链滴

# Redis 过期删除策略与内存淘汰策略

作者：Hildaquan

原文链接：https://ld246.com/article/1673062016223

来源网站：链滴

原文：Redis 过期删除策略和内存淘汰策略有什么区别？ | 小林coding (xiaolincoding.com)

# 大纲

过期删除策略和内存淘汰策略不要记混了！


# 过期删除策略

## 如何设置过期时间?

先说一下对 key 设置过期时间的命令。 设置 key 过期时间的命令一共有 4 个：

- expire ：设置 key 在 n 秒后过期，比如 expire key 100 表示设置 key 在 100 秒后过期；
- pexpire ：设置 key 在 n 毫秒后过期，比如 pexpire key2 100000 表示设置 key2 在 100000 毫秒（100 秒）后过期。
- expireat ：设置 key 在某个时间戳（精确到秒）之后过期，比如 expireat key3 1655654400 表示 key3 在时间戳 1655654400 后过期（精确到秒）；
- pexpireat ：设置 key 在某个时间戳（精确到毫秒）之后过期，比如 pexpireat key4 1655654 00000 表示 key4 在时间戳 1655654400000 后过期（精确到毫秒）

当然，在设置字符串时，也可以同时对 key 设置过期时间，共有 3 种命令：

- set ex：设置键值对的时候，同时指定过期时间（精确到秒）；
- set px：设置键值对的时候，同时指定过期时间（精确到毫秒）；
- setex：设置键值对的时候，同时指定过期时间（精确到秒）。

如果你想查看某个 key 剩余的存活时间，可以使用 TTL 命令。

```
# 设置键值对的时候，同时指定过期时间位 60 秒
> setex key1 60 value1
OK

# 查看 key1 过期时间还剩多少
> ttl key1
(integer) 56
> ttl key1
(integer) 52
```

如果突然反悔，取消 key 的过期时间，则可以使用 PERSIST 命令。

```
# 取消 key1 的过期时间
> persist key1
(integer) 1

# 使用完 persist 命令之后，
# 查下 key1 的存活时间结果是 -1，表明 key1 永不过期
> ttl key1
(integer) -1
```

# 如何判定 key 已过期了?

每当我们对一个 key 设置了过期时间时，Redis 会把该 key 带上过期时间存储到一个**过期字典**（expires dict）中，也就是说「过期字典」保存了数据库中所有 key 的过期时间。

