还不懂Redis？看完这个故事就明白了！

菜鸟教程 昨天

以下文章来源于编程技术宇宙 ，作者轩辕之风O

作者 | 轩辕之风O
来源 | 编程技术宇宙

我是Redis

你好，我是**Redis**，一个叫**Antirez**的男人把我带到了这个世界上。

说起我的诞生，跟关系数据库**MySQL**还挺有渊源的。

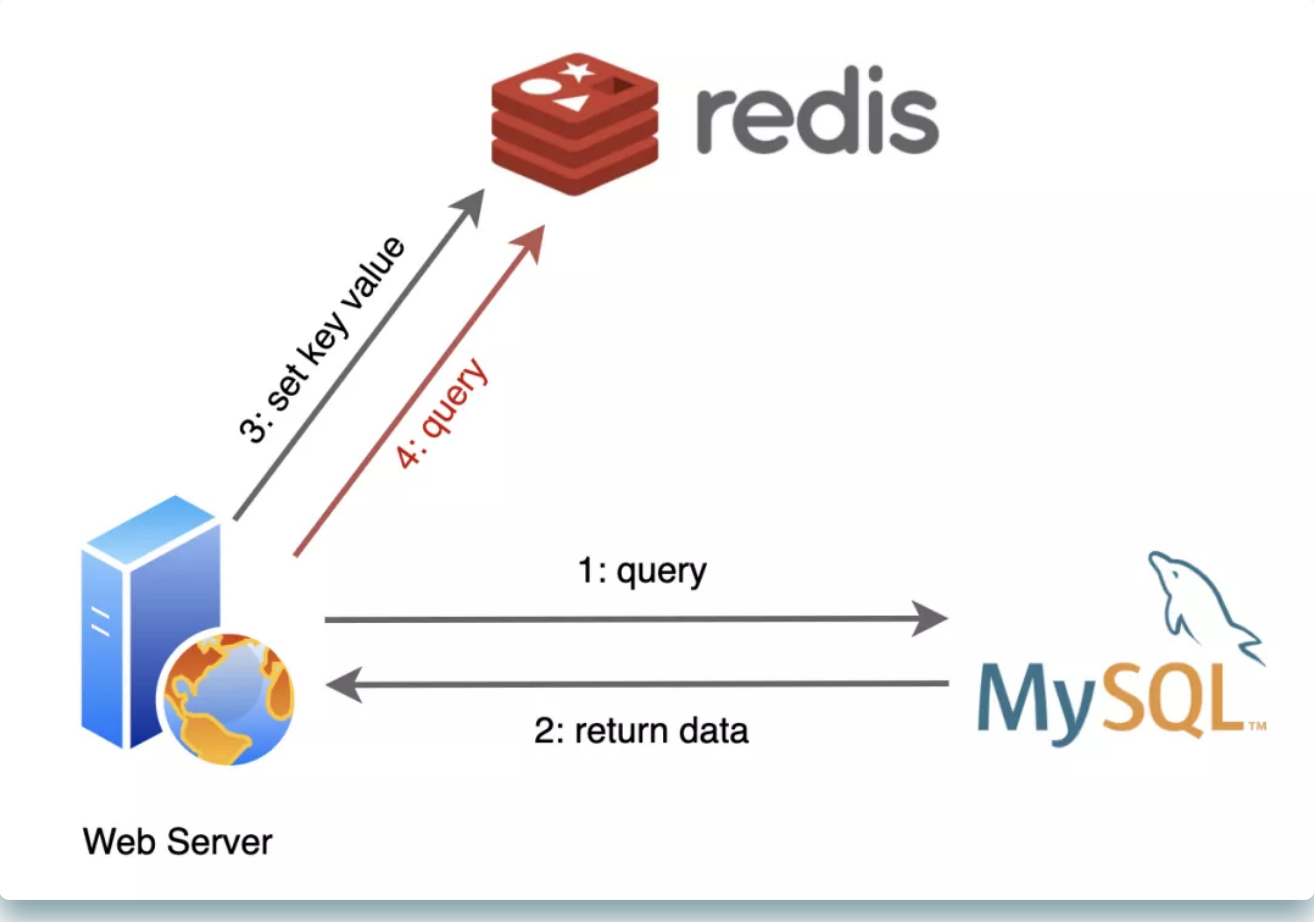
在我还没来到这个世界上的时候，MySQL过得很辛苦，互联网发展得越来越快，它容纳的数据也越来越多，用户请求也随之暴涨，而每一个用户请求都变成了对它的一个又一个读写操作，MySQL是苦不堪言。尤其是到“双11”、“618”这种全民购物狂欢的日子，都是MySQL受苦受难的日子。

