## 20-删除数据后，为什么内存占用率还是很高？

你好，我是蒋德钧。

在使用Redis时，我们经常会遇到这样一个问题：明明做了数据删除，数据量已经不大了，为什么使用top命令查看时，还会发现Redis占用了很多内存呢？

实际上，这是因为，当数据删除后，Redis释放的内存空间会由内存分配器管理，并不会立即返回给操作系统。所以，操作系统仍然会记录着给Redis分配了大量内存。

但是，这往往会伴随一个潜在的风险点：Redis释放的内存空间可能并不是连续的，那么，这些不连续的内存空间很有可能处于一种闲置的状态。这就会导致一个问题：虽然有空闲空间，Redis却无法用来保存数据，不仅会减少Redis能够实际保存的数据量，还会降低Redis运行机器的成本回报率。

打个形象的比喻。我们可以把Redis的内存空间比作高铁上的车厢座位数。如果高铁的车厢座位数很多，但运送的乘客数很少，那么，高铁运行一次的效率低，成本高，性价比就会降低，Redis也是一样。如果你正好租用了一台16GB内存的云主机运行Redis，但是却只保存了8GB的数据，那么，你租用这台云主机的成本回报率也会降低一半，这个结果肯定不是你想要的。

所以，这节课，我就和你聊聊Redis的内存空间存储效率问题，探索一下，为什么数据已经删除了，但内存却闲置着没有用，以及相应的解决方案。

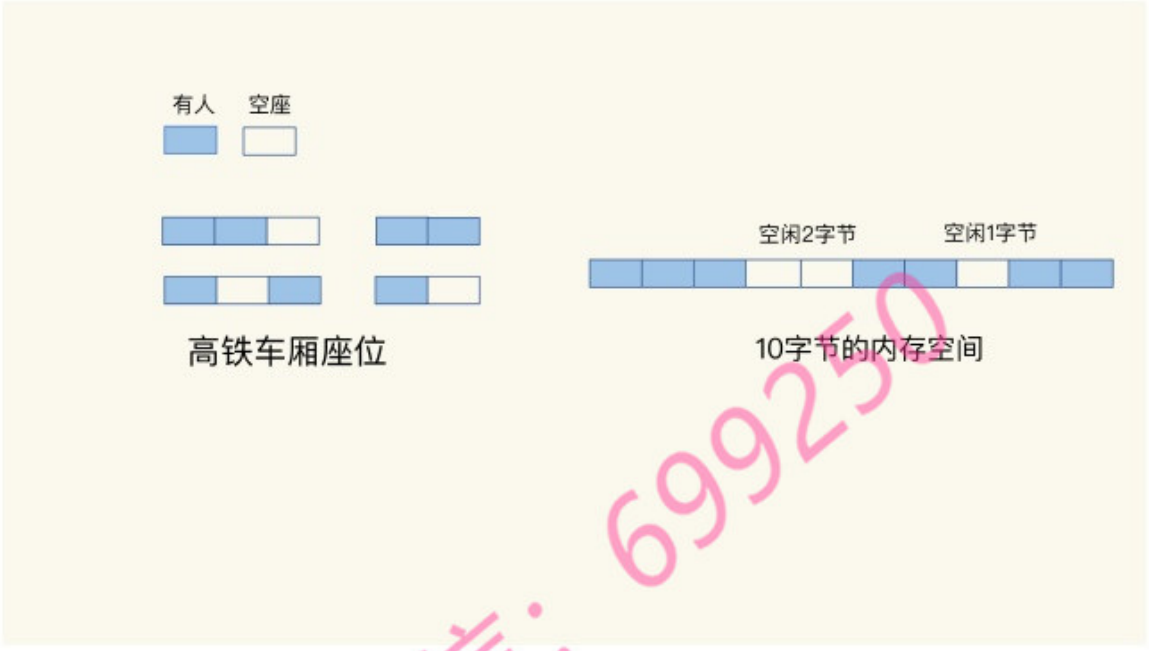


### 什么是内存碎片？

通常情况下，内存空间闲置，往往是因为操作系统发生了较为严重的内存碎片。那么，什么是内存碎片呢？

为了方便你理解，我还是借助高铁的车厢座位来进行解释。假设一个车厢的座位总共有60个，现在已经卖了57张票，你和2个小伙伴要乘坐高铁出门旅行，刚好需要三张票。不过，你们想要坐在一起，这样可以在路上聊天。但是，在选座位时，你们却发现，已经买不到连续的座位了。于是，你们只好换了一趟车。这样一来，你们需要改变出行时间，而且这趟车就空置了三个座位。

