

# 面试官别再问我Redis内存满了该怎么办了

程序员面试 5月25日

以下文章来源于非科班的科班，作者黎杜

非科班的科班

世界上并没有什么救世主，假如有那便是你自己；世界上也没有什么奇迹，假如有那只...



微信扫一扫  
关注该公众号

## 概述

Redis 的文章，我之前写过一篇关于「Redis的缓存的三大问题」，累计阅读也快800了，对于还只有3k左右的粉丝量，能够达到这个阅读量，已经是比较难了。

这说明那篇文章写的还过得去，收到很多人的阅读肯定，感兴趣的看一下[看完这篇Redis缓存三大问题，你能和面试官互扯。]。

「三大缓存问题」只是Redis的其中的一小部分的知识点，想要深入学习Redis还要学习比较多的知识点。

那么今天就带来了一个面试常问的一个问题：「假如你的Redis内存满了怎么办？」长期的把Redis作为缓存使用，总有一天会存满的时候对吧。

这个面试题不慌呀，在Redis中有配置参数 maxmemory 可以「设置Redis内存的大小」。

在Redis的配置文件 redis.conf 文件中，配置 maxmemory 的大小参数如下所示：

实际生产中肯定不是 100mb 的大小咯，不要给误导了，这里我只是让大家认识这个参数，一般小的公司都是设置为 3G 左右的大小。

除了在配置文件中配置生效外，还可以通过命令行参数的形式，进行配置，具体的配置命令行如下所示：

```

//获取maxmemory配置参数的大小
127.0.0.1:6379> config get maxmemory
//设置maxmemory参数为100mb
127.0.0.1:6379> config set maxmemory 100mb
```

倘若实际的存储中超出了Redis的配置参数的大小时，Redis中有「淘汰策略」，把「需要淘汰的key给淘汰掉，整理出干净的一块内存给新的key值使用」。

接下来我们就详细的聊一聊Redis中的淘汰策略，并且深入的理解每个淘汰策略的原理和应用的场景。

## 淘汰策略

Redis提供了「6种的淘汰策略」，其中默认的是 noeviction，这6种淘汰策略如下：

- noeviction（「默认策略」）：若是内存的大小达到阈值的时候，所有申请内存的指令都会报错。
- allkeys-lru：所有key都是使用「LRU算法」进行淘汰。
- volatile-lru：所有「设置了过期时间的key使用LRU算法」进行淘汰。
- allkeys-random：所有的key使用「随机淘汰」的方式进行淘汰。
- volatile-random：所有「设置了过期时间的key使用随机淘汰」的方式进行淘汰。
- volatile-ttl：所有设置了过期时间的key「根据过期时间进行淘汰，越早过期就越快被淘汰」。

假如在Redis中的数据有「一部分是热点数据，而剩下的数据是冷门数据」，或者「我们不太清楚我们应用的缓存访问分布状况」，这时可以使用 allkeys-lru。

假如所有的数据访问的频率大概一样，就可以使用 allkeys-random 的淘汰策略。

假如要配置具体的淘汰策略，可以在 redis.conf 配置文件中配置，具体配置如下所示：

```

# The default is:
# maxmemory-policy noeviction
```

这只需要把注释给打开就可以，并且配置指定的策略方式，另一种的配置方式就是命令的方式进行配置，具体的执行命令如下所示：

```

// 获取maxmemory-policy配置
127.0.0.1:6379> config get maxmemory-policy
// 设置maxmemory-policy配置为allkeys-lru
127.0.0.1:6379> config set maxmemory-policy allkeys-lru
```

在介绍6种的淘汰策略方式的时候，说到了LRU算法，「那么什么是LRU算法呢？」

## LRU算法

LRU(Least Recently Used)即表示最近最少使用，也就是在最近的时间内最少被访问的key，算法根据数据的历史访问记录来进行淘汰数据。

它的核心的思想就是：「假如一个key值在最近很少被使用到，那么在将来也很少会被访问」。

实际上Redis实现的LRU并不是真正的LRU算法，也就是名义上我们使用LRU算法淘汰键，但是实际上被淘汰的键并不一定是真正的最久没用的。

Redis使用的是近似的LRU算法，「通过随机采集淘汰key，每次都会随机选出5个key，然后淘汰里面最近最少使用的key」。

这里的5个key只是默认的个数，具体的个数也可以在配置文件中配置，在配置文件中的配置如下图所示：

```

# The default of 5 produces good enough results. 10 Approximates very closely
# true LRU but costs more CPU. 3 is faster but not very accurate.
# maxmemory-samples 5
```

