> 你知道的越多，你不知道的越多

> 点赞再看，养成习惯

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（**请允许我使用一下夸张的修辞手法**），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，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！

一点感慨

本来都把稿子放到公众号保存了，洗澡的时候想了一下晚上的比赛，觉得还是打开电脑写点东西，跟文章内容没关系，只是一点个人的感慨，不知道多少小伙伴看了昨天**SKT VS G2**的比赛，又不知道多少小伙伴还记得**Faker**手抖的那一幕。

不知道你们看了是什么感受，我看到他手抖的时候我内心也抖了，世界赛我支持的都是**LPL**的队伍，但是我喜欢李哥这个人，那种对胜利的执著，这么多年了那种坚持自己的坚持，这么多利益诱惑在面前却只想要胜利，这样的人我好喜欢啊，我想很多人也喜欢。

可能就像很多网友说的那样，英雄迟暮，但是我觉得他还是有点东西，就像很多人说我们程序员只能吃年轻饭一样，但是如果你坚持自己的坚持，做个腹有诗书气自华的仔，我想最后肯定会得到自己的得到。

好了我也不煽情了，我们开始讲技术吧。

正文

上一期吊打系列我们提到了Redis的基础知识，还没看的小伙伴可以回顾一下

《吊打面试官》系列-Redis基础

那提到**Redis**我相信各位在面试，或者实际开发过程中对缓存**雪崩**，**穿透**，**击穿**也不陌生吧，就算没遇到过但是你肯定听过，那三者到底有什么区别，我们又应该怎么去防止这样的情况发生呢，我们有请下一位受害者。

面试开始

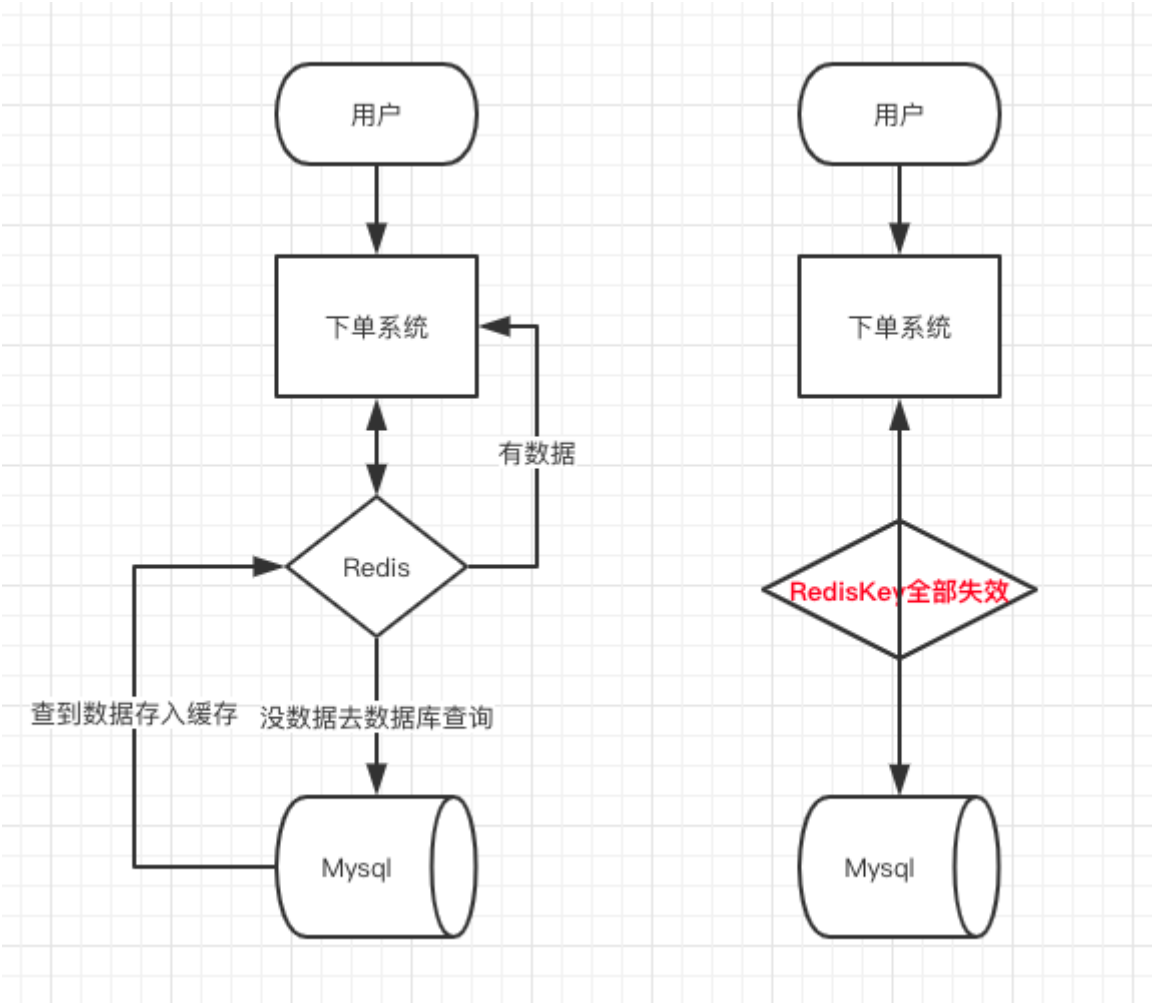
> 一个大腹便便，穿着格子衬衣的中年男子，拿着一个满是划痕的mac向你走来，看着快秃顶的头发，心想着肯定是尼玛顶级架构师吧！但是我们腹有诗书气自华，虚都不虚。