其实，这趟车的空座位是和你们的人数相匹配的，只是这些空座位是分散的，如下图所示：






我们可以把这些分散的空座位叫作“车厢座位碎片”，知道了这一点，操作系统的内存碎片就很容易理解了。虽然操作系统的剩余内存空间总量足够，但是，应用申请的是一块连续地址空间的N字节，但在剩余的内存空间中，没有大小为N字节的连续空间了，那么，这些剩余空间就是内存碎片（比如上图中的“空闲2字节”和“空闲1字节”，就是这样的碎片）。

那么，Redis中的内存碎片是什么原因导致的呢？接下来，我带你来具体看一看。我们只有了解了内存碎片的成因，才能对症下药，把Redis占用的内存空间充分利用起来，增加存储的数据量。

### 内存碎片是如何形成的？

其实，内存碎片的形成有内因和外因两个层面的原因。简单来说，内因是操作系统的内存分配机制，外因是Redis的负载特征。

### 内因：内存分配器的分配策略

内存分配器的分配策略就决定了操作系统无法做到“按需分配”。这是因为，内存分配器一般是按固定大小来分配内存，而不是完全按照应用程序申请的内存空间大小给程序分配。

Redis可以使用libc、jemalloc、tcmalloc多种内存分配器来分配内存，默认使用jemalloc。接下来，我就以jemalloc为例，来具体解释一下。其他分配器也存在类似的问题。

jemalloc的分配策略之一，是按照一系列固定的大小划分内存空间，例如8字节、16字节、32字节、48字节，…，2KB、4KB、8KB等。当程序申请的内存最接近某个固定值时，jemalloc会给它分配相应大小的空间。

这样的分配方式本身是为了减少分配次数。例如，Redis申请一个20字节的空间保存数据，jemalloc就会分配32字节，此时，如果应用还要写入10字节的数据，Redis就不用再向操作系统申请空间了，因为刚才分配的32字节已经够用了，这就避免了一次分配操作。

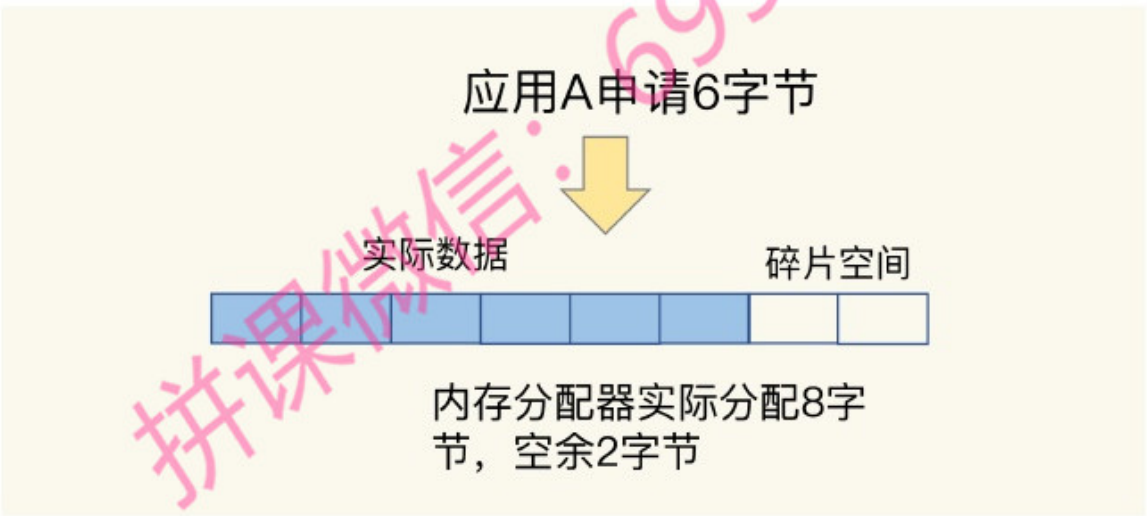
但是，如果Redis每次向分配器申请的内存空间大小不一样，这种分配方式就会有形成碎片的风险，而这正

### 外因：键值对大小不一样和删改操作

Redis通常作为共用的缓存系统或键值数据库对外提供服务，所以，不同业务应用的数据都可能保存在Redis中，这就会带来不同大小的键值对。这样一来，Redis申请内存空间分配时，本身就会有大小不一的空间需求。这是第一个外因。

