# Redis内存管理策略

## 为什么需要内存回收？

1、在Redis中，set指令可以指定key的过期时间，当过期时间到达以后，key就失效了；

2、Redis是基于内存操作的，所有数据都是保存在内存中，一台机器的内存是有限且宝贵的

基于以上两点，为了保证Redis能继续提供可靠的服务，Redis需要一种机制清理掉不常用的，无效的，多余的数据，失效后的数据需要及时清理，这就需要内存回收了

## Redis的内存回收机制

Redis的内存回收主要分为过期删除策略和内存淘汰策略

### 过期删除策略

删除达到过期时间的Key

#### 1、定时删除

对于每一个设置了过期时间的key都会创建一个定时器，一旦到达过期时间就立即删除。该策略可以立即清除过期数据，对内存友好，但是缺点是占用大量的CPU资源去处理过期的数据，会影响Redis的吞吐量和响应时间

#### 2、惰性删除

当访问一个key时，才判断Key是否过期，过期则删除，改策略能最大限度地节省CPU资源，但是对内存却十分不友好。有一种极端的情况是可能出现大量的过期Key没有被再次访问，因此不会被清楚，导致占用了大量的内存

#### 3、定期删除

每隔一段时间，扫描Redis中过期的Key字典，并清除部分过期的Key。该策略是前两者的一种折中方案，还可以通过调整定时扫描的时间间隔和每次扫描的限定耗时，在不同情况下使得CPU和内存资源达到最优的平衡效果。

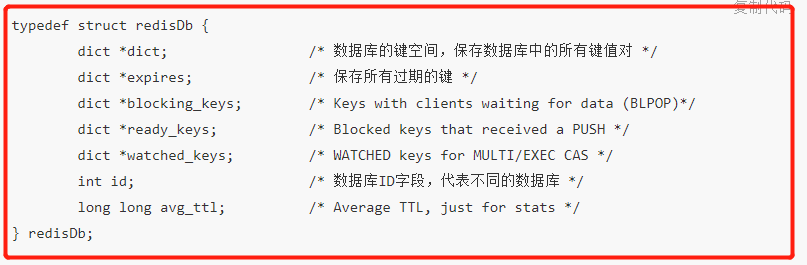
**在Redis中，同时使用了定期删除和惰性删除。**

## 过期删除策略原理

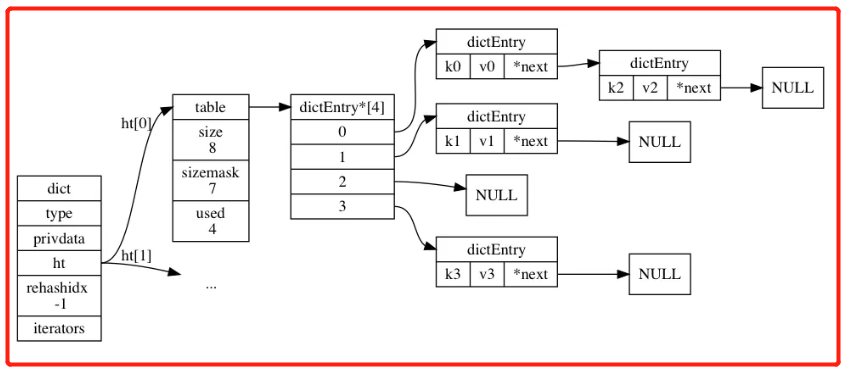
为了大家听起来不会觉得疑惑，在正式介绍过期删除策略原理之前，先给大家介绍一点可能会用到的相关Redis基础知识。

### redisDb结构体定义

我们知道，Redis是一个键值对数据库，对于每一个redis数据库，redis使用一个redisDb的结构体来保存，它的结构如下：



从结构定义中我们可以发现，对于每一个Redis数据库，都会使用一个字典的数据结构来保存每一个键值对，dict的结构图如下：



以上就是过期策略实现时用到比较核心的数据结构。程序=数据结构+算法，介绍完数据结构以后，接下来继续看看处理的算法是怎样的。

### expires属性

redisDb定义的第二个属性时expires，它的类型也是字典，Redis会把所有过期的键值对加入到expires，之后通过定期删除来清理expires里面的值，加入expires的场景有：

1. set指定过期时间expire

如果在设置Key的时候指定了过期时间，Redis会将这个Key直接加入到expires字典中

1. 调用expire命令

显式指定某个key的过期时间

1. 恢复或修改数据

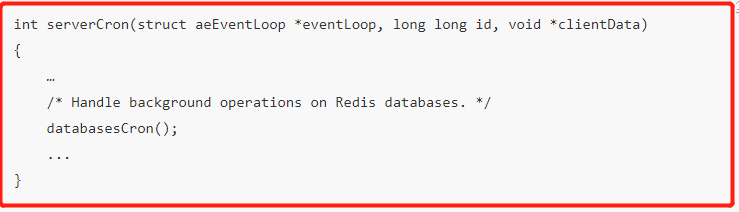
从Redis持久化文件中恢复文件或修改Key，如果数据中的Key已经设置了过期时间，就将这个Key加入到expires字典中。

以上这些操作都会将过期的key保存到expires。redis会定期从expires字典中清理过期的key。

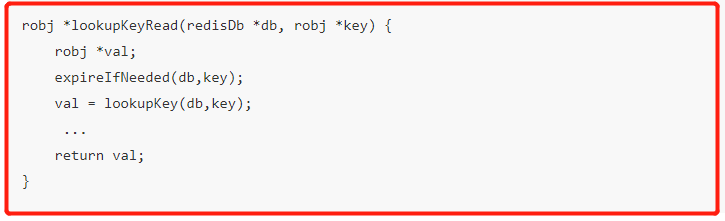
### Redis清除Key的时机

1、Redis在启动的时候，会注册两种事件，一种是文件事件，一种是时间事件。时间事件主要是Redis处理后台操作的一类事件，比如客户端超时，删除过期Key;文件事件是处理请求。

