该文档是极速PDF编辑器生成，
如果想去掉该提示,请访问并下载：
http://www.jisupdfeditor.com/

拓展 4：朝生暮死 —— 过期策略

Redis 所有的数据结构都可以设置过期时间，时间一到，就会自动删除。你可以想象 Redis 内部有一个死神，时刻盯着所有设置了过期时间的 key，寿命一到就会立即收割。

你还可以进一步站在死神的角度思考，会不会因为同一时间太多的 key 过期，以至于忙不过来。同时因为 Redis 是单线程的，收割的时间也会占用线程的处理时间，如果收割的太过于繁忙，会不会导致线上读写指令出现卡顿。

这些问题 Antirez 早就想到了，所有在过期这件事上，Redis 非常小心。

过期的 key 集合

redis 会将每个设置了过期时间的 key 放入到一个独立的字典中，以后会定时遍历这个字典来删除到期的 key。除了定时遍历之外，它还会使用惰性策略来删除

该文档是极速PDF编辑器生成，
如果想去掉该提示,请访问并下载：
http://www.jisupdfeditor.com/

过期的 key，所谓惰性策略就是在客户端访问这个 key 的时候，redis 对 key 的过期时间进行检查，如果过期了就立即删除。定时删除是集中处理，惰性删除是零散处理。

定时扫描策略

Redis 默认会每秒进行十次过期扫描，过期扫描不会遍历过期字典中所有的 key，而是采用了一种简单的贪心策略。

1. 从过期字典中随机 20 个 key；
2. 删除这 20 个 key 中已经过期的 key；
3. 如果过期的 key 比率超过 1/4，那就重复步骤 1；

同时，为了保证过期扫描不会出现循环过度，导致线程卡死现象，算法还增加了扫描时间的上限，默认不会超过 25ms。

设想一个大型的 Redis 实例中所有的 key 在同一时间过期了，会出现怎样的结果?

毫无疑问，Redis 会持续扫描过期字典 (循环多次)，直到过期字典中过期的 key 变得稀疏，才会停止 (循环次数明显下降)。这就会导致线上读写请求出现明显的卡顿现象。导致这种卡顿的另外一种原因是内存管理器需要频繁回收内存页，这也会产生一定的 CPU 消耗。

也许你会争辩说“扫描不是有 25ms 的时间上限了么，怎么会导致卡顿呢”？这里打个比方，假如有 101 个客户端同时将请求发过来了，然后前 100 个请求的执行时间都是 25ms，那么第 101 个指令需要等待多久才能执行？2500ms，这个就是客户端的卡顿时间，是由服务器不间断的小卡顿积少成多导致的。

所以业务开发人员一定要注意过期时间，如果有大批量的 key 过期，要给过期时间设置一个随机范围，而不能全部在同一时间过期。

py

```
# 在目标过期时间上增加一天的随机时间
redis.expire_at(key, random.randint(86400) + expire_ts)
```

在一些活动系统中，因为活动是一期一会，下一期活动举办时，前面几期的很多数据都可以丢弃了，所以需要给相关的活动数据设置一个过期时间，以减少不必

该文档是极速PDF编辑器生成，
如果想去掉该提示,请访问并下载：
http://www.jisupdfeditor.com/

要的 Redis 内存占用。如果不加注意，你可能会将过期时间设置为活动结束时间再增加一个常量的冗余时间，如果参与活动的人数太多，就会导致大量的 key 同时过期。

掌阅服务端在开发过程中就曾出现过多次因为大量 key 同时过期导致的卡顿报警现象，通过将过期时间随机化总是能很好地解决了这个问题，希望读者们今后能少犯这样的错误。

从库的过期策略

从库不会进行过期扫描，从库对过期的处理是被动的。主库在 key 到期时，会在 AOF 文件里增加一条 `del` 指令，同步到所有的从库，从库通过执行这条 `del` 指令来删除过期的 key。

因为指令同步是异步进行的，所以主库过期的 key 的 `del` 指令没有及时同步到从库的话，会出现主从数据的不一致，主库没有的数据在从库里还存在，比如上一节的集群环境分布式锁的算法漏洞就是因为这个同步延迟产生的。