# 生产问题

## Redis大key问题

在日常Redis的使用中，我们都遇到过因为key存储了过大的数据而造成请求缓慢甚至阻塞的情况

阿里云Redis 最佳实践中提到：

合理的 Key 中 Value 的字节大小，推荐小于10 KB。过大的 Value 会引发数据倾斜、热点Key、实例流量或 CPU 性能被占满等问题，应从设计源头上避免此类问题带来的性能影响。

### 发现

1. string类型可使用命令 ​​--bigkeys​​

2、非 string 类型通常有以下两种方式统计：

### 解决

不可以直接删除，因为redies内部仅一个线程，如果执行删除，则后续操作无法及时执行。

方法：

1. 低峰期删除
2. Scan分批删除法
3. 异步删除：unlink替代del来删除

**4.0 以前**

string，list，set，hash  不同数据类型的大 key，删除方式有所不同。

一般有两种情况：del 命令删除单个很大的 key  和  del 批量删除 大 key。

直接 del 命令粗暴的删大 key 容易造成 redis 线程阻塞。

4.0 以前要优雅的删除就是针对不同的类型 写脚本，拆分链表，hash 表，分批删除。

**4.0 以后：**

1. 主动删除大 key

127.0.0.1:6379> *UNLINK mykey*

unlink 命令是  **del 的异步版本，由 Lazyfree 机制实现**。

Lazyfree 机制的原理是在*删除的时候只进行逻辑删除，把 key 释放操作放在 bio (Background I/O)单独的子线程中惰性处理*，减少删除大 key 对 redis 主线程的阻塞，有效地避免因删除大key带来的性能问题。unlink 即使在批量删除大key时，也不会对阻塞造成阻塞。

2、被动删除大 key

被动删除是指 Redis 自身的 key 清除策略，一个 大 key 过期或者被淘汰时，如何被清除，会不会导致阻塞？4.0 以前自动清除是有可能阻塞主线程的。

4.0 以后的版本，被动删除策略是可选的配置参数，允许 以 Lazyfree 的方式清除。但是参数默认是关闭的，可配置如下参数开启。

*lazyfree-lazy-expire on    # 过期惰性删除*

*lazyfree-lazy-eviction on    # 超过最大内存惰性删除*

*lazyfree-lazy-server-del on    # 服务端被动惰性删除*