小伙子我看你的简历上写到了Redis，那么我们直接开门见山，直接怼常见的几个大问题，Redis雪崩了解么？

帅气迷人的面试官您好，我了解的，目前电商首页以及热点数据都会去做缓存 ，一般缓存都是定时任务去刷新，或者是查不到之后去更新的，定时任务刷新就有一个问题。

举个简单的例子：如果所有首页的Key失效时间都是12小时，中午12点刷新的，我零点有个秒杀活动大量用户涌入，假设当时每秒 6000 个请求，本来缓存在可以扛住每秒 5000 个请求，但是缓存当时所有的Key都失效了。此时 1 秒 6000 个请求全部落数据库，数据库必然扛不住，它会报一下警，真实情况可能DBA都没反应过来就直接挂了。此时，如果没用什么特别的方案来处理这个故障，DBA 很着急，重启数据库，但是数据库立马又被新的流量给打死了。这就是我理解的缓存雪崩。

我刻意看了下我做过的项目感觉再吊的都不允许这么大的QPS直接打DB去，不过没慢SQL加上分库，大表分表可能还还算能顶，但是跟用了Redis的差距还是很大



同一时间大面积失效，那一瞬间Redis跟没有一样，那这个数量级别的请求直接打到数据库几乎是灾难性的，你想想如果打挂的是一个用户服务的库，那其他依赖他的库所有的接口几乎都会报错，如果没做熔断等策略基本上就是瞬间挂一片的节奏，你怎么重启用户都会把你打挂，等你能重启的时候，用户早就睡觉去了，并且对你的产品失去了信心，什么垃圾产品。

面试官摸了摸自己的头发，嗯还不错，那这种情况咋整？你都是怎么去应对的？

处理缓存雪崩简单，在批量往**Redis**存数据的时候，把每个Key的失效时间都加个随机值就好了，这样可以保证数据不会在同一时间大面积失效，我相信，Redis这点流量还是顶得住的。

```
setRedis (Key, value, time + Math.random() * 10000) ;
```

如果**Redis**是集群部署，将热点数据均匀分布在不同的**Redis**库中也能避免全部失效的问题，不过本渣我在生产环境中操作集群的时候，单个服务都是对应的单个**Redis**分片，是为了方便数据的管理，但是也同样有了可能会失效这样的弊端，失效时间随机是个好策略。

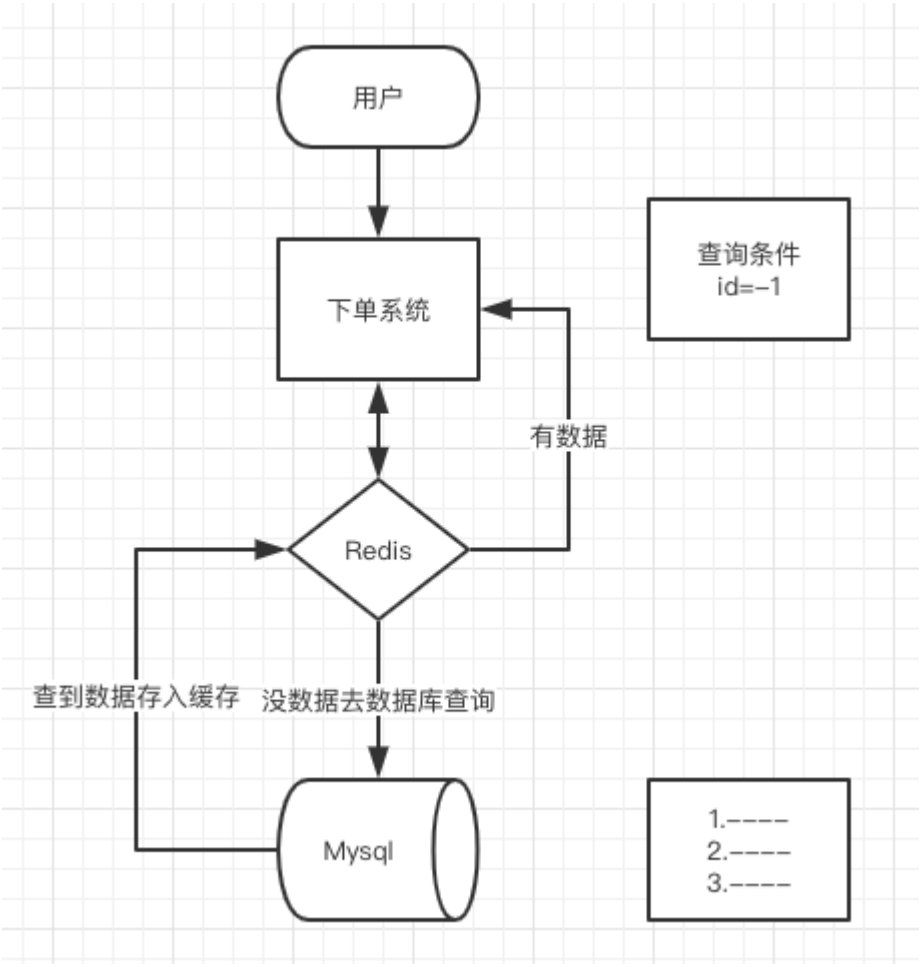
或者设置热点数据永远不过期，有更新操作就更新缓存就好了（比如运维更新了首页商品，那你刷下缓存就完事了，不要设置过期时间），电商首页的数据也可以用这个操作，保险。

