链滴

缓存穿透、缓存击穿和缓存雪崩

作者：jianzh5

原文链接：https://ld246.com/article/1604364141310

来源网站：链滴

在Redis缓存中有三个必须要知道概念：**缓存穿透、缓存击穿和缓存雪崩。**

缓存穿透

什么是缓存穿透呢？它是指当用户在查询一条数据的时候，而此时数据库和缓存却没有关于这条数据任何记录，而这条数据在缓存中没找到就会向数据库请求获取数据。它拿不到数据时，是会一直查询据库，这样会对数据库的访问造成很大的压力。

如：用户查询一个 id = -1 的商品信息，一般数据库 id 值都是从 1 开始自增，很明显这条信息是不数据库中，当没有信息返回时，会一直向数据库查询，给当前数据库的造成很大的访问压力。

这时候我们要想一想，该如何解决这个问题呢？o(╥﹏╥)o

一般我们可以想到从缓存开始出发，想如果我们给缓存设置一个如果当前数据库不存在的信息，把它存成一个空对象，返回给用户。

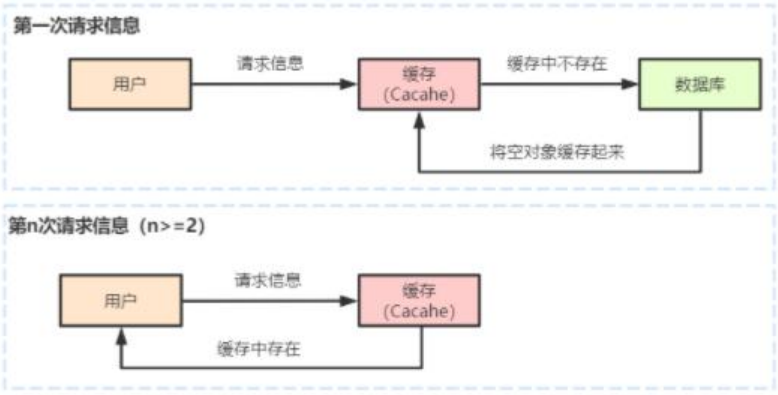
^_^没错，这是一个解决方案，也就是我们常说的缓存空对象（代码维护简单，但是效果不是很好）。

Redis 也为我们提供了一种解决方案，那就是布隆过滤器（代码维护比较复杂，效果挺好的）。

那接下来，二哈先解释下这两种方案：

缓存空对象

缓存空对象它就是指一个请求发送过来，如果此时缓存中和数据库都不存在这个请求所要查询的相关息，那么数据库就会返回一个空对象，并将这个空对象和请求关联起来存到缓存中，当下次还是这个求过来的时候，这时缓存就会命中，就直接从缓存中返回这个空对象，这样可以减少访问数据库的压，提高当前数据库的访问性能。我们接下来可以看下面这个流程



这时候，我们就会问了呀 ~，如果大量不存在的请求过来，那么这时候缓存岂不是会缓存许多空对象吗~~~

没错哦！这也是使用缓存空对象会导致的一个问题：如果时间一长这样会导致缓存中存在大量空对象这样不仅会占用许多的内存空间，还会浪费许多资源呀！。那这有没有什么可以解决的方法呢？我们想一想：我们可以将这些对象在一段时间之后清理下不久可以了吗 ~

嗯嗯，没错！在想想 Redis 里是不是给我们提供了有关过期时间的命令呀(^▽^)，这样我们可以在设空对象的时间，顺便设置一个过期时间，就可以解决个问题了呀！

setex key seconds valule:设置键值对的同时指定过期时间(s)

在Java 中直接调用 API 操作即可：

```
redisCache.put(Integer.toString(id), null, 60) //过期时间为 60s
```

布隆过滤器

布隆过滤器是一种基于概率的数据结构，主要使用来判断当前某个元素是否在该集合中，运行速度快我们也可以简单理解为是一个不怎么精确的 set 结构（set 具有去重的效果）。但是有个小问题是：你使用它的 contains 方法去判断某个对象是否存在时，它可能会误判。也就是说布隆过滤器不是特不精确，但是只要参数设置的合理，它的精确度可以控制的相对足够精确，只会有小小的误判概率（是可以接受的呀 ~）。当布隆过滤器说某个值存在时，这个值可能不存在；当它说不存在时，那就肯不存在。

