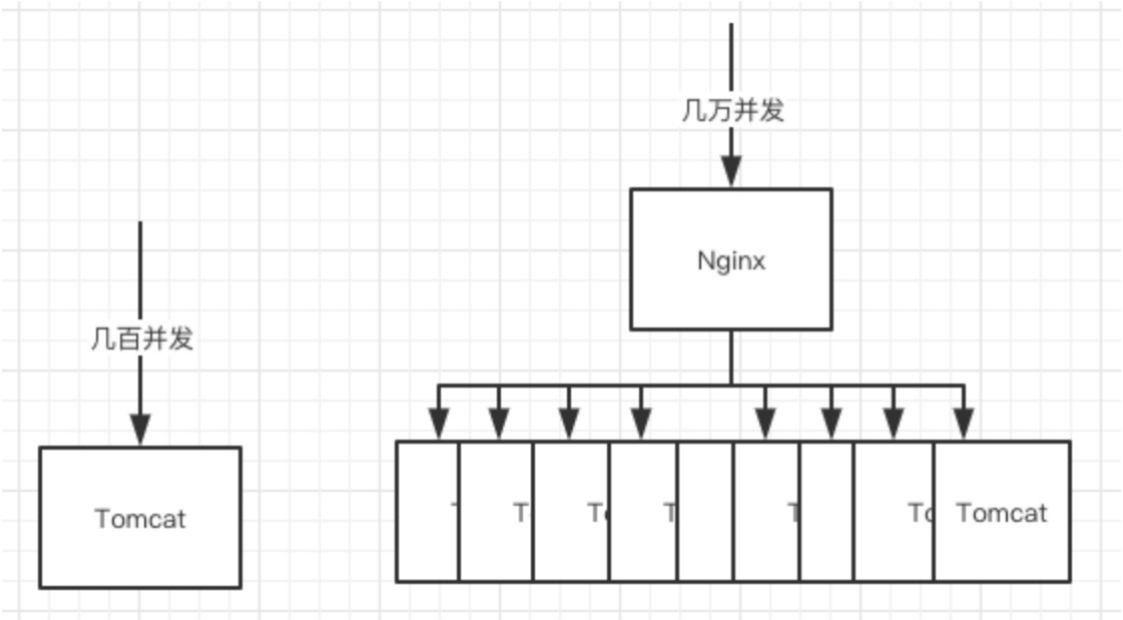
之前不是说单机的**Redis**顶不住嘛，那简单多找几个兄弟啊，秒杀本来就是读多写少，那你们是不是瞬间想起来我之前跟你们提到过的，**Redis集群**，**主从同步**、**读写分离**，我们还搞点**哨兵**，开启**持久化**直接无敌高可用！




**Nginx：**

**Nginx**大家想必都不陌生了吧，这玩意是**高性能的web服务器**，并发也随便顶几万不是梦，但是我们的**Tomcat**只能顶几百的并发呀，那简单呀**负载均衡**嘛，一台服务几百，那就多搞点，在秒杀的时候多租点**流量机。**

Tip：据我所知国内某大厂就是在去年春节活动期间租光了亚洲所有的服务器，小公司也很喜欢在双十一期间买流量机来顶住压力。




**这样一对比是不是觉得你的集群能顶很多了。**

**恶意请求拦截**也需要用到它，一般单个用户请求次数太夸张，不像人为的请求在网关那一层就得拦截掉了，不然请求多了他抢不抢得到是一回事，服务器压力上去了，可能占用网络带宽或者把**服务器打崩、缓存击穿**等等。

**资源静态化：**

秒杀一般都是特定的商品还有页面模板，现在一般都是前后端分离的，所以页面一般都是不会经过后端的，但是前端也要自己的服务器啊，那就把能提前放入**cdn服务器**的东西都放进去，反正把所有能提升效率的步骤都做一下，减少真正秒杀时候服务器的压力。