但是咱们刚刚讲过，内存分配器只能按固定大小分配内存，所以，分配的内存空间一般都会比申请的空间大一些，不会完全一致，这本身就会造成一定的碎片，降低内存空间存储效率。

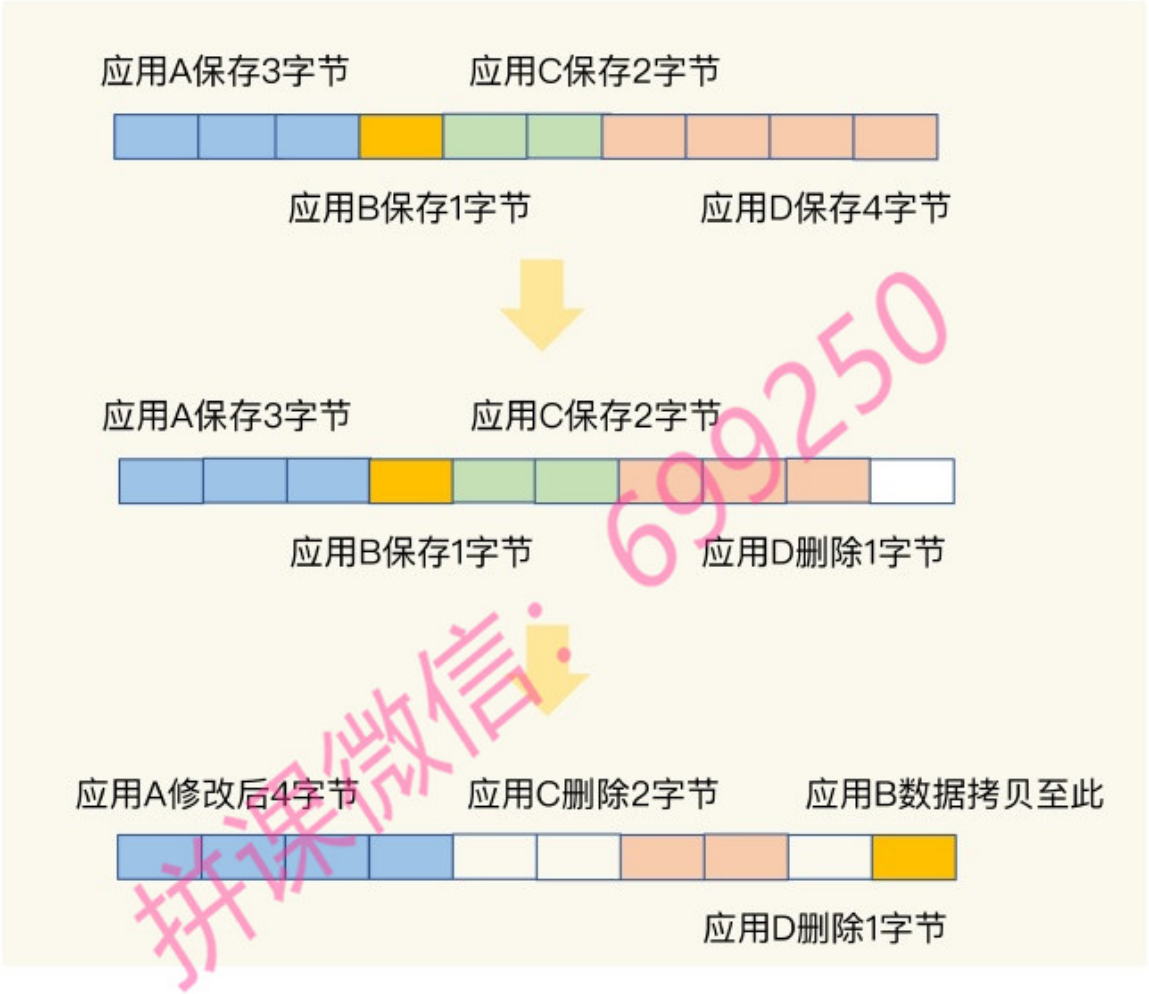
比如说，应用A保存6字节数据，jemalloc按分配策略分配8字节。如果应用A不再保存新数据，那么，这里多出来的2字节空间就是内存碎片了，如下图所示：