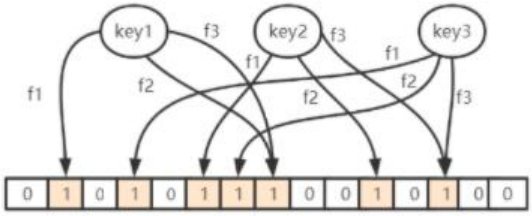
这里有个典型的例子呀，来自钱大：

打个比方，当它说不认识你时，肯定就不认识；当它说见过你时，可能根本就没见过面，不过因为你脸跟它认识的人中某脸比较相似 (某些熟脸的系数组合)，所以误判以前见过你。在上面的使用场景中布隆过滤器能准确过滤掉那些已经看过的内容，那些没有看过的新内容，它也会过滤掉极小一部分（判)，但是绝大多数新内容它都能准确识别。这样就可以完全保证推荐给用户的内容都是无重复的。

说了这么久，那布隆过滤器到底有什么特点呢：

1. 一个非常大的二进制位数组（数组中只存在 0 和 1）
2. 拥有若干个哈希函数（Hash Function）
3. 在空间效率和查询效率都非常高
4. 布隆过滤器不会提供删除方法，在代码维护上比较困难。

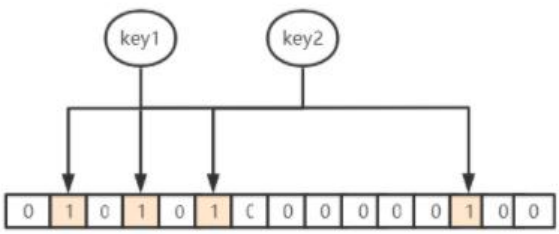
每个布隆过滤器对应到 Redis 的数据结构里面就是一个大型的位数组和几个不一样的无偏 hash 函数所谓无偏就是能够把元素的 hash 值算得比较均匀。



向布隆过滤器中添加 key 时，会使用多个 hash 函数对 key 进行 hash 算得一个整数索引值然后对位组长度进行取模运算得到一个位置，每个 hash 函数都会算得一个不同的位置。再把位数组的这几个置都置为 1 就完成了 add 操作。（每一个 key 都通过若干的hash函数映射到一个巨大位数组上，射成功后，会在把位数组上对应的位置改为1。）

那为什么布隆过滤器会存在误判率呢?

如下图所示：



当 key1 和 key2 映射到位数组上的位置为 1 时，假设这时候来了个 key3，要查询是不是在里面，恰 key3 对应位置也映射到了这之间，那么布隆过滤器会认为它是存在的，这时候就会产生误判（因为明 key3 是不在的）。

O(∩_∩)O哈哈~，这时候你会问了：如何提高布隆过滤器的准确率呢?

要提高布隆过滤器的准确率，就要说到影响它的三个重要因素：

1. 哈希函数的好坏
2. 存储空间大小
3. 哈希函数个数

hash函数的设计也是一个十分重要的问题，对于好的hash函数能大大降低布隆过滤器的误判率。

（这就好比优秀的配件之所以能够运行这么顺畅就在于其内部设计的得当。）

同时，对于一个布隆过滤器来说，如果其位数组越大的话，那么每个key通过hash函数映射的位置会得稀疏许多，不会那么紧凑，有利于提高布隆过滤器的准确率。同时，对于一个布隆过滤器来说，如key通过许多hash函数映射，那么在位数组上就会有许多位置有标志，这样当用户查询的时候，在通布隆过滤器来找的时候，误判率也会相应降低。