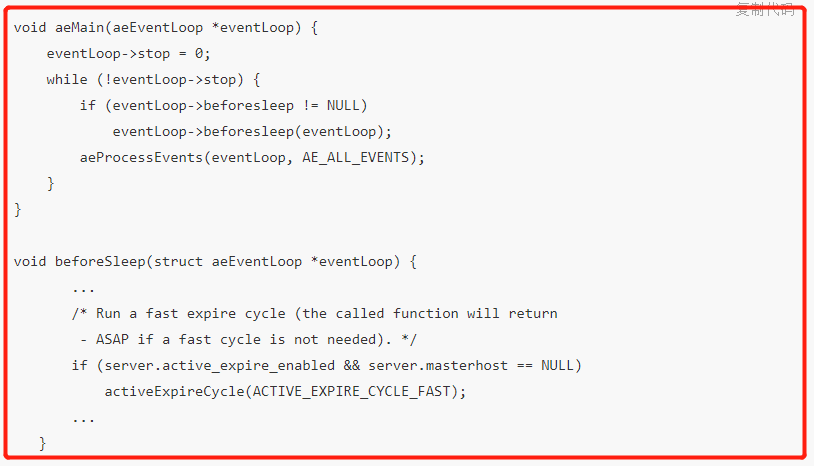
在时间事件中，redis注册的回调函数是serverCron，在定时任务回调函数中，通过调用databaseCron清理部分Key（这里是定期删除实现的）



2、每次访问key的时候，都会调用expirefNeeded函数判断Key是否过期，如果是，清理Key。（这里是惰性删除的实现）

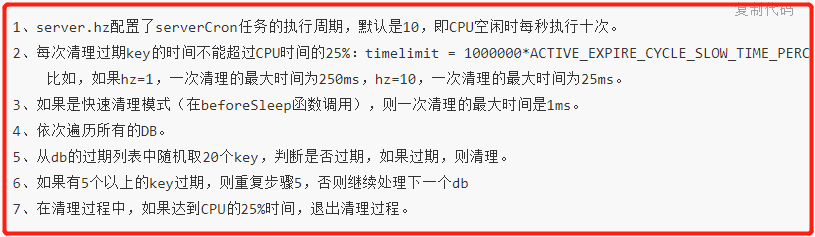


1. 每次事件循环执行时，主动清理部分过期key。（这也是惰性删除的实现。）



### 过期策略的实现

我们知道，Redis是以单线程运行的，在清理key是不能占用过多的时间和CPU，需要在尽量不影响正常的服务情况下，进行过期key的清理。过期清理的算法如下：



从实现的算法中可以看出，这只是基于概率的简单算法，且是随机的抽取，因此是无法删除所有的过期key，通过调高hz参数可以提升清理的频率，过期key可以更及时的被删除，但hz太高会增加CPU时间的消耗。

## 内存淘汰策略

Redis的内存淘汰策略，是指内存达到maxmemory极限时，使用某种算法来决定清理掉哪些数据，以保证新数据的存入。

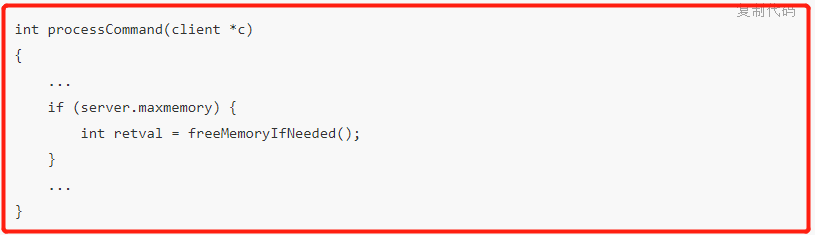
### Redis内存淘汰策略

* noeviction: 当内存不足以容纳新写入数据时，新写入操作会报错。
* allkeys-lru:当内存不足以容纳新写入数据时，在键空间（server.db[i].dict）中，移除最近最少使用的 key（这个是最常用的）。
* allkeys-random:当内存不足以容纳新写入数据时，在键空间(server.db[i].dict）中，随机移除某个key
* volatile-lru: 当内存不足以容纳新写入数据时，在设置了过期时间的键空间（server.db[i].expires）中，移除最近最少使用的 key。
* volatile-random:当内存不足以容纳新写入数据时，在设置了过期时间的键空间（server.db[i].expires）中，随机移除某个 key。
* volatile-ttl:当内存不足以容纳新写入数据时，在设置了过期时间的键空间（server.db[i].expires）中，有更早过期时间的 key 优先移除。

在配置文件中，通过maxmemory-policy可以配置要使用哪一个淘汰机制。

### 什么时候会进行淘汰？

Redis会在每一次处理命令的时候（processCommand函数调用freeMemoryIfNeeded）判断当前redis是否达到了内存的最大限制，如果达到限制，则使用对应的算法去处理需要删除的key。伪代码如下：



### LRU实现原理

在淘汰key时，Redis默认最常用的是LRU算法（Latest Recently Used）。Redis通过在每一个redisObject保存lru属性来保存key最近的访问时间，在实现LRU算法时直接读取key的lru属性。