那你了解缓存穿透和击穿么，可以说说他们跟雪崩的区别么？

嗯，了解，我先说一下缓存穿透吧，缓存穿透是指缓存和数据库中都没有的数据，而用户不断发起请求，我们数据库的 id 都是1开始自增上去的，如发起为id值为 -1 的数据或 id 为特别大不存在的数据。这时的用户很可

能是攻击者，攻击会导致数据库压力过大，严重会击垮数据库。

小点的单机系统，基本上用postman就能搞死，比如我自己买的阿里云服务



像这种你如果不对参数做校验，数据库id都是大于0的，我一直用小于0的参数去请求你，每次都能绕开Redis直接打到数据库，数据库也查不到，每次都这样，并发高点就容易崩掉了。

至于**缓存击穿**嘛，这个跟**缓存雪崩**有点像，但是又有一点不一样，缓存雪崩是因为大面积的缓存失效，打崩了DB，而缓存击穿不同的是**缓存击穿**是指一个Key非常热点，在不停的扛着大并发，大并发集中对这一个点进行访问，当这个Key在失效的瞬间，持续的大并发就穿破缓存，直接请求数据库，就像在一个完好无损的桶上凿开了一个洞。

面试官露出欣慰的眼光，那他们分别怎么解决

缓存穿透我会在接口层增加校验，比如用户鉴权校验，参数做校验，不合法的参数直接代码Return，比如：id做基础校验，id <=0的直接拦截等。

这里我想提的一点就是，我们在开发程序的时候都要有一颗“不信任”的心，就是不要相信任何调用方，比如你提供了API接口出去，你有这几个参数，那我觉得作为被调用方，任何可能的参数情况都应该被考虑到，做校验，因为你不相信调用你的人，你不知道他会传什么参数给你。

举个简单的例子，你这个接口是分页查询的，但是你没对分页参数的大小做限制，调用的人万一一口气查Integer.MAX_VALUE 一次请求就要你几秒，多几个并发你不就挂了么？是公司同事调用还好大不了发现了改

掉，但是如果是黑客或者竞争对手呢？在你双十一当天就调你这个接口会发生什么，就不用我说了吧。这是之前的Leader跟我说的，我觉得大家也都应该了解下。

从缓存取不到的数据，在数据库中也没有取到，这时也可以将对应Key的Value对写为null、位置错误、稍后重试这样的值具体取啥问产品，或者看具体的场景，缓存有效时间可以设置短点，如30秒（设置太长会导致正常情况也没法使用）。

这样可以防止攻击用户反复用同一个id暴力攻击，但是我们要知道正常用户是不会在单秒内发起这么多次请求的，那网关层**Nginx**本渣我也记得有配置项，可以让运维大大对单个IP每秒访问次数超出阈值的IP都拉黑。

那你还有别的办法么？

还有我记得**Redis**还有一个高级用法**布隆过滤器（Bloom Filter）**这个也能很好的防止缓存穿透的发生，他的原理也很简单就是利用高效的数据结构和算法快速判断出你这个Key是否在数据库中存在，不存在你return就好了，存在你就去查了DB刷新KV再return。