据后来MySQL告诉我说，其实有一大半的用户请求都是读操作，而且经常都是重复查询一个东西，浪费它很多时间去进行磁盘I/O。

后来有人就琢磨，是不是可以学学CPU，给数据库也加一个缓存呢？于是我就诞生了！

出生不久，我就和MySQL成为了好朋友，我们俩常常携手出现在后端服务器中。

应用程序们从MySQL查询到的数据，在我这里登记一下，后面再需要用到的时候，就先找我要，我这里没有再找MySQL要。



为了方便使用，我支持好几种数据结构的存储：

- String
- Hash
- List
- Set
- SortedSet

- Bitmap
- ……

因为我把登记的数据都记录在内存中，不用去执行慢如蜗牛的I/O操作，所以找我要比找MySQL要省去了不少的时间呢。

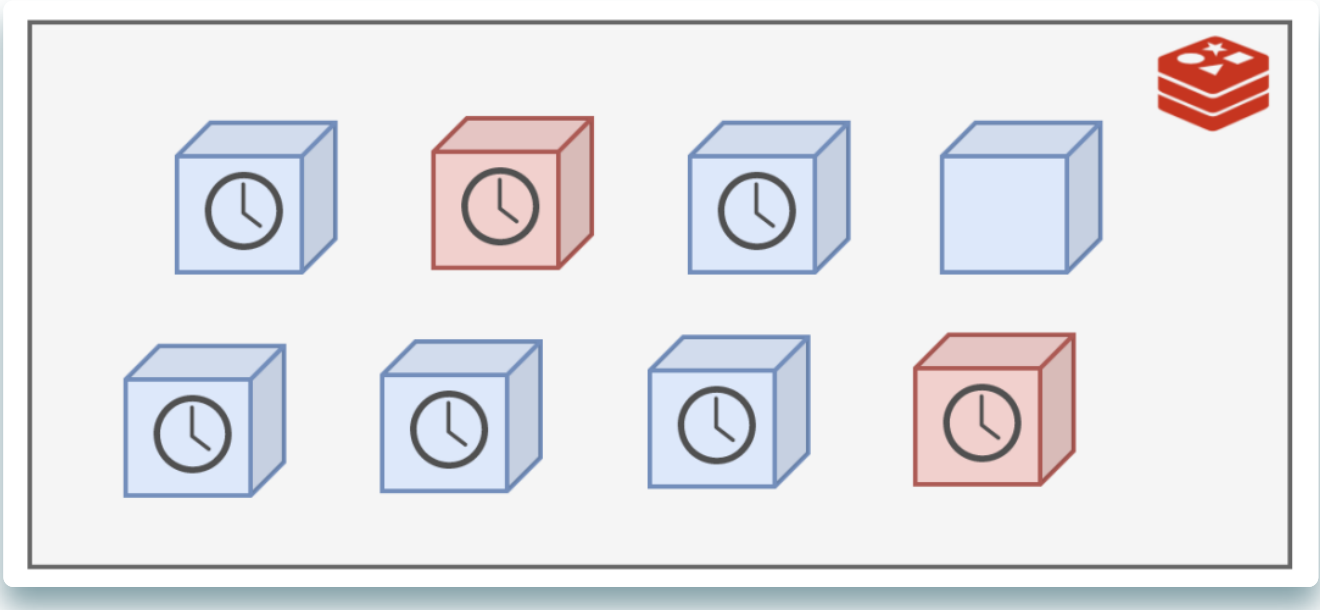
可别小瞧这简单的一个改变，我可为MySQL减轻了不小的负担！随着程序的运行，我缓存的数据越来越多，有相当部分时间我都给它挡住了用户请求，这一下它可乐得清闲自在了！