**按钮控制：**

大家有没有发现没到秒杀前，一般按钮都是**置灰**的，只有时间到了，才能点击。

这是因为怕大家在时间快到的最后几秒秒疯狂请求服务器，然后还没到秒杀的时候基本上服务器就挂了。

这个时候就需要前端的配合，定时去请求你的后端服务器，获取最新的北京时间，到时间点再给按钮可用状态。

按钮可以点击之后也得给他置灰几秒，不然他一样在开始之后一直点的。**你敢说你们秒杀的时候不是这样的？**

**对，你没有听错**

**限流：**

限流这里我觉得应该分为**前端限流**和**后端限流。**

**前端限流**：这个很简单，一般秒杀不会让你一直点的，一般都是点击一下或者两下然后几秒之后才可以继续点击，这也是保护服务器的一种手段。

**后端限流**：秒杀的时候肯定是涉及到后续的订单生成和支付等操作，但是都只是成功的幸运儿才会走到那一步，那一旦100个产品卖光了，return了一个false，前端直接秒杀结束，然后你后端也关闭后续无效请求的介入了。

Tip：真正的限流还会有限流组件的加入例如：阿里的Sentinel、Hystrix等。我这里就不展开了，就说一下物理的限流。

**库存预热：**

**秒杀的本质，就是对库存的抢夺**，每个秒杀的用户来你都去数据库查询库存校验库存，然后扣减库存，撇开性能因数，你不觉得这样好繁琐，对业务开发人员都不友好，而且数据库顶不住啊。

**开发：你tm总算为我着想一次了。**


**那怎么办?**

我们都知道数据库顶不住但是他的兄弟非关系型的数据库**Redis**能顶啊!

那不简单了，我们要开始秒杀前你通过定时任务或者运维同学**提前把商品的库存加载到Redis中**去，让整个流程都在Redis里面去做，然后等秒杀介绍了，再异步的去修改库存就好了。

但是用了Redis就有一个问题了，我们上面说了我们采用**主从**，就是我们会去读取库存然后再判断然后有库存才去减库存，正常情况没问题，但是高并发的情况问题就很大了。

这里我就不画图了，我本来想画图的，想了半天我觉得语言可能更好表达一点。

**多品几遍！！！**就比如现在库存只剩下1个了，我们高并发嘛，4个服务器一起查询了发现都是还有1个，那大家都觉得是自己抢到了，就都去扣库存，那结果就变成了-3，是的只有一个是真的抢到了，别的都是超卖的。咋办?

Lua:

之前的文章就简单的提到了他，我今天就多一定点篇幅说一下吧。

> **Lua** 脚本功能是 Reids在 2.6 版本的最大亮点， 通过内嵌对 Lua 环境的支持， Redis 解决了长久以来不能高效地处理 **CAS** （check-and-set）命令的缺点， 并且可以通过组合使用多个命令， 轻松实现以前很难实现或者不能高效实现的模式。

**Lua脚本是类似Redis事务，有一定的原子性，不会被其他命令插队，可以完成一些Redis事务性的操作。**这点是关键。

知道原理了，我们就写一个脚本把判断库存扣减库存的操作都写在一个脚本丢给Redis去做，那到0了后面的都Return False了是吧，一个失败了你修改一个开关，直接挡住所有的请求，然后再做后面的事情嘛。

**限流&降级&熔断&隔离:**

这个为啥要做呢，不怕一万就怕万一，万一你真的顶不住了，**限流**，顶不住就挡一部分出去但是不能说不行，**降级**，降级了还是被打挂了，**熔断**，至少不要影响别的系统，**隔离**，你本身就独立的，但是你会调用其他的系统嘛，你快不行了你别拖累兄弟们啊。

**削峰填谷:**

一说到这个名词，很多小伙伴就知道了，对的**MQ**，你买东西少了你直接100个请求改库我觉得没问题，但是万一秒杀一万个，10万个呢？服务器挂了，**程序员又要背锅的。**