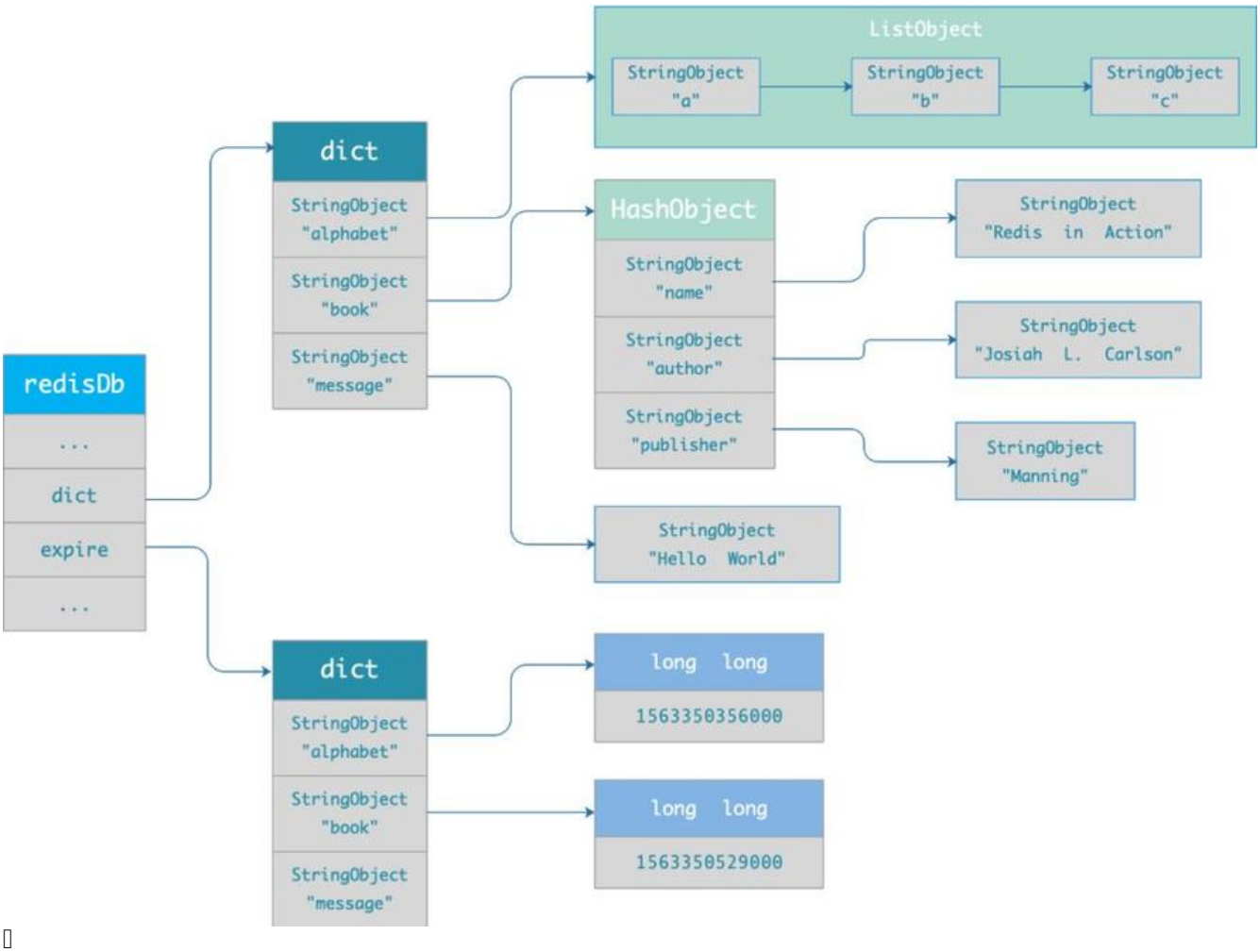
过期字典存储在 redisDb 结构中，如下：

```
typedef struct redisDb {
    dict *dict;    /* 数据库键空间，存放着所有的键值对 */
    dict *expires; /* 键的过期时间 */
    ....
} redisDb;
```

