

# redis-糊涂点

---

1.单线程

2.跳表

3.redis key过期了为什么内存没释放

4.LRU 和 LFU

5.主从架构

6.redis主从架构执行命令出现死循环问题

## 1.单线程

redis指的是对命令执行的时候是单线程

## 2.跳表

跳表用于zset结构，用链表存取所有的数据并且有序，所以为提高链表查询效率，每两个结点在上面建立索引层，根据二分查找

## 3.redis key过期了为什么内存没释放

你在使用 Redis 时，肯定经常使用 SET 命令

SET 除了可以设置 key-value 之外，还可以设置 key 的过期时间，就像下面这样：

```
1 127.0.0.1:6379> SET tuling zhuge EX 120
2 OK
3 127.0.0.1:6379> TTL tuling
4 (integer) 117
```

此时如果你想修改 key 的值，但只是单纯地使用 SET 命令，而没有加上过期时间的参数，那这个 key 的过期时间将会被擦除

```
1 127.0.0.1:6379> SET tuling zhuge666
2 OK
3 127.0.0.1:6379> TTL tuling // key永远不过期了！
4 (integer) -1
```

- 定期删除，过期删除

## 4.LRU 和 LFU

## Redis淘汰Key的算法LRU与LFU区别

LRU 算法(Least Recently Used, 最近最少使用): 淘汰很久没被访问过的数据, LRU强调访问时间, 时间作为参考

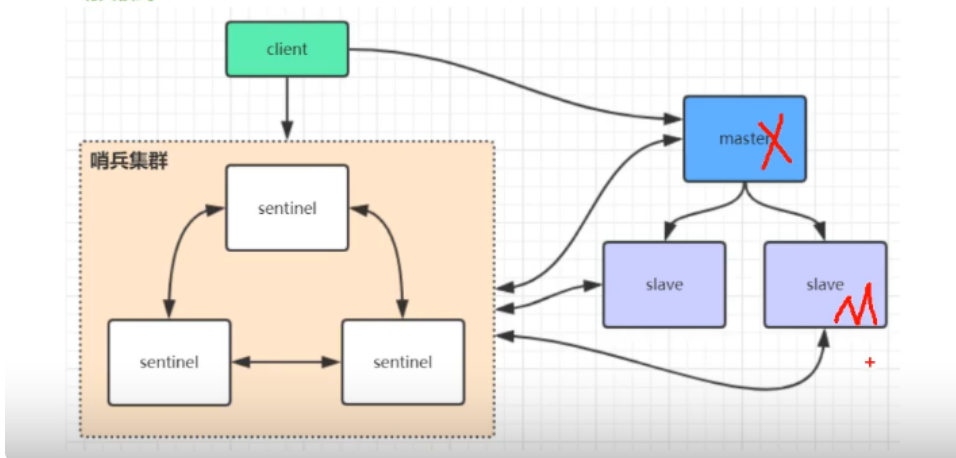
LFU 算法(Least Frequently Used, 最不经常使用): 淘汰最近一段时间被访问次数最少的数据, LFU强调访问次数, 次数作为参考

绝大多数情况我们都可以用LRU策略, 当存在大量的热点缓存数据时, LFU可能更好点

假如热点数据 1分钟访问了500次, 但后面5分钟没访问过, 用LRU这不给把热点数据干掉了

## 5.主从架构

### 哨兵模式



主从架构的流畅: 当客户端发送过来的请求操作redis,都会经过sentinel哨兵, 哨兵会将请求转发到主redis上, 如果主节点挂了, 哨兵会重新选举一台redis担任主节点

缺点: 所有的请求都在主节点上, 主节点容易出故障, 虽然主从切换可以保证高可用, 但是主从切换需要时间的, 可能会产生瞬断情况, 一台主节点对外开放, 高并发不是很高

## 6.redis主从架构执行命令出现死循环问题

如果在 slave 上执行 RANOMEKY, 那么问题会更严重。

slave 自己是不清理过期 key, 当一个 key 要过期时, master 会先清理删除它, 之后 master 向 slave 发送一个 DEL 命令, 告知 slave 也删除这个 key, 以此达到主从库的数据一致性。

假设Redis 中存在大量已过期还未被清理的 key, 那在 slave 上执行 RANDOMKEY 时, 就会发生以下问题:

- 1、slave 随机取出一个 key, 判断是否已过期。
- 2、key 已过期, 但 slave 不会删除它, 而是继续随机寻找不过期的 key。
- 3、由于大量 key 都已过期, 那 slave 就会寻找不到符合条件的 key, 此时就会陷入死循环。

也就是说, 在 slave 上执行 RANDOMKEY, 有可能会造成整个 Redis 实例卡死。

这其实是 Redis 的一个 Bug, 这个 Bug 一直持续到 5.0 才被修复, 修复的解决方案就是在slave中最多找一定的次数, 无论是否能找到, 都会退出循环。