有了我的加入，网络服务的性能提升了不少，这都归功于我为数据库挨了不少枪子儿。

缓存过期 && 缓存淘汰

不过很快我发现事情不妙了，我缓存的数据都是在内存中，可是就算是在服务器上，内存的空间资源还是很有限的，不能无节制地这么存下去，我得想个办法，不然吃枣药丸。

不久，我想到了一个办法：**给缓存内容设置一个超时时间**，具体设置多长交给应用程序们去设置，我要做的就是把过期了的内容从我里面删除掉，及时腾出空间就行了。

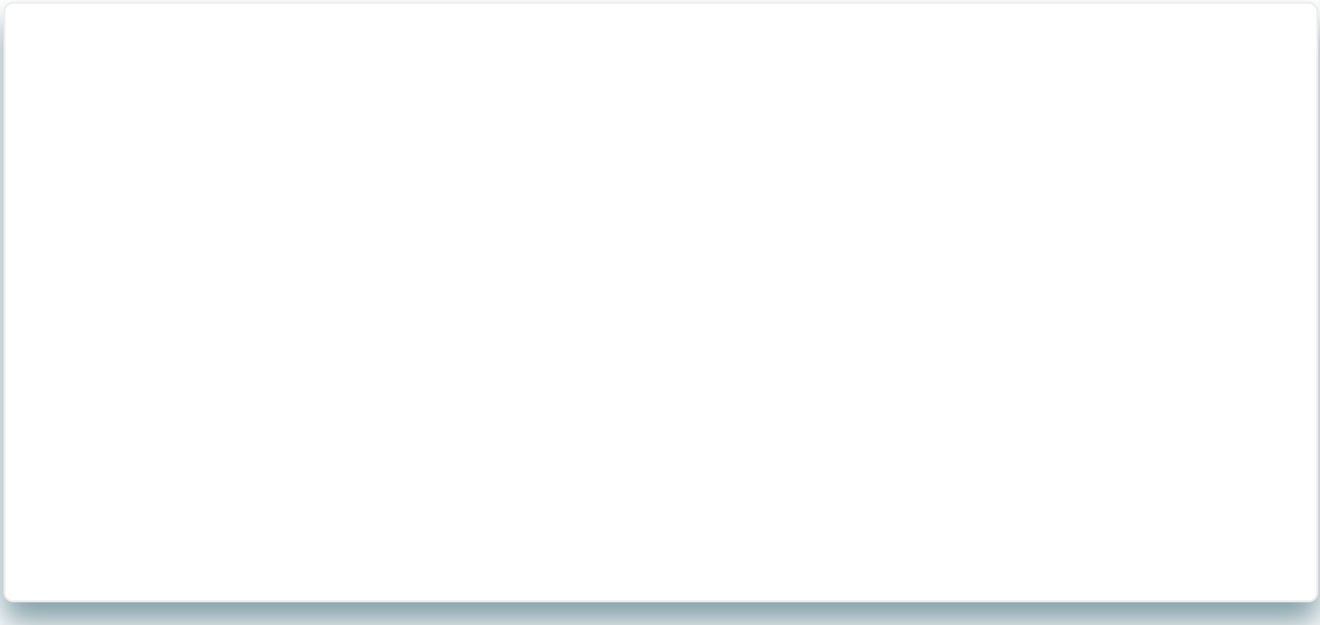


超时时间有了，我该在什么时候去干这个清理的活呢？

最简单的就是**定期删除**，我决定**100ms**就做一次，一秒钟就是10次！

我清理的时候也不能一口气把所有过期的都给删除掉，我这里面存了大量的数据，要全面扫一遍的话那不知道要花多久时间，会严重影响我接待新的客户请求的！

时间紧任务重，我只好随机选择一部分来清理，能缓解内存压力就行了。



就这样过了一段日子，我发现有些个键值运气比较好，每次都没有被我的随机算法选中，每次都能幸免于难，这可不行，这些长时间过期的数据一直霸占着不少的内存空间！气抖冷！

我眼里可揉不得沙子！于是在原来定期删除的基础上，又加了一招：

那些原来逃脱我随机选择算法的键值，一旦遇到查询请求，被我发现已经超期了，那我就绝不客气，立即删除。