Tip：**可能小伙伴说我们业务达不到这个量级，没必要。但是我想说我们写代码，就不应该写出有逻辑漏洞的代码，至少以后公司体量上去了，别人一看居然不用改代码，一看代码作者是敖丙？有点东西！**

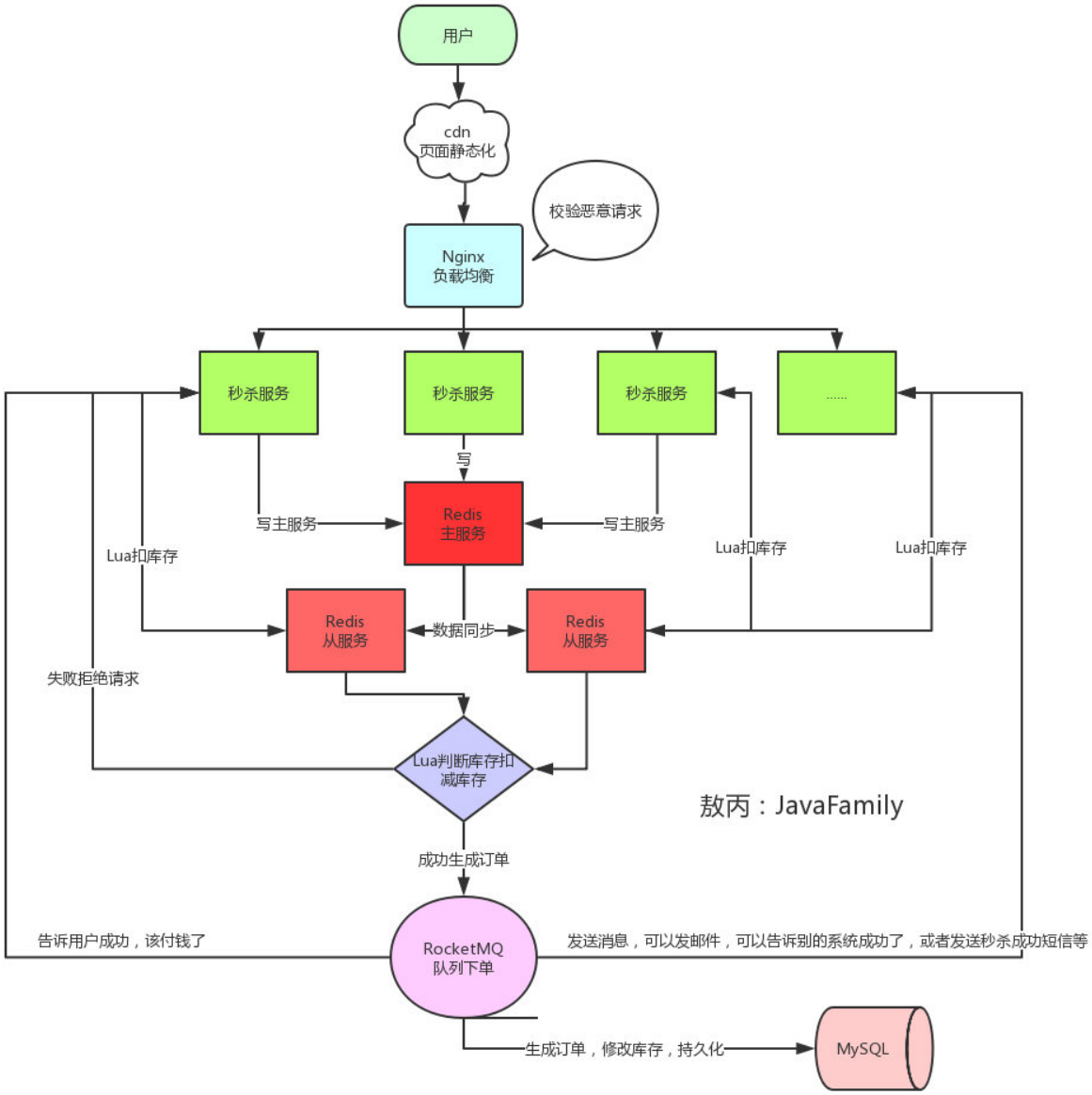
你可以把它放消息队列，然后一点点消费去改库存就好了嘛，不过单个商品其实一次修改就够了，我这里说的是**某个点多个商品**一起秒杀的场景，像极了双十一零点。