第二个外因是，这些键值对会被修改和删除，这会导致空间的扩容和释放。具体来说，一方面，如果修改后的键值对变大或变小了，就需要占用额外的空间或者释放不用的空间。另一方面，删除的键值对就不再需要内存空间了，此时，就会把空间释放出来，形成空闲空间。

我画了下面这张图来帮助你理解。





一开始，应用A、B、C、D分别保存了3、1、2、4字节的数据，并占据了相应的内存空间。然后，应用D删除了1个字节，这个1字节的内存空间就空出来了。紧接着，应用A修改了数据，从3字节变成了4字节。为了保持A数据的空间连续性，操作系统就需要把B的数据拷贝到别的空间，比如拷贝到D刚刚释放的空间中。此时，应用C和D也分别删除了2字节和1字节的数据，整个内存空间上就分别出现了2字节和1字节的空闲碎片。如果应用E想要一个3字节的连续空间，显然是不能得到满足的。因为，虽然空间总量够，但却是碎片空间，并不是连续的。

好了，到这里，我们就知道了造成内存碎片的内外因素，其中，内存分配器策略是内因，而Redis的负载属于外因，包括了大小不一的键值对和键值对修改删除带来的内存空间变化。

大量内存碎片的存在，会造成Redis的内存实际利用率变低，接下来，我们就要来解决这个问题了。不过，在解决问题前，我们要先判断Redis运行过程中是否存在内存碎片。

### 如何判断是否有内存碎片？

Redis是内存数据库，内存利用率的高低直接关系到Redis运行效率的高低。为了让用户能监控到实时的内存使用情况，Redis自身提供了INFO命令，可以用来查询内存使用的详细信息，命令如下：

```
INFO memory
# Memory
used_memory:1073741736
used_memory_human:1024.00M
```

```
used_memory_rss_human:1.86G
…
mem_fragmentation_ratio:1.86
```

这里有一个mem_fragmentation_ratio的指标，它表示的就是Redis当前的内存碎片率。那么，这个碎片率是怎么计算的呢？其实，就是上面的命令中的两个指标used_memory_rss和used_memory相除的结果。

```
mem_fragmentation_ratio = used_memory_rss/ used_memory
```

used_memory_rss是操作系统实际分配给Redis的物理内存空间，里面就包含了碎片；而used_memory是Redis为了保存数据实际申请使用的空间。

我简单举个例子。例如，Redis申请使用了100字节（used_memory），操作系统实际分配了128字节（used_memory_rss），此时，mem_fragmentation_ratio就是1.28。

那么，知道了这个指标，我们该如何使用呢？在这儿，我提供一些经验阈值：

- **mem_fragmentation_ratio 大于1但小于1.5**。这种情况是合理的。这是因为，刚才我介绍的那些因素是难以避免的。毕竟，内因的内存分配器是一定要使用的，分配策略都是通用的，不会轻易修改；而外因由Redis负载决定，也无法限制。所以，存在内存碎片也是正常的。
- **mem_fragmentation_ratio 大于 1.5** 。这表明内存碎片率已经超过了50%。一般情况下，这个时候，我们就需要采取一些措施来降低内存碎片率了。



### 如何清理内存碎片？

当Redis发生内存碎片后，一个“简单粗暴”的方法就是**重启Redis实例**。当然，这并不是一个“优雅”的方法，毕竟，重启Redis会带来两个后果：

- 如果Redis中的数据没有持久化，那么，数据就会丢失；
- 即使Redis数据持久化了，我们还需要通过AOF或RDB进行恢复，恢复时长取决于AOF或RDB的大小，如果只有一个Redis实例，恢复阶段无法提供服务。