过期字典数据结构结构如下：

- 过期字典的 key 是一个指针，指向某个键对象；
- 过期字典的 value 是一个 long long 类型的整数，这个整数保存了 key 的过期时间；

过期字典的数据结构如下图所示：



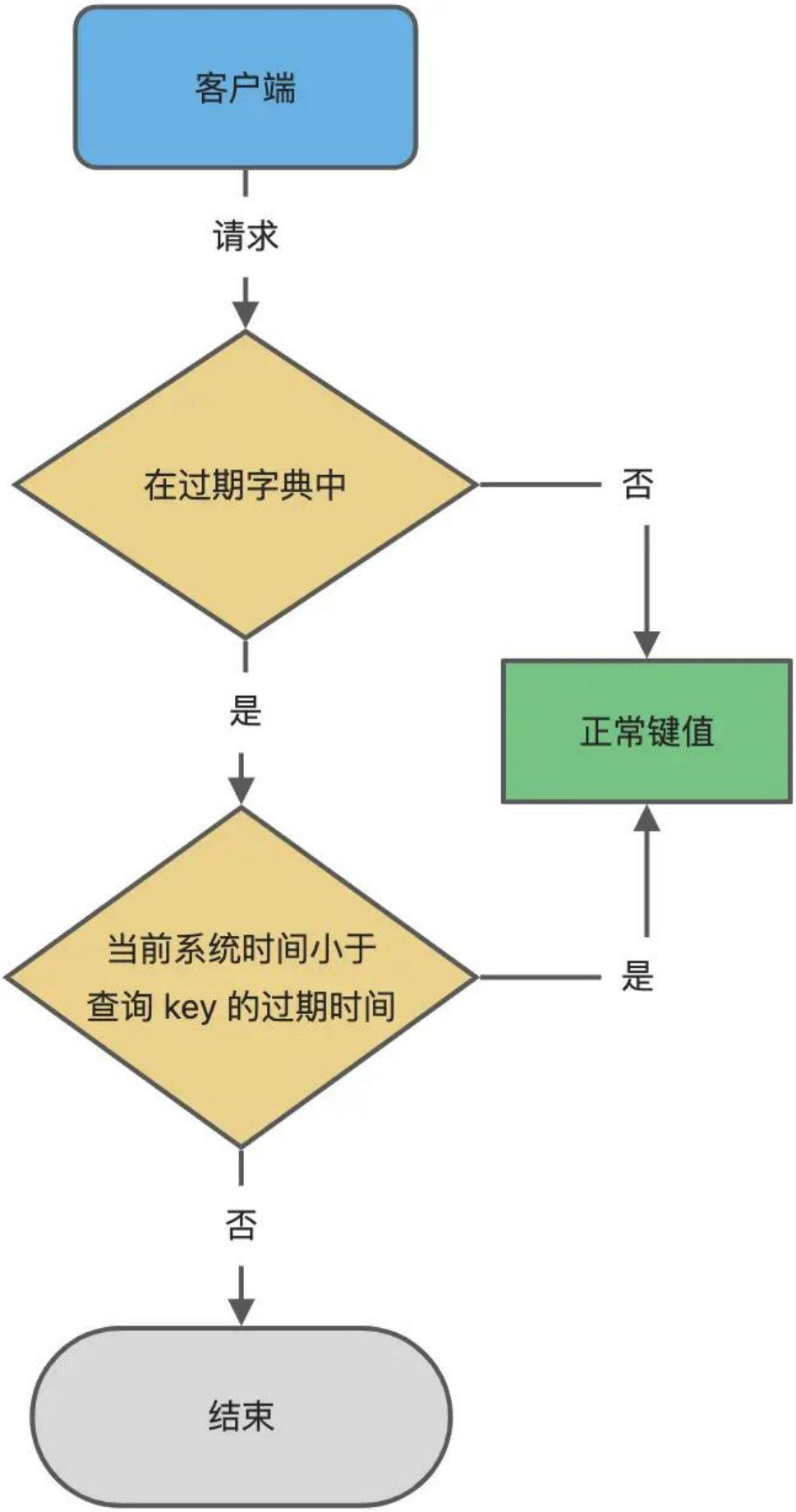

字典实际上是哈希表，哈希表的最大好处就是让我们可以用 O(1) 的时间复杂度来快速查找。当们查询一个 key 时，Redis 首先检查该 key 是否存在于过期字典中：

- 如果不在，则正常读取键值；
- 如果存在，则会获取该 key 的过期时间，然后与当前系统时间进行比对，如果比系统时间大，那就有过期，否则判定该 key 已过期。

看样子 Godis 的设计是没错的！！！

过期键判断流程如下图所示：

原文链接：Redis 过期删除策略与内存淘汰策略




原文链接：Redis 过期删除策略与内存淘汰策略

# 过期删除策略

在说 Redis 过期删除策略之前，先跟大家介绍下，常见的三种过期删除策略：

- 定时删除；
- 惰性删除；
- 定期删除；

接下来，分别分析它们的优缺点。

# 定时删除策略

定时删除策略的做法是，**在设置 key 的过期时间时，同时创建一个定时事件，当时间到达时，由件处理器自动执行 key 的删除操作。**

定时删除策略的**优点**：

- 可以保证过期 key 会被尽快删除，也就是内存可以被尽快地释放。因此，定时删除对内存是最友好。

定时删除策略的**缺点**：

- 在过期 key 比较多的情况下，删除过期 key 可能会占用相当一部分 CPU 时间，在内存不紧张但 CP 时间紧张的情况下，将 CPU 时间用于删除和当前任务无关的过期键上，无疑会对服务器的响应时间吞吐量造成影响。所以，定时删除策略对 CPU 不友好。