这种方式因为是被动式触发的，不查询就不会发生，所以也叫**惰性删除**！

可是，还是有部分键值，既逃脱了我的随机选择算法，又一直没有被查询，导致它们一直逍遥法外！而与此同时，可以使用的内存空间却越来越少。

而且就算退一步讲，我能够把过期的数据都删除掉，那万一过期时间设置得很长，还没等到我去清理，内存就吃满了，一样要吃枣药丸，所以我还得想个办法。

我苦思良久，终于憋出了个大招：**内存淘汰策略**，这一次我要彻底解决问题！

我提供了8种策略供应用程序选择，用于我遇到内存不足时该如何决策：

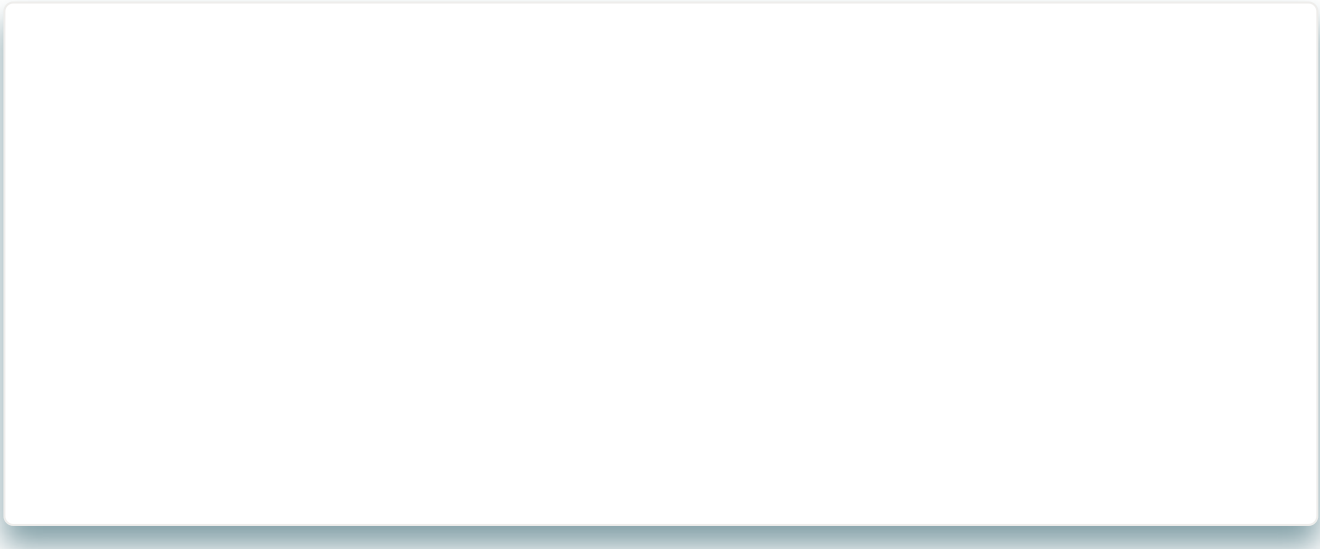
- **noeviction**：返回错误，不会删除任何键值
- **allkeys-lru**：使用LRU算法删除最近最少使用的键值
- **volatile-lru**：使用LRU算法从设置了过期时间的键集合中删除最近最少使用的键值
- **allkeys-random**：从所有key随机删除
- **volatile-random**：从设置了过期时间的键的集合中随机删除
- **volatile-ttl**：从设置了过期时间的键中删除剩余时间最短的键
- **volatile-lfu**：从配置了过期时间的键中删除使用频率最少的键
- **allkeys-lfu**：从所有键中删除使用频率最少的键

