1，增加内存；

**2，使用内存淘汰策略。**

**3，Redis集群。**

**重点介绍下23；**

**第2点：**

我们知道，redis设置配置文件的**maxmemory**参数，可以控制其最大可用内存大小（字节）。

那么当所需内存，超过**maxmemory**怎么办？

这个时候就该配置文件中的**maxmemory-policy**出场了。

其默认值是**noeviction**。

下面我将列出当可用内存不足时，删除redis键具有的淘汰规则。

|  |  |
| --- | --- |
| **规则名称** | **规则说明** |
| volatile-lru | 使用LRU算法删除一个键（只对设置了生存时间的键） |
| allkeys-lru | 使用LRU算法删除一个键 |
| volatile-random | 随机删除一个键（只对设置了生存时间的键） |
| allkeys-random | 随机删除一个键 |
| volatile-ttl | 删除生存时间最近的一个键 |
| noeviction | 不删除键，只返回错误 |

**LRU算法**，least Recently Used，**最近最少使用算法**。也就是说默认删除最近最少使用的键。

但是**一定要注意一点**！redis中并不会准确的删除所有键中最近最少使用的键，而是随机抽取**3个键**，删除这三个键中最近最少使用的键。

那么3这个数字也是可以设置的，对应位置是配置文件中的**maxmeory-samples**.

**第3点：**

**3.集群怎么做**

Redis仅支持单实例，内存一般最多10~20GB。对于内存动辄100~200GB的系统，就需要通过集群来支持了。

Redis集群有三种方式：**客户端分片、代理分片、Redis Cluster(在之后一篇文章详细说一下。)**

* 客户端分片

通过业务代码自己实现路由

优势：可以自己控制分片算法、性能比代理的好

劣势：维护成本高、扩容/缩容 等运维操作都需要自己研发

* 代理分片

代理程序接收到来自业务程序的数据请求，根据路由规则，将这些请求分发给正确的Redis实例并返回给业务程序。使用类似Twemproxy、Codis等中间件实现。

优势：运维方便、程序不用关心如何链接Redis实例

劣势：会带来性能消耗（大概20%）、无法平滑扩容/缩容，需要执行脚本迁移数据，不方便(Codis在Twemproxy基础上优化并实现了预分片来达到Auto Rebalance)。

* Redis Cluster

优势：官方集群解决方案、无中心节点，和客户端直连，性能较好

劣势：方案太重、无法平滑扩容/缩容，需要执行相应的脚本，不方便、太新，没有相应成熟的解决案例