# 惰性删除策略

惰性删除策略的做法是，**不主动删除过期键，每次从数据库访问 key 时，都检测 key 是否过期如果过期则删除该 key。**

惰性删除策略的**优点**：

- 因为每次访问时，才会检查 key 是否过期，所以此策略只会使用很少的系统资源，因此，惰性删除略对 CPU 时间最友好。

惰性删除策略的**缺点**：

- 如果一个 key 已经过期，而这个 key 又仍然保留在数据库中，那么只要这个过期 key 一直没有被问，它所占用的内存就不会释放，造成了一定的内存空间浪费。所以，惰性删除策略对内存不友好。

# 定期删除策略

定期删除策略的做法是，**每隔一段时间「随机」从数据库中取出一定数量的 key 进行检查，并删其中的过期key。**

定期删除策略的**优点**：

● 通过限制删除操作执行的时长和频率，来减少删除操作对 CPU 的影响，同时也能删除一部分过期数据减少了过期键对空间的无效占用。

定期删除策略的**缺点**：

● 内存清理方面没有定时删除效果好，同时没有惰性删除使用的系统资源少。

● 难以确定删除操作执行的时长和频率。如果执行的太频繁，定期删除策略变得和定时删除策略一样对CPU不友好；如果执行的太少，那又和惰性删除一样了，过期 key 占用的内存不会及时得到释放

# Redis 实现定期删除

再回忆一下，定期删除策略的做法：**每隔一段时间「随机」从数据库中取出一定数量的 key 进行查，并删除其中的过期key。**

1、这个间隔检查的时间是多长呢？

在 Redis 中，默认每秒进行 10 次过期检查一次数据库，此配置可通过 Redis 的配置文件 redis.conf 进行配置，配置键为 hz 它的默认值是 hz 10。

特别强调下，每次检查数据库并不是遍历过期字典中的所有 key，而是从数据库中随机抽取一定量的 key 进行过期检查。

2、随机抽查的数量是多少呢？

我查了下源码，定期删除的实现在 expire.c 文件下的 activeExpireCycle() 函数中，其中随机抽的数量由 ACTIVE_EXPIRE_CYCLE_LOOKUPS_PER_LOOP() 定义的，它是写死在代码中的，数值是 2。

也就是说，数据库每轮抽查时，会随机选择 20 个 key 判断是否过期。

# Redis 的定期删除的流程：

1. 从过期字典中随机抽取 20 个 key；

2. 检查这 20 个 key 是否过期，并删除已过期的 key；

3. 如果本轮检查的已过期 key 的数量，超过 5 个（20/4），也就是「已过期 key 的数量」占比「随抽取 key 的数量」大于 25%，则继续重复步骤 1；如果已过期的 key 比例小于 25%，则停止继续删过期 key，然后等待下一轮再检查。