所以，还有什么其他好办法吗？

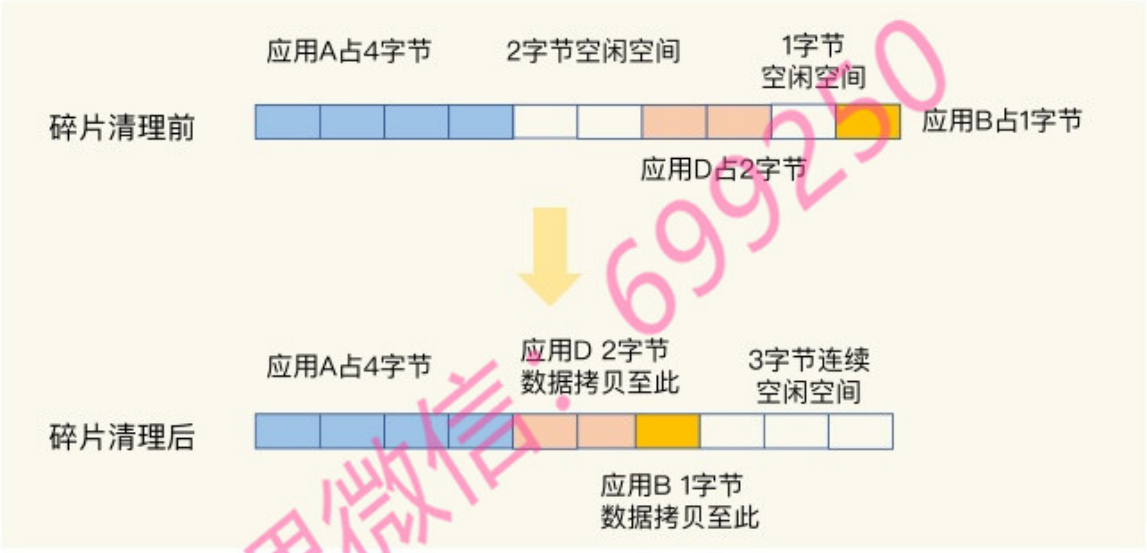
幸运的是，从4.0-RC3版本以后，Redis自身提供了一种内存碎片自动清理的方法，我们先来看这个方法的基本机制。

内存碎片清理，简单来说，就是“搬家让位，合并空间”。

我还以刚才的高铁车厢选座为例，来解释一下。你和小伙伴不想耽误时间，所以直接买了座位不在一起的三张票。但是，上车后，你和小伙伴通过和别人调换座位，又坐到了一起。

时，操作系统就会把数据拷贝到别处。此时，数据拷贝需要能把这些数据原来占用的空间都空出来，把原本不连续的内存空间变成连续的空间。否则，如果数据拷贝后，并没有形成连续的内存空间，这就不能算是清理了。

我画一张图来解释一下。






在进行碎片清理前，这段10字节的空间中分别有1个2字节和1个1字节的空闲空间，只是这两个空间并不连续。操作系统在清理碎片时，会先把应用D的数据拷贝到2字节的空闲空间中，并释放D原先所占的空间。然后，再把B的数据拷贝到D原来的空间中。这样一来，这段10字节空间的最后三个字节就是一块连续空间了。到这里，碎片清理结束。

不过，需要注意的是：**碎片清理是有代价的**，操作系统需要把多份数据拷贝到新位置，把原有空间释放出来，这会带来时间开销。因为Redis是单线程，在数据拷贝时，Redis只能等着，这就导致Redis无法及时处理请求，性能就会降低。而且，有的时候，数据拷贝还需要注意顺序，就像刚刚说的清理内存碎片的例子，操作系统需要先拷贝D，并释放D的空间后，才能拷贝B。这种对顺序性的要求，会进一步增加Redis的等待时间，导致性能降低。

那么，有什么办法可以尽量缓解这个问题吗？这就要提到，Redis专门为自动内存碎片清理功机制设置的参数了。我们可以通过设置参数，来控制碎片清理的开始和结束时机，以及占用的CPU比例，从而减少碎片清理对Redis本身请求处理的性能影响。

首先，Redis需要启用自动内存碎片清理，可以把activedefrag配置项设置为yes，命令如下：

```
config set activedefrag yes
```

这个命令只是启用了自动清理功能，但是，具体什么时候清理，会受到下面这两个参数的控制。这两个参数分别设置了触发内存清理的一个条件，如果同时满足这两个条件，就开始清理。在清理的过程中，只要有一个条件不满足了，就停止自动清理。

- **active-defrag-threshold-lower 10**：表示内存碎片空间占操作系统分配给Redis的总空间比例达到10%时，开始清理。

为了尽可能减少碎片清理对Redis正常请求处理的影响，自动内存碎片清理功能在执行时，还会监控清理操作占用的CPU时间，而且还设置了两个参数，分别用于控制清理操作占用的CPU时间比例的上、下限，既保证清理工作能正常进行，又避免了降低Redis性能。这两个参数具体如下：

- **active-defrag-cycle-min 25**：表示自动清理过程所用CPU时间的比例不低于25%，保证清理能正常开展；
- **active-defrag-cycle-max 75**：表示自动清理过程所用CPU时间的比例不高于75%，一旦超过，就停止清理，从而避免在清理时，大量的内存拷贝阻塞Redis，导致响应延迟升高。