有了上面几套组合拳，我再也不用担心过期数据多了把空间撑满的问题了~

缓存穿透 && 布隆过滤器

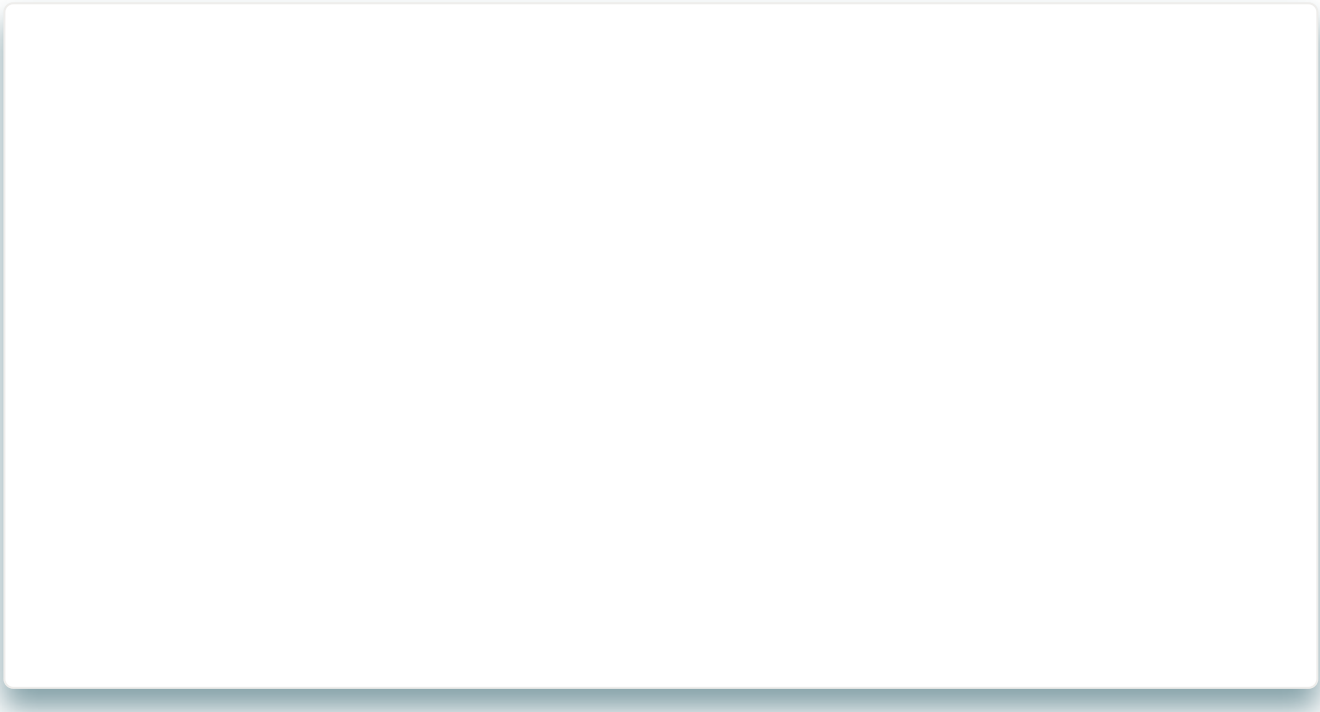
我的日子过得还挺舒坦，不过MySQL大哥就没我这么舒坦了，有时候遇到些烦人的请求，查询的数据不存在，MySQL就要白忙活一场！不仅如此，因为不存