那又有小伙伴说了如果黑客有很多个IP同时发起攻击呢？这点我一直也不是很想得通，但是一般级别的黑客没这么多肉鸡，再者正常级别的**Redis**集群都能抗住这种级别的访问的，小公司我想他们不会感兴趣的。把系统的高可用做好了，集群还是很能顶的。

缓存击穿的话，设置热点数据永远不过期。或者加上互斥锁就能搞定了

作为暖男，代码我肯定帮你们准备好了

```java
/**
 * 获取数据
 * @param Key                          查询参数
 * @return data                        数据
 * @throws InterruptedException        异常
 * @author 敖丙
 */
public static String getData(String Key) throws InterruptedException {
    // 从redis查询数据
    String result = getDataByKV(Key);
    // 参数校验
    if (StringUtils.isBlank(result)) {
        // 获取锁
        if (reenLock.tryLock()) {
            // 去数据库查询
            result = getDataByDB(Key);
            // 校验
            if (StringUtils.isNotBlank(result)) {
                // 搞进缓存
                setDataToKV(Key, result);
            }
            // !!!释放锁 正常会在finally里面释放
            reenLock.unLock();
        } else {
            // 睡一会再拿
            Thread.sleep(100L);
            result = getData(Key);
        }
    }
    return result;
}
// 这里面的锁都是单机玩玩，分布式锁还是得靠lua脚本这样的
```

##面试结束

嗯嗯还不错，三个点都回答得很好，今天也不早了，面试就先到这里，明天你再过来二面我继续问一下你关于Redis集群高可用，主从同步，哨兵等知识点的问题。

晕居然还有下一轮面试！（强行下一期的伏笔哈哈）但是为了offer还是得舔，嗯嗯，好的帅气面试官。

能回答得这么全面这么细节还是忍不住点赞

（**暗示点赞，每次都看了不点赞，你们想白嫖我么？你们好坏喲，不过我喜欢**）

总结

> 我们玩归玩，闹归闹，别拿面试开玩笑。
>
> 本文简单的介绍了，**Redis**的**雪崩**，**击穿**，**穿透**，三者其实都差不多，但是又有一些区别，在面试中其实这是问到缓存必问的，大家不要把三者搞混了，因为缓存雪崩、穿透和击穿，是缓存最大的问题，要么不出现，一旦出现就是致命性的问题，所以面试官一定会问你。
>
> 大家一定要理解是**怎么发生的**，以及是怎么去**避免**的，发生之后又怎么去**抢救**，你可以不是知道很深入，但是你不能一点都不去想，面试有时候不一定是对知识面的拷问，或许是对你的态度的拷问，如果你思路清晰，然后**知其然还知其所以然**那就很赞，还知道怎么预防那来上班吧。
>
> ### 最后暖男我继续给你们做个小的技术总结：
>
> 一般避免以上情况发生我们从三个时间段去分析下：
>
> - 事前：**Redis** 高可用，主从+哨兵，**Redis cluster**，避免全盘崩溃。
> - 事中：本地 **ehcache** 缓存 + **Hystrix** 限流+降级，避免** MySQL** 被打死。

- 事后：**Redis** 持久化 **RDB**+**AOF**，一旦重启，自动从磁盘上加载数据，快速恢复缓存数据。

上面的几点我会在吊打系列Redis篇全部讲一下这个月应该可以吧Redis更完，限流组件，可以设置每秒的请求，有多少能通过组件，剩余的未通过的请求，怎么办？**走降级**！可以返回一些默认的值，或者友情提示，或者空白的值。

好处：

数据库绝对不会死，限流组件确保了每秒只有多少个请求能通过。 只要数据库不死，就是说，对用户来说，3/5 的请求都是可以被处理的。 只要有 3/5 的请求可以被处理，就意味着你的系统没死，对用户来说，可能就是点击几次刷不出来页面，但是多点几次，就可以刷出来一次。

这个在目前主流的互联网大厂里面是最常见的，你是不是好奇，某明星爆出什么事情，你发现你去微博怎么刷都空白界面，但是有的人又直接进了，你多刷几次也出来了，现在知道了吧，那是做了降级，牺牲部分用户的体验换来服务器的安全，可还行？

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Java技术栈相关的文章。如果你有什么想知道的，也可以留言给我，我一有时间就会写出来，我们共同进步。

非常感谢**靓仔/靓女**您能看到这里，如果这个文章写得还不错的话 **求点赞 求关注 求分享 求留言** **（对我非常有用）**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，拜了个拜！

敖丙 | 文 【原创】

> 每周都会持续更新《吊打面试官》系列可以关注我的公众号第一时间阅读和催更，也可以在公众号回复【人才】加入人才交流群一起讨论面试题，就业和工作上有什么问题也可以直接滴滴我，我也是个新人，不过不影响我们一起进步，作为渣男，我给不了你工作，还给不了你温暖嘛？


有点
东西
《吊打面试官》
每周一更