目录

[1.线上出现redis大key，如何删除 1](#_Toc128254722)

[方案一（业务低峰期删除） 1](#_Toc128254723)

[方案二（ 1](#_Toc128254724)

[方案三（异步删除法） 2](#_Toc128254725)

[经典生产问题 2](#_Toc128254726)

[redis大 key基本概念及常见场景 2](#_Toc128254727)

[大key常见场景 3](#_Toc128254728)

[大key的产生往往是业务方设计不合理，没有预见 vaule的动态增长问题. 3](#_Toc128254729)

[接下来我们看看几类比较经典的场景: 3](#_Toc128254730)

[Redis大key带来的影响 3](#_Toc128254731)

[Redis大key如何检测 4](#_Toc128254732)

[Redis大可以如何删除 4](#_Toc128254733)

[1)分批次渐进式删除 4](#_Toc128254734)

[2)采用unlink + bigkey异步非阻塞删除。这个命令是在redis 4.0+提供的代替del命令，不会阻塞主线程。 6](#_Toc128254735)

# 1.线上出现redis大key，如何删除

大key本身就是一个线程，如果执行执行大key可能会造成阻塞。阻塞期间所有的请求都可能超时，当超时越多，新的请求不断进来，这样会造成redis连接池耗尽，引发线上各种依赖redis的业务出现异常。

## 方案一（业务低峰期删除）

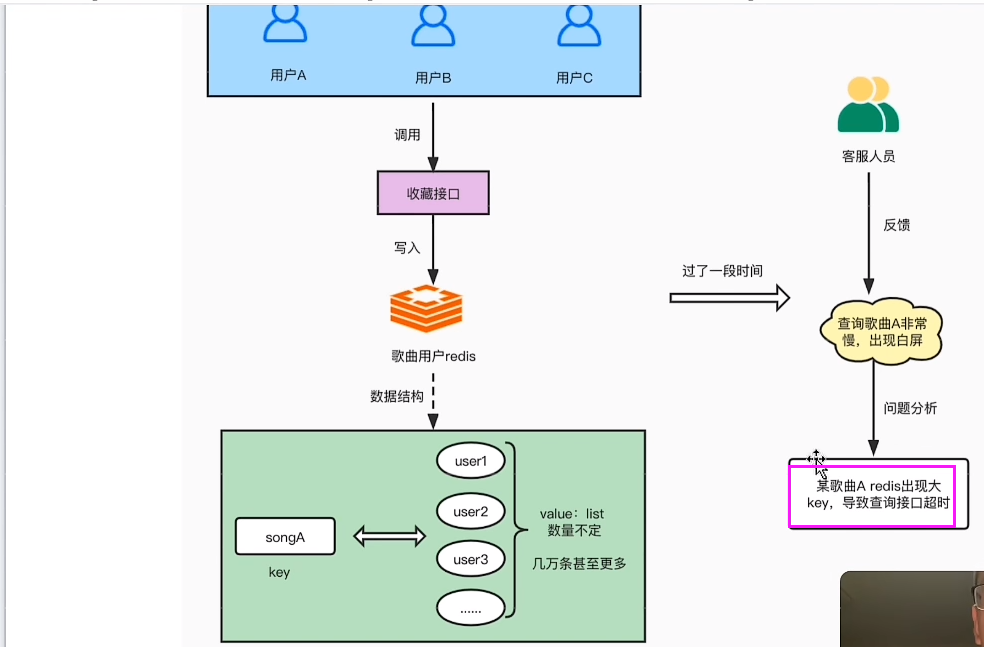
业务低峰期删除，造成影响小。无法避免阻塞，适合执行期间QPS非常小的任务。

方案二（scan分批删除法）  
scan分批删除法，最常见的大key，一般都是hash，set，zset，list，删除的时候注意分批。对于hash可以用scan扫面法删除，对于集合可以通过srandmember每次随机获取一批数据然后删除，对于有序集合可以通过zremrangbyrank直接删除一批数据，对于列表，可以直接pop出数据即可

## 方案三（异步删除法）

异步删除法，通过异步线程执行删除操作，用unlink代替del执行删除操作

## 经典生产问题



某音乐节，收到客服反馈通知，说APP查询某爆款歌曲收藏的用户列表非常缓慢。于是开发人员找到一个出问题的爆款歌曲A，通过搜索日志系统找到traceld，根据traceld 从我们的skywalking 分布式调用链路系统跟踪发现，是操作redis 时间比较久，并且从日志系统搜到一些redis查询超时的异常。最后我们定位到的原因如下:

在收藏歌曲接口的时候拿redis做了一个缓存，记录收藏该歌曲的用户列表, redis数据结构: key = song,value =用户id列表,而redis 查询发现多了许多大 key，体现在一个爆款歌曲儿力甚主史多的用尸收臧，导致Hedis啊应缓慢，查调经常超时，导敛服务这边出现服务器异常,APP页面没拿到接口的数据从而出现白屏的现象。

## redis大 key基本概念及常见场景

很多朋友肯定在想reds 的 key能有多大呀?这里就有个误区了，所谓的大key问题是某个key对应的value 比较天，所以本质上是大value问题。key往往是开发过程中可以自行设置，可以控制大小, value往往不受程序控制跟业务场景有关系，因此可能导致value很大。

在Redis中，大key指的是key对应的value值所占的内存空间比较大。value是string类型，大小控制在10Kkb 以内

value是hash、list、set、zset等集合类型，元素个数不要超过5000(或则1万、几万)上述的定义并不绝对，主要是根据value 的大小和元素个数来确定，业务也可以根据自己的场景确定标准。

## 大key常见场景

### 大key的产生往往是业务方设计不合理，没有预见 vaule的动态增长问题.

* 一直往value塞数据，没有删除及过期机制,迟早要爆炸
* 数据没有合理做分片，将大key变成以一个个小key

### 接下来我们看看几类比较经典的场景:

社交类:如果某些明星或者大v的粉丝列表不精心设计下，必是 bigkey.

统计类:例如统计某游戏活动玩家用户的榜单列表,除非没几个人完,否则必是bigkey.

缓存类:将数据从数据库load出来序列化放到Redis里，这个方式非常常用，但有两个地方需要注意:

第一，是不是有必要把所有字段都缓存

第二，有没有相关联的数据，关联数据分开存储

例如:遇到过一个例子，该同学将某明星一个专辑下所有视频信息都缓存一个巨大的json中，造成这个 json达到6MB，后来这个明星发了一个官宣。用户浏览专辑，因为redis大 key 瞬间扛不住了.

## Redis大key带来的影响

* 客户端超时阻塞。由于Redis单线程的特性，操作bigkey的通常比较耗时，也就意味着

阻塞Redis可能性越大，这样会造成客户端阻塞或者引起故障切换，会出现各种 redis慢查询中。

* 内存空间不均剁集群模式在slot 分片均匀情况下，会出现数据和查询倾斜情况，部分

有大key 的Redis节点占用内存多，QPS高。

* 引发网络阻塞。每次获取大key产生的网络流量较大,如果一个key的大小为1MB，每

秒访问量为1000，那么每秒会产生1000MB 的流量。这对于普通千兆网卡的服务器来说是灾难性的。

* 阻塞工作线程。执行大key删除时，在低版本redis 中可能阻塞线程。

## Redis大key如何检测

* 改写redis客户端，在sdk中加入埋点，实时上报数据给redis 大key〉测平台、监控

告警。

* scan + debug object bigkey命令。循环遍历redis key序列化后的长度.debug object

bigkey可能会比较慢，它存在阻塞Redis的可能，建议在从节点执行该命令，官方不推荐。

* scan + memory usage。该命令是在Redis 4.0+以后提供的，可以循环遍历统计计算每

个键值的字节数。

* 通过python脚本迭代的scan key，对每次scan 的内容进行判断是否大key.
* redis-cli --bigkeys。可以找到某个redis实例5种数据类型(string、hash、list、set,

zset)的最大key。但如果redis key比较多，执行该命令会比较慢，建议在从节点执行该命令。

* rdbtools开源工具包。rdbtools是 python写的一个第三方开源工具，用来解析Redis

快照文件, redis实例上执行bgsave，然后对dump出来的rdb文件进行分析，找到其中的大key.例如: rdb dump.rdb -c memory --bytes 10240 -f redis.csv；从dump.rdb 快照文件统计(bgsave),将所有>10kb 的 key输出到一个csv文件

