# 删除策略

**过期数据：曾今设置有效时间的数据到达了有效期，最终留下来的数据**

# 过期数据

## Redis中的数据特征

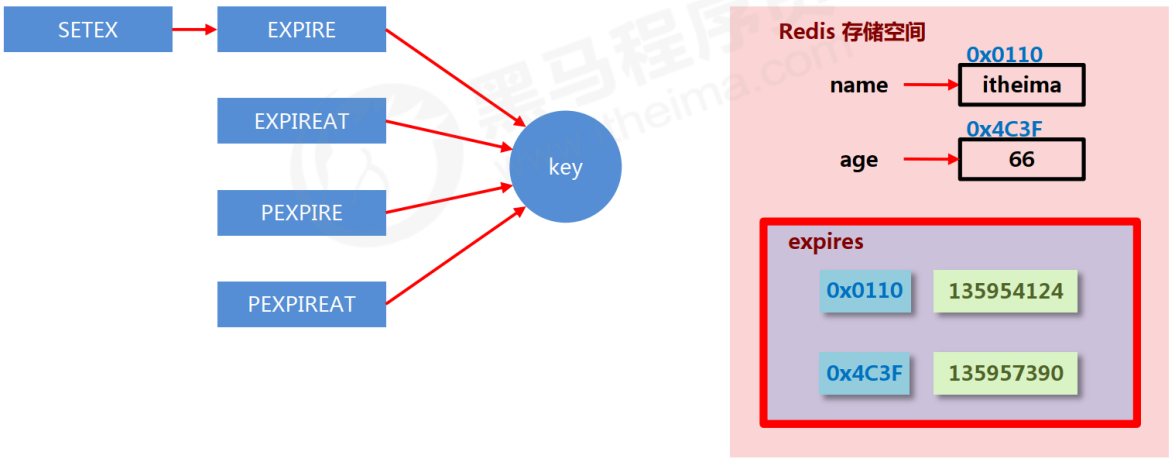
- Redis是一种内存级数据库，所有数据均放在内存中，内存中的数据可以通过TTL指令获取其状态、
  - 任意数字：具有时效性的数据
  - -1：永久有效的数据
  - -2：已过期的数据或被删除的数据或为定义的数据

# 数据删除策略

### 目标：在内存占用与CPU之间寻找一种平衡，顾此失彼都会造成整体redis性能下降，甚至引发服务器宕机或内存泄露

## 时效性数据存储结构

### 时效性数据的存储结构




**每一个数据库中存在一个expires空间，用来存储设定了过期时间的值得地址，expires空间才用的是hash结构存储**

1. **定时删除（处理器性能换取存储空间，空间换时间）**

- 创建一个定时器，当key设置有过期时间，且过期时间到达时，由定时任务立即执行对键的删除操作。会同时删除expires中的数据和对应内存中的数据
- 优点：节约内存，到时就删除，快速释放掉不必要的内存占用
- 缺点：CPU压力很大，无论CPU此时负载量多高，均占用CPU，会影响redis服务器响应时间和指令吞吐量

## 2. 惰性删除（存储空间换取处理器性能，时间换空间）

- 数据到达过期时间，不做处理，等下次访问改数据时，在每次访问数据之前，也就是执行get操作之前，都会执行一个函数expireIfNeeded()，用来判断数据是否过期。
  - 如果未过期，返回数据
  - 发现已过期，同时删除expires中的数据和内存中对应地址的数据，返回不存在
- 优点：节约CPU性能，发现必须删除的时候才删除
- 缺点：内存压力很大，出现长期占用内存的数据

## 3. 定期删除（前两种方式都走极端，这种是折中方案，用cpu给定的25%的时间？？？）

**理解：Redis每秒钟执行10次服务器轮训，轮训所有数据库中的expires，相当于每100ms轮训一次所有的expire是而对于每一个expires又花费25ms轮训其中的值**

- Redis启动服务其初始化时，读取配置server.hz的值，默认为10
- 每秒钟执行server.hz次serverCron()操作，对服务器进行定时轮训
- databasesCron()：对Redis中的每个库的expires进行访问**（轮训每一个库中的expires）**
- activeExpireCycle()：对每个expired[*]*逐一进行检测，每次执行250ms/server.hz**（轮训每一个expires）**
- 对某个expires[*]检测时，随机挑选W个key检测
  - 如果key超时，删除key
  - 如果一轮中删除的key的数量>W*25%，循环该过程
  - 如果一轮中删除的key的数量<=W*25%，检查下一个expires[*]*,0-15循环
  - W取值=ACTIVE_EXPIRE_CYCLE_LOOKUPS_PER_LOOP属性值
- 参数current_db用于记录activeExpireCycle() 进入哪个expires[*] 执行
- 如果activeExpireCycle()执行时间到期，下次从current_db继续向下执行
- 周期性轮询redis库中的时效性数据，采用随机抽取的策略，利用过期数据占比的方式控制删除频度

- 特点1： CPU性能占用设置有峰值，检测频度可自定义设置
- 特点2：内存压力不是很大，长期占用内存的冷数据会被持续清理
- 总结：周期性抽查存储空间（随机抽查，重点抽查）

## 删除策略对比

| 类别 | 内存 | CPU | 总结 |
| --- | --- | --- | --- |
| 定时删除 | 节约内存，无占用 | 不分时段占用CPU资源，频度高 | 时间换空间 |
| 惰性删除 | 内存占用严重 | 延时执行，CPU利用率高 | 空间换时间 |
| 定期删除 | 内存定期随机清理 | 每秒花费固定的CPU资源维护内存 | 随机抽查，重点抽查 |
| | | | |

# 逐出算法

### 当数据进入Redis时，如果内存不足怎么办

- Redis使用内存存储数据，在执行每一个命令前，会调用freeMemoryIfNeeded()检测内存是否充足。如 果内存不满足新加入数据的最低存储要求， redis要临时删除一些数据为当前指令清理存储空间。清理数据 的策略称为逐出算法。 、
- 注意：逐出数据的过程不是100%能够清理出足够的可使用的内存空间，如果不成功则反复执行。当对所 有数据尝试完毕后，如果不能达到内存清理的要求，将出现错误信息。

```
(error) OOM command not allowed when used memory >'maxmemory'
```

### 影响数据逐出的相关配置

- maxmemory：最大可用内存，占用物理内存的比例，默认值为0，表示不限制，生产环境中根据需求设定，通常设置在50%以上
- maxmemory-samples：每次选取待删除数据的个数，选取数据时如果全库扫描，会导致严重的性能消耗，降低读写性能，因此采用随机获取数据的方式作为待检测删除数据
- maxmemory-policy：删除策略，达到最大内存后，对被挑选出来的数据进行删除的策略

## 策略

- 检测易失数据（可能会过期的数据集server.db[i].expires）
  i. volatile-lru：最近最久未使用

ii. volatile-lfu：最近最少次数使用
iii. volatile-ttl：挑选将要过期的数据
iv. volatile-random：任意选择数据淘汰

- 检测全库数据（所有数据集server.db[i].dict）
  i. allkeys-lru：最近最久未使用
  ii. allkeys-lfu：最近最少使用
  iii. allkeys-random：任意选择数据淘汰
- 放弃数据驱逐
  - no-enviction（驱逐）：禁止驱逐数据（ redis4.0中默认策略），会引发错误OOM（ Out Of Memory）

配置方式：maxmemory-policy volatile-lru（常用）

# 数据逐出策略配置依据

使用INFO命令输出监控信息，查询缓存 hit 和 miss 的次数，根据业务需求调优Redis配置