缓存有效期和淘汰策略

2021年10月7日 1:14

有效期

缓存有限，无法存放所有数据。当我们将数据放到缓存中时，要设定一个有效期，超过有效期的缓存会被删除

淘汰策略

当缓存已满，但还有新的数据要放入，需要用什么样的方法将数据从缓存中淘汰

缓存清理方法

常用方法

1. 定时过期
   放入数据时，为每一个数据设定定时器，到时即清理
   这种做法不现实
   缓存中有大量数据，每个数据存入时间不同，有效期不同。如果为每个数据都添加定时器，会大量占用CPU资源

2. 惰性过期
   只有当访问到一个数据时，才判断这个数据有效期是否已经到了，如果已到即清理，没到即可使用
   这种做法对CPU友好，但对内存不友好，没被访问的数据会一直留存
3. 定期过期
   设定一个间隔，每个一段时间就扫描缓存中的数据，清除过期的数据
   可以设定扫描间隔和每次扫描多少数据

Redis策略：惰性过期+定期过期

默认情况下，每隔100ms就会扫描缓存，扫描到过期数据即清除
扫描范围不是全部扫描，而是采取随机抽取

缓存淘汰机制

常用方法

1. noeviction
   缓存不足，还要放入新数据，就会报错
   这种方法我们一般不用

注：我们不使用第一种方法，而是采用算法——最近最少使用LRU、最近不常使用LFU、随机
针对的数据有两个范围——volatile（在有效期内的数据）、allkeys（所有数据）

2. allkeys-lru
   把所有数据中所有最近最少使用的数据清理掉
3. allkeys-random
   从所有数据中随机拿一个数据清除掉

4. volatile-lru
   在有效期内的数据中把最近最少使用的数据清除
5. volatile-random
   略
6. volatile-ttl
   在有效期内数据中，取**有效期最早**的数据清除掉
7. allkeys-lfu
8. volatile-lfu

LRU和LFU差别

LRU：最近最少使用

LRU例子：

一开始我们将ABCDE按照最近的使用顺序排序

访问到D，将D移到队头

访问到B，将B移到队头

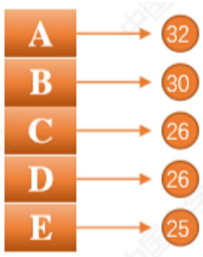
访问到一个新数据F，队列中没有，需要插入
将最近最少使用的E移除，F放到队头

LRU存在问题：

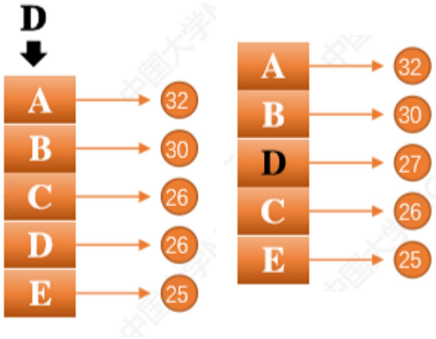
只能表现出数据访问次序，无法表现数据访问频率
如果有些访问频繁的数据近一段事件没被访问，而突然涌入大量数据，那么那些访问频繁的数据就会被挤出去

LFU：记录访问频率进行排序

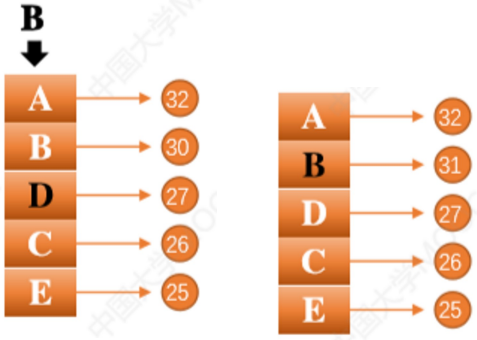
LFU例子：



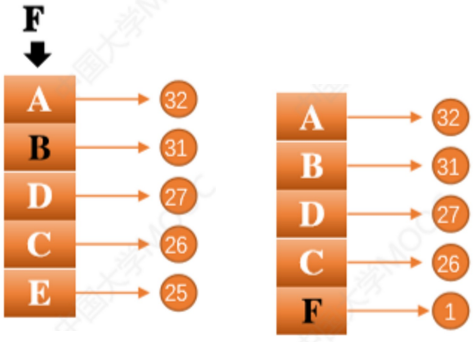
一开始我们有ABCDE五个数据，他们的访问频率分别为32、30、26、26、25



D被访问一次，D的频率增加，D排到C前



B被访问一次，B的频率增加，B排名不变



新数据F到来，将最近最不常访问的E淘汰，F入队，排在队尾

存在问题：

1. F进来之后排在队尾，如果之后再进来新数据，F又会很快被淘汰，因LFU对于新数据不友好
   所以LFU通常会为新数据设定一个5或6的初始值，不至于很快被淘汰
2. 老数据可能留存时间过长，如A一直没被访问，但始终在队头
   所以LFU会做一个定期的衰减，过一段时间会把所有数据访问次数减去一定值，使老数据也会慢慢被淘汰

Redis淘汰策略：近似LRU/LFU算法

在需要淘汰时，从内存中随机选取N个数据，在这个N个数据中执行LRU/LFU算法