## Redis大可以如何删除

如果对这类大key直接使用del 命令进行删除,会导致长时间阻塞,甚至崩溃。因为 del命令在删除集合类型数据时，时间复杂度为O(M),M是集合中元素的个数。Redis是单线程的，单个命令执行时间过长就会阻塞其他命令，容易引起雪崩。那我们怎么解决呢?√主动删除大 key

### 1)分批次渐进式删除

一般来说，对于string数据类型使用del命令不会产生阻塞。其它数据类型分批删除，通过scan命令遍历大kel，每次取得少部分元素进行删除，然后再获取和删除下一批元素。对Hash, Sorted Set, List,Set分别处理，思路相同，先对key改名进行逻辑删除，使客户端无法使用原key，然后使用批量小步删除。

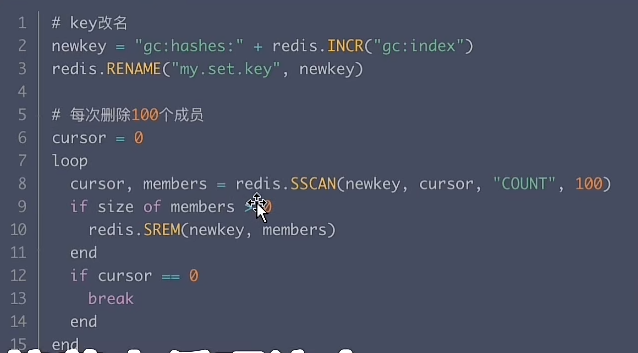
#### 删除大Hash



#### 删除大list



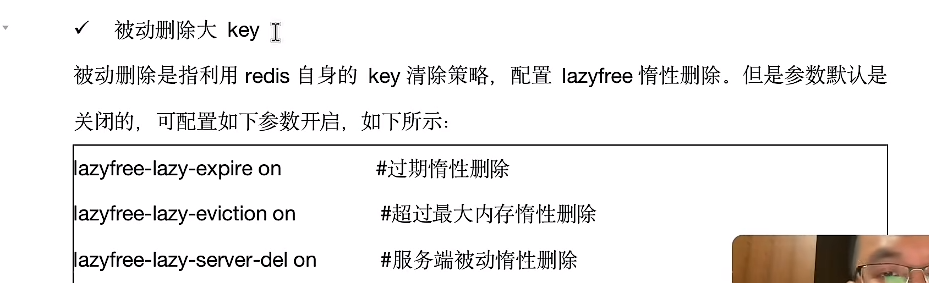
#### 删除大set



#### 删除大Sorted set

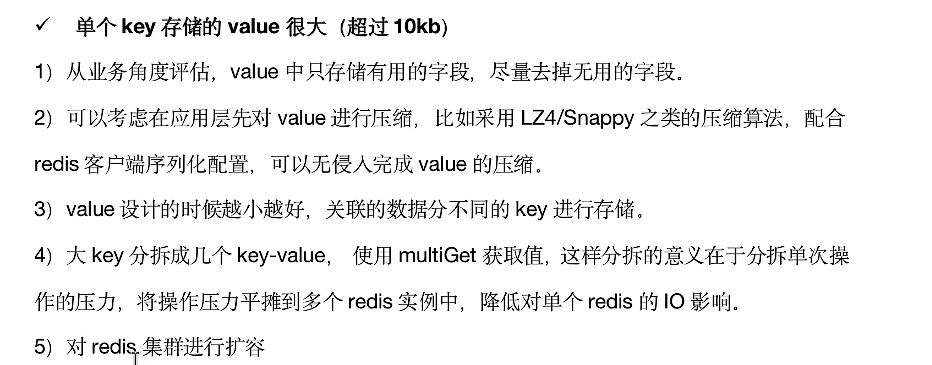


### 2)采用unlink + bigkey异步非阻塞删除。这个命令是在redis 4.0+提供的代替del命令，不会阻塞主线程。



## Redis大key如何设计和优化

### 单个key存储的value很大



### 集合数据类型hash，list，set，sorted set等存储过多的元素（超过5000）个

以hash为例，原先的正常存取流程是hget(hashKey, field) ; hset(hashKey, field, value)现在,我们可以分拆构建一个新的newHashKey,具体做法:固定一个桶的数量,比如10000,每次存取的时候， 先在本地计算field的 hash值，取模10000，确定了该fiel落在哪个newHashKey 上。