在，我也没法缓存啊，导致同样的请求来了每次都要去让MySQL白忙活一场。我作为缓存的价值就没得到体现啦！这就是人们常说的**缓存穿透**。



这一来二去，MySQL大哥忍不住了：“唉，兄弟，能不能帮忙想个办法，把那些明知道不会有结果的查询请求给我挡一下。”

这时我想到了我的另外一个好朋友：**布隆过滤器**。



我这位朋友别的本事没有，就擅长从超大的数据集中快速告诉你查找的数据存不存在（**悄悄告诉你，我的这位朋友有一点不靠谱，它告诉你存在的话不能全信，其实有可能是不存在的，不过它要是告诉你不存在的话，那就一定不存在**）。

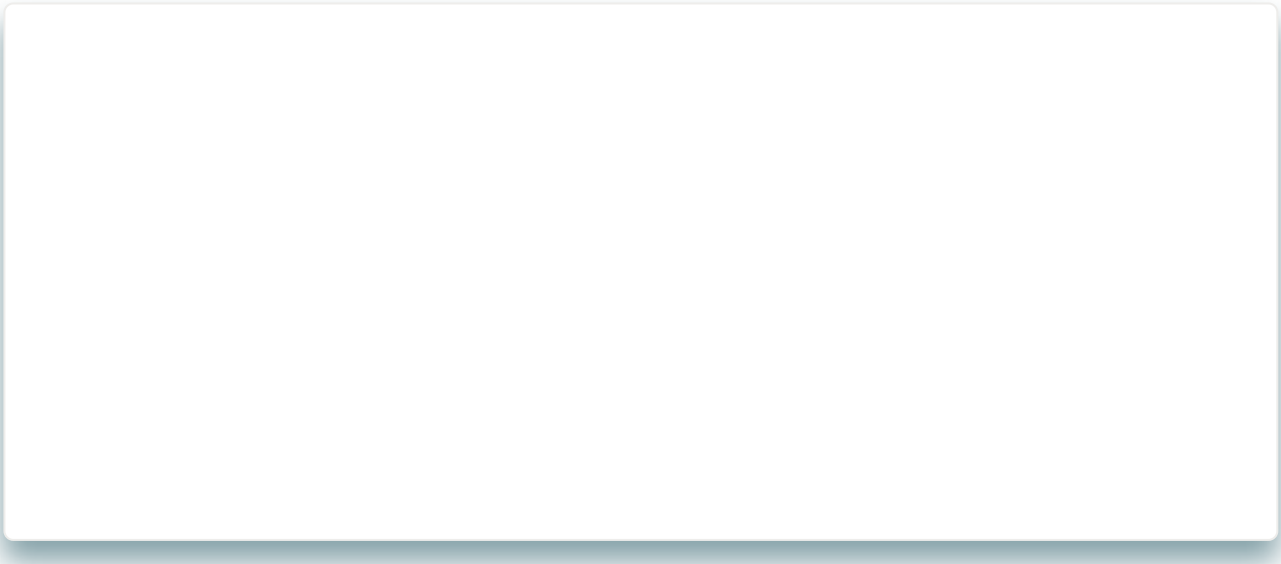
我把这位朋友介绍给了应用程序，不存在的数据就不必去叨扰MySQL了，轻松帮忙解决了缓存穿透的问题。

缓存击穿 && 缓存雪崩

这之后过了一段时间太平日子，直到那一天···

有一次，MySQL那家伙正优哉游哉地摸鱼，突然一大堆请求给他怼了过去，打了他一个措手不及。

一阵忙活之后，MySQL怒气冲冲地找到了我，“兄弟，咋回事啊，怎么一下子来得这么猛？”



我查看了日志，赶紧解释到：“大哥，实在不好意思，刚刚有一个热点数据到了过期时间，被我删掉了，不巧的是随后就有对这个数据的大量查询请求来了，我这里已经删了，所以请求都发到你那里去了。”