当近似LRU算法取值越大的时候就会越接近真实的LRU算法，可以这样理解，因为「取值越大那么获取的数据就越全，淘汰中的数据的就越接近最近最少使用的数据」。

那么为了实现根据时间实现LRU算法，Redis必须为每个key中额外的增加一个内存空间用于存储每个key的时间，大小是3字节。

在Redis 3.0中对近似的LRU算法做了一些优化，Redis中会维护大小是 16 的一个候选池的内存。

当第一次随机选取的采样数据，数据都会被放进候选池中，并且候选池中的数据会根据时间进行排序。

当第二次以后选取的数据，只有「小于候选池内的最小时间」的才会被放进候选池中。

当某一时刻候选池的数据满了，那么时间最大的key就会被挤出候选池。当执行淘汰时，直接从候选池中选取最近访问时间最小的key进行淘汰。

这样做的目的就是选取出最近拟合最近最少被访问的key值，能够正确的淘汰key值，因为随机选取的样本中的最小时间可能不是真正意义上的最小时间。

但是LRU算法有一个弊端：就是假如一个key值在以前都没有被访问到，然而最近一次被访问到了，那么就会认为它是热点数据，不会被淘汰。

然而有些数据以前经常被访问到，只是最近的时间内没有被访问到，这样就导致这些数据很可能被淘汰掉，这样一来就会出现误判而淘汰热点数据。

于是在Redis 4.0的时候除了LRU算法，新加了一种LFU算法，「那么什么是LFU算法算法呢？」

## LFU算法

LFU(Least Frequently Used)即表示最近频繁被使用，也就是最近的时间段内，频繁被访问的key，它以最近的时间段的被访问次数的频率作为一种判断标准。

它的核心思想就是：根据key最近被访问的频率进行淘汰，比较少被访问的key优先淘汰，反之则优先保留。

LFU算法反映了一个key的热度情况，不会因为LRU算法的偶尔一次被访问被认为是热点数据。

在LFU算法中支持 volatile-lfu 策略和 allkeys-lfu 策略。

以上介绍了Redis的6种淘汰策略，这6种淘汰策略旨在告诉我们怎么做，但是什么时候做？这个还没说，下面我们就来详细的了解Redis什么时候执行淘汰策略。

## 删除过期键策略

在Redis中有三种删除的操作此策略，分别是：

- 「定时删除」：创建一个定时器，定时的执行对key的删除操作。
- 「惰性删除」：每次只有再访问key的时候，才会检查key的过期时间，若是已经过期了就执行删除。
- 「定期删除」：每隔一段时间，就会检查删除掉过期的key。

「定时删除」对于「内存来说是友好的」，定时清理出干净的空间，但是对于「cpu来说并不是友好的」，程序需要维护一个定时器，这就会占用cpu资源。

「惰性的删除」对于「cpu来说是友好的」，cpu不需要维护其它额外的操作，但是对于「内存来说是不友好的」，因为要是有些key一直没有被访问到，就会一直占用着内存。

定期删除是上面两种方案的折中方案\*\*，每隔一段时间删除过期的key，也就是根据具体的业务，合理的取一个时间定期的删除key\*\*。

通过「最合理控制删除的时间间隔」来删除key，减「少对cpu的资源占用消耗」，使删除操作合理化。

## RDB和AOF 的淘汰处理

在Redis中持久化的方式有两种 RDB 和 AOF，具体这两种详细的持久化介绍，可以参考这一篇文章[面试造飞机系列：面对Redis持久化连环Call，你还顶得住吗？]。

在RDB中是以快照的形式获取内存中某一时间点的数据副本，在创建RDB文件的时候可以通过 save 和 bgsave 命令执行创建RDB文件。

「这两个命令都不会把过期的key保存到RDB文件中」，这样也能达到删除过期key的效果。

当在启动Redis载入RDB文件的时候， Master 不会把过期的key载入，而 Slave 会把过期的key载入。

在AOF模式下，Redis提供了Rewrite的优化措施，执行的命令分别是 REWRITEAOF 和 BGREWRITEAOF，「这两个命令都不会把过期的key写入到AOF文件中，也能删除过期key」。

长按订阅更多面经分享

