# 杨雪婷工作周报（2021.04.19 - 2021.04.23）

1. 想调查两种经典算法（LRU / LFU）的现有改进算法。
2. LFU : LFU\* , LFU - Aging , LFU\* - Aging , Window - LFU

改进大概分为：除了原本的访问次数还引入访问时间的考虑，以及限制记 录时间来减少空间复杂度。

1. LRU : LRU - K , 2Q , LIRS

改进思路：不要让hot cache被只访问一次的cache轻易替换掉，实现的时候巧妙结合数据结构的应用（优先队列，栈等）。

1. 次周总结：

（今明两天有期中考试，这周没有用很多时间来看非常惭愧QAQ.

将LFU的改动大概了解了（改动不大而且不是很复杂，好像改动之后性能也一般般没有在实际Redis等数据库结构中应用到）

LRU的改进算法本身不太复杂，这周只了解了LRU - K和2Q，LIRS还没有看。对于LRU - K和2Q算法网上仅有的资料还有些区别（/捂脸）。

区别大概在于，这两种改进方案还有两点问题没考虑：

1. 关联访问（首次访问后可能连带着一连串的访问，可能造成错误判断，引入Correlated References Period（块首次访问后的一段时间），首次访问后过了这段时间后才算做另一次访问。
2. 无记忆性的问题：cache被替换后所有访问信息丢失，可能将hot块抛弃。因此引入Reference Retained Information Period（对于替换出cache后的块会继续保留访问信息一段时间）。

考虑到这两个问题后，整个算法的实现数据结构（队列等）都变了。

1. 下周计划：

- 了解LIRS的算法思路

- 看上述两个改进后存在的新问题是怎么和实现的数据结构结合起来的。

19 数媒技 杨雪婷

2021.04.23

# 经典算法的几种现有改进

### 零、几种request类型：

1.顺序访问。所有的块一个接一个被访问，不存在重访问。

2.循环访问。所有块都按照一定的间隔重复访问

3.时间密集访问。最近被访问的块是将来最有可能被访问的。

4.概率访问。所有块都有固定的访问概率，所有块都互相独立地根据概率被访问。

5.关联访问(Correlated References)，块被首次访问之后，紧接着的短时间内会有数次访问。

## 一、LFU算法的改进

算法根据数据的历史访问频率来淘汰数据，其核心思想是“如果数据过去被访问多次，那么将来被访问的频率也更高”。

### 1、LFU\*算法

#### 思路：

基于LFU的改进算法，其核心思想是“只淘汰**访问过一次**的数据”。

且如果所有引用计数为1的数据大小之和都没有新加入的数据那么大，则不淘汰数据，新的数据也不缓存。

### 2、LFU - Aging算法

#### 思路：

其核心思想是“除了访问次数外，还要考虑访问时间”。

通过**平均引用计数**来标识时间。当当前缓存中的数据“引用计数平均值”达到或者超过“最大平均引用计数”时，则将所有数据的引用计数都减少。减少的方法有多种，可以直接减为原来的一半，也可以减去固定的值等。

### 3、LFU\*-Aging算法