对于其内部原理，有兴趣的同学可以看看关于布隆过滤的数学知识，里面有关于它的设计算法和数学识。（其实也挺简单~）

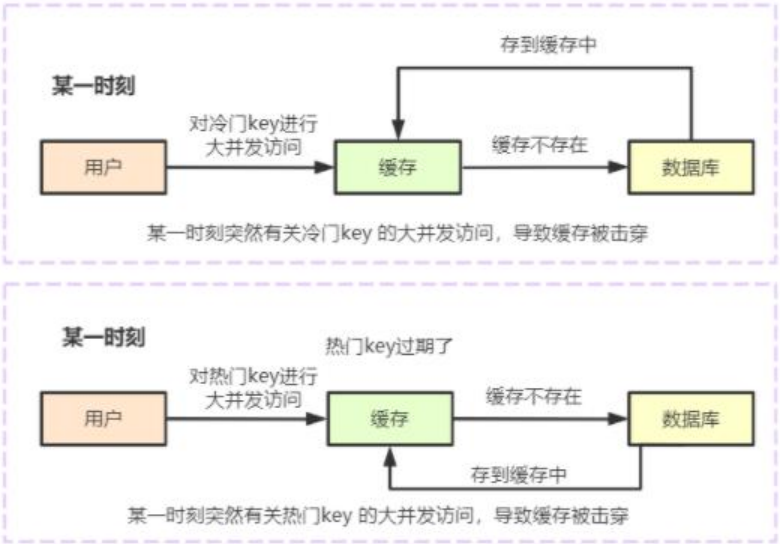
缓存击穿

缓存击穿是指有某个key经常被查询，经常被用户特殊关怀，用户非常 love 它 (^▽^)，也就类比“熟” 或者 一个key经常不被访问。但是这时候，如果这个key在缓存的过期时间失效的时候或者这是个门key时，这时候突然有大量有关这个key的访问请求，这样会导致大并发请求直接穿透缓存，请求数库，瞬间对数据库的访问压力增大。

归纳起来：造成缓存击穿的原因有两个。

（1）一个“冷门”key，突然被大量用户请求访问。

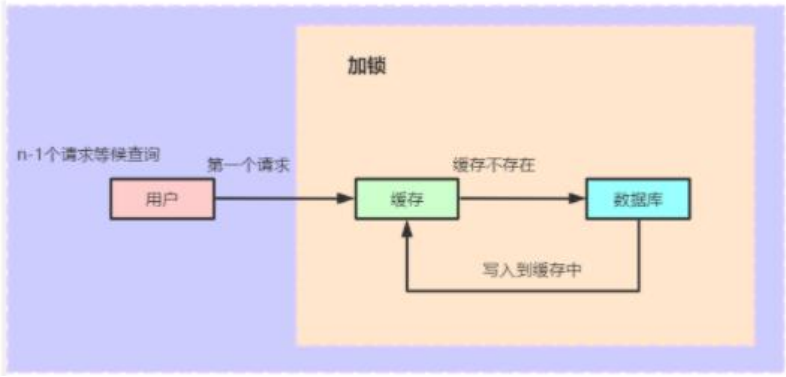
（2）一个“热门”key，在缓存中时间恰好过期，这时有大量用户来进行访问。



原文链接：缓存穿透、缓存击穿和缓存雪崩

对于缓存击穿的问题：我们常用的解决方案是加锁。对于key过期的时候，当key要查询数据库的时候上一把锁，这时只能让第一个请求进行查询数据库，然后把从数据库中查询到的值存储到缓存中，对剩下的相同的key，可以直接从缓存中获取即可。

如果我们是在单机环境下：直接使用常用的锁即可（如： Lock、Synchronized等），在分布式环境我们可以使用分布式锁，如：基于数据库、基于Redis或者zookeeper 的分布式锁。



缓存雪崩

缓存雪崩是指在某一个时间段内，缓存集中过期失效，如果这个时间段内有大量请求，而查询数据量大，所有的请求都会达到存储层，存储层的调用量会暴增，引起数据库压力过大甚至宕机。