可以看到，定期删除是一个循环的流程。

那 Redis 为了保证定期删除不会出现循环过度，导致线程卡死现象，为此增加了定期删除循环流的时间上限，默认不会超过 25ms。

针对定期删除的流程，我写了个伪代码：

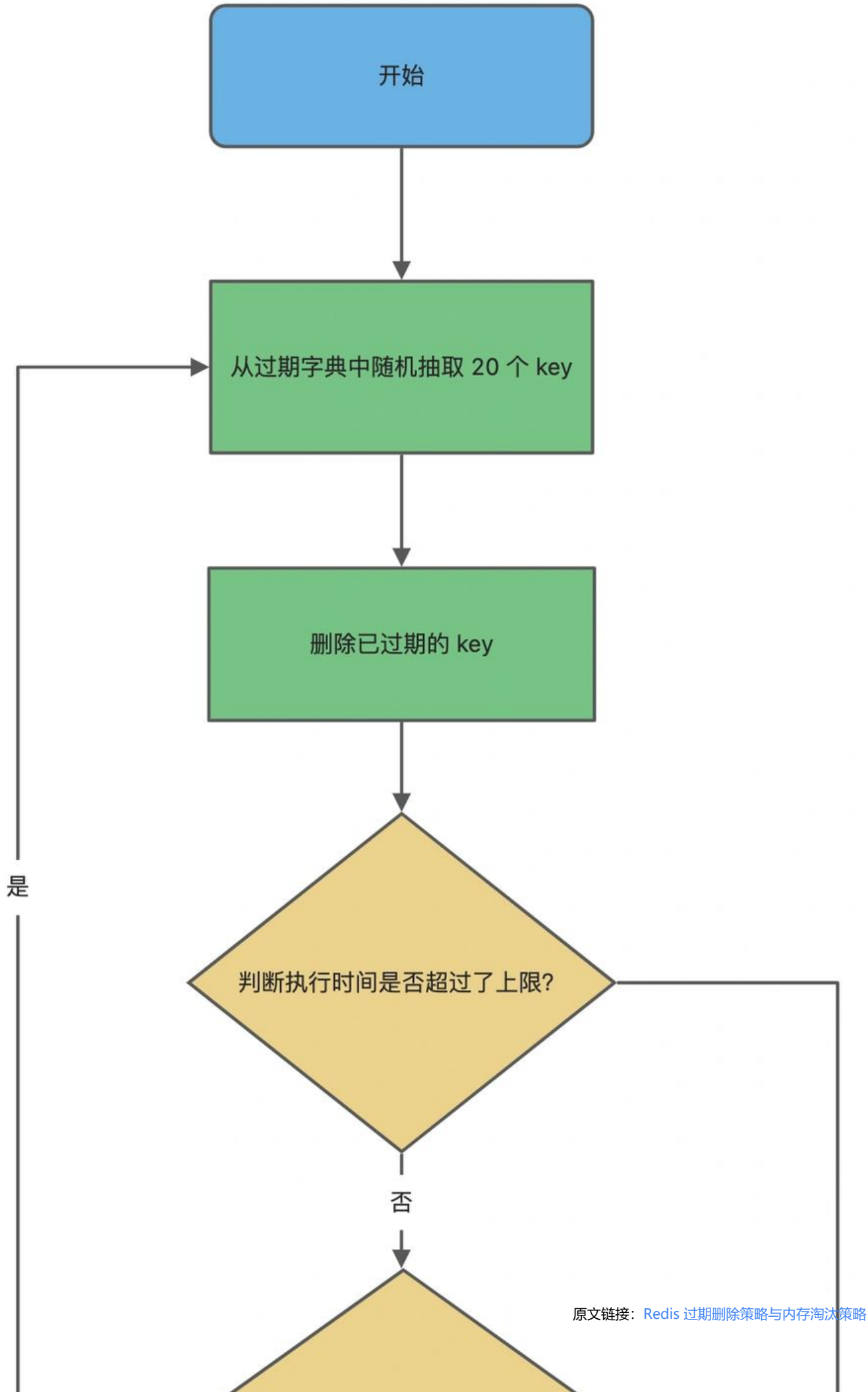
```
do {
    //已过期的数量
    expired = 0;
    //随机抽取的数量
    num = 20;
    while (num--) {
        //1. 从过期字典中随机抽取 1 个 key
        //2. 判断该 key 是否过期，如果已过期则进行删除，同时对 expired++
    }

    // 超过时间限制则退出
    if (timelimit_exit) return;

  /* 如果本轮检查的已过期 key 的数量，超过 25%，则继续随机抽查，否则退出本轮检查 */
} while (expired > 20/4);
```

定期删除的流程如下：

原文链接：Redis 过期删除策略与内存淘汰策略




原文链接：Redis 过期删除策略与内存淘汰策略