自动内存碎片清理机制在控制碎片清理启停的时机上，既考虑了碎片的空间占比、对Redis内存使用效率的影响，还考虑了清理机制本身的CPU时间占比、对Redis性能的影响。而且，清理机制还提供了4个参数，让我们可以根据实际应用中的数据量需求和性能要求灵活使用，建议你在实践中好好地把这个机制用起来。



## 小结

这节课，我和你一起了解了Redis的内存空间效率问题，这里面的一个关键技术点就是要识别和处理内存碎片。简单来说，就是“三个一”：

- info memory命令是一个**好工具**，可以帮助你查看碎片率的情况；
- 碎片率阈值是一个**好经验**，可以帮忙你有效地判断是否要进行碎片清理了；
- 内存碎片自动清理是一个**好方法**，可以避免因为碎片导致Redis的内存实际利用率降低，提升成本收益率。

内存碎片并不可怕，我们要做的就是了解它，重视它，并借用高效的方法解决它。

最后，我再给你提供一个小贴士：内存碎片自动清理涉及内存拷贝，这对Redis而言，是个潜在的风险。如果你在实践过程中遇到Redis性能变慢，记得通过日志看下是否正在进行碎片清理。如果Redis的确正在清理碎片，那么，我建议你调小active-defrag-cycle-max的值，以减轻对正常请求处理的影响。

## 每课一问

按照惯例，我给你提一个小问题。在这节课中，我提到，可以使用mem_fragmentation_ratio来判断Redis当前的内存碎片率是否严重，我给出的经验阈值都是大于1的。那么，我想请你来聊一聊，如果mem_fragmentation_ratio小于1了，Redis的内存使用是什么情况呢？会对Redis的性能和内存空间利用率造成什么影响呢？

欢迎你在留言区写下你的思考和答案，和我一起交流讨论，如果觉得今天的内容对你有所帮助，也欢迎分享给你的朋友或同事，我们下节课见。

## 精选留言：

- Kaito 2020-09-23 00:08:55
  如果 mem_fragmentation_ratio 小于 1 了，Redis 的内存使用是什么情况呢？会对 Redis 的性能和内存空间利用率造成什么影响？

mem_fragmentation_ratio小于1，说明used_memory_rss小于了used_memory，这意味着操作系统分配给Redis进程的物理内存，要小于Redis实际存储数据的内存，也就是说Redis没有足够的物理内存可以使用了，这会导致Redis一部分内存数据会被换到Swap中，之后当Redis访问Swap中的数据时，延迟会变大，性能下降。

通过这篇文章了解到，Redis在进行内存碎片整理时，由于是主线程操作的，所以这块也是一个影响Redis性能的风险点。

其中active-defrag-ignore-bytes和active-defrag-threshold-lower参数主要用于控制达到什么阈值后开始碎片整理，如果配置的碎片大小和碎片率在可接受的范围内，那么Redis不会进行碎片整理，也就不会对Redis产生性能影响。

而达到设定阈值开始碎片整理后，active-defrag-cycle-min和active-defrag-cycle-max参数则用来控制在这期间，Redis主线程资源使用的上下限，这个需要根据碎片整理的时间、Redis的响应延迟进行权衡，合理配置。

我个人认为，应该优先保证Redis性能尽量不受影响，让碎片整理期间的资源消耗控制在稳定的范围内，并尽量缩短碎片整理的时间。
[34赞]



- test 2020-09-23 08:41:50
mem_fragmentation_ratio小于1，说明redis内存不够用了，换了一部分到swap中，会严重影响性能。[2赞]

- 可怜大灰狼 2020-09-23 11:43:18
老师您好，4.0-RC3 版本之前没有自动清理，是不是只能重启服务？ [1赞]

- 青青子衿 2020-09-23 09:02:06
这篇写的很好，感觉受益不小 [1赞]

- 小袁 2020-09-24 22:33:16
为什么不说下内存整理具体是怎么实现的？jemalloc应该不能控制，那到底怎么处理？

- 树斌 2020-09-24 18:42:15
实际案例，redis-cluster三主三从，检查所有节点的内存碎片率均小于1，在0.7-0.9之间，used_memory基本每个节点都只有12m左右，但是检查swap确认是没有虚拟内存交换的，不知道这种情况作何解释？一直没闹明白

- yeek 2020-09-23 08:59:52
小于的话，应该是发生了内存到磁盘的swap吧，这个影响就很大了，swap的数据访问效率会严重降低