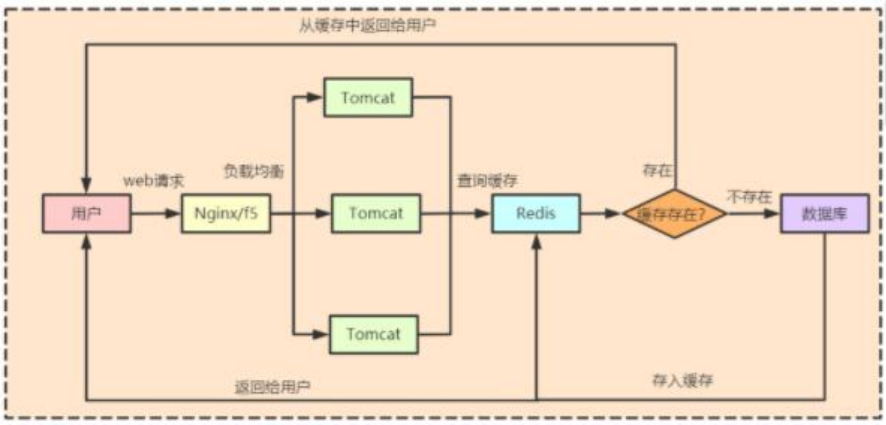
原因：

1. Redis突然宕机
2. 大部分数据失效

举个例子理解下吧：

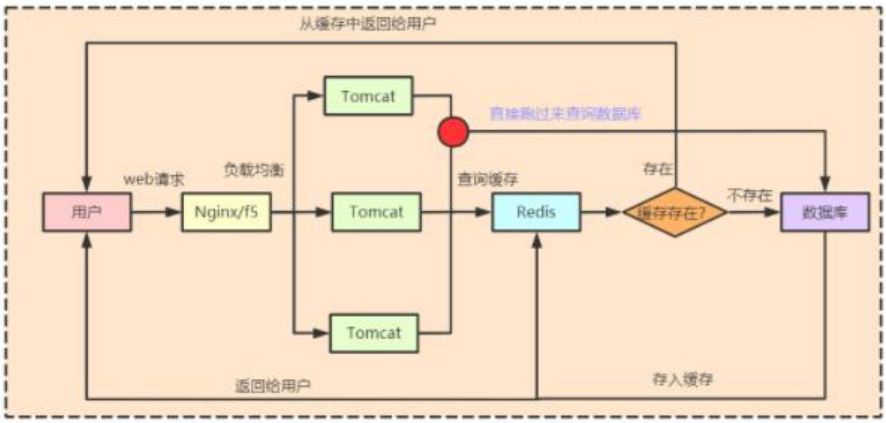
比如我们基本上都经历过购物狂欢节，假设商家举办 23:00-24:00 商品打骨折促销活动。程序小哥哥设计的时候，在 23:00 把商家打骨折的商品放到缓存中，并通过redis的expire设置了过期时间为1小。这个时间段许多用户访问这些商品信息、购买等等。但是刚好到了24:00点的时候，恰好还有许多户在访问这些商品，这时候对这些商品的访问都会落到数据库上，导致数据库要抗住巨大的压力，稍不慎会导致，数据库直接宕机（over）。

当商品没有失效的时候是这样的：



当缓存GG（失效）的时候却是这样的：

原文链接：缓存穿透、缓存击穿和缓存雪崩



对于缓存雪崩有以下解决方案：

（1）redis高可用

redis有可能挂掉，多增加几台redis实例，（一主多从或者多主多从），这样一台挂掉之后其他的还以继续工作，其实就是搭建的集群。

（2）限流降级

在缓存失效后，通过加锁或者队列来控制读数据库写缓存的线程数量，对某个key只允许一个线程查数据和写缓存，其他线程等待。

（3）数据预热

数据加热的含义就是在正式部署之前，我先把可能的数据先预先访问一遍，这样部分可能大量访问的据就会加载到缓存中。在即将发生大并发访问前手动触发加载缓存不同的key。

（4）不同的过期时间

设置不同的过期时间，让缓存失效的时间点尽量均匀。

来源：https://www.cnblogs.com/xiaowei123/p/13211403.html

作者：xiaowei123