## 总结

到这里我想我已经基本上把该考虑的点还有对应的解决方案也都说了一下，不知道还有没有没考虑到的，但是就算没考虑到我想我这个设计，应该也能撑住一个完整的秒杀流程。

（有大佬的话给敖丙点多的思路，去GitHub https://github.com/JavaFamily 上给我提，也有我的联系）

最后我就画个完整的流程图给大家收个尾吧！




Tip：这个链路还是比较简单的，很多细节的点全部画出来就太复杂了，我上面已经提到了所有的注意点了，大家都看看，真正的秒杀有比我这个简单的，也有比我这个复杂N倍的，之前的电商老东家就做的很高级，有机会也可以跟你们探讨，不过是面试嘛，我就给思路，让你理解比较关键的点。

秒杀这章我脑细胞死了很多，考虑了很多个点，最后还是出来了，忍不住给自己**点赞**！

**（这章是真的不要白嫖，每次都看了不点赞，你们想白嫖我么？你们好坏喲，不过我好喜欢）**

# 总结

我们玩归玩，闹归闹，别拿面试开玩笑。

秒杀不一定是每个同学都会问到的，至少肯定没**Redis基础**那样常问，但是一旦问到，大家一定要回答到点上。

至少你得说出**可能出现的情况**，**需要注意的情况**，以及对于的**解决思路和方案。**

最后就是需要对整个链路比较熟悉，注意是一个完整的链路，前端怎么设计的呀，网关的作用呀，怎么**解决Redis的并发竞争**啊，**数据的同步方式**呀，**MQ的作用**啊。

（提到MQ又是一整条的知识链路，什么异步、削峰、解耦等等，所以面试，我们还是不打没有把握的胜仗）

### 流着泪说再见

**Redis系列**到此是真的要跟大家说再见了，写了7篇文章，其实很多大佬的思路和片段真心赞，其实大家看出来了我的文章个人风格色彩特别浓厚，我个人在生活中就是这么说话的，也希望用这种风格把原本枯燥乏味的知识点让大家都像看小说一样津津有味的看下去，不知道大家什么感受，好的不好的都请给我留言。

我这个系列的我会写到我**GitHub** https://github.com/JavaFamily 图中所有的知识点，以后就麻烦大家多多关照了，我写作的时间都是业余时间，基本上周末和晚上的时间都贡献出来了，我也是个新人很多点也没接触到，也要看书看资料才能写出来，所以有时候还是希望大家多多包涵。

那我们下期见！

下期写_____?

不告诉你，哈哈！

## 日常求赞

好了各位，以上就是这篇文章的全部内容了，能看到这里的人呀，都是**人才**。

我后面会每周都更新几篇《吊打面试官》系列和互联网常用技术栈相关的文章，非常感谢**人才**们能看到这里，如果这个文章写得还不错，觉得「敖丙」我**有点东西**的话 **求点赞**👍 **求关注**♡ **求分享**👥 对暖男我来说真的 **非常有用**！！！

创作不易，各位的支持和认可，就是我创作的最大动力，我们下篇文章见！

敖丙 | 文 【原创】【转载请联系本人】 如果本篇博客有任何错误，请批评指教，不胜感激！

---

> 《吊打面试官》系列每周持续更新，可以关注我的公众号「**JavaFamily**」第一时间阅读和催更（公众号比博客早一到两篇哟），本文**GitHub**上已经收录https://github.com/JavaFamily，有一线大厂面试点思维导图，欢迎Star和完善，里面也有我个人联系方式有什么问题也可以直接找我，也有人才交流群，我们一起有点东西。


有点
东西
《吊打面试官》
每周两更