“你这干的叫啥事，下次注意点啊。”MySQL大哥一脸不高兴地离开了。

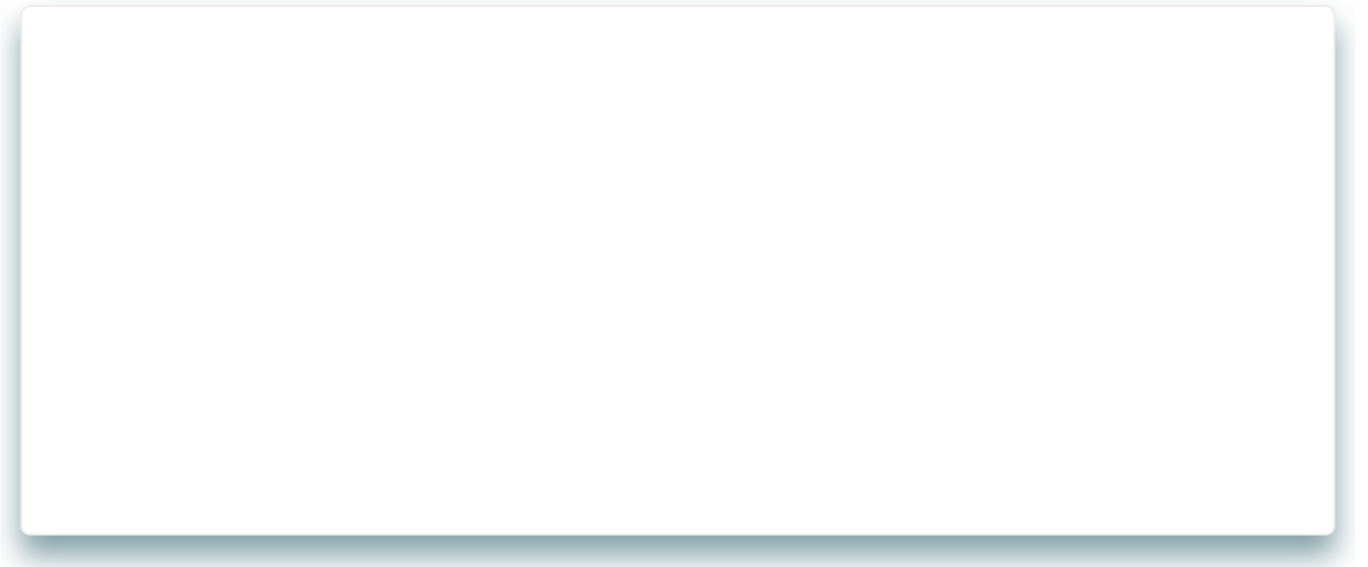
这一件小事我也没怎么放在心上，随后就抛之脑后了，却没曾想几天之后竟捅了更大的篓子。

那一天，又出现了大量的网络请求发到了MySQL那边，比上一次的规模大得多，MySQL大哥一会儿功夫就给干趴下了好几次！

等了好半天这一波流量才算过去，MySQL才缓过神来。

“老弟，这一次又是什么原因？”MySQL大哥累得没了力气。

“这一次比上一次更不巧，这一次是一大批数据几乎同时过了有效期，然后又发生了很多对这些数据的请求，所以比起上一次这规模更大了。”



MySQL大哥听了眉头一皱：“那你倒是想个办法啊，三天两头折磨我，**这谁顶得住啊？**”

“其实我也很无奈，这个时间也不是我设置的，要不我去找应用程序说说，让他把缓存过期时间设置得均匀一些？至少别让大量数据集体失效。”

“走，咱俩一起去！”

后来，我俩去找应用程序商量了，不仅把键值的过期时间随机了一下，还设置了热点数据永不过期，这个问题缓解了不少。哦对了，我们还把这两次发生的问题分别取了名字：**缓存击穿**和**缓存雪崩**。

我们终于又过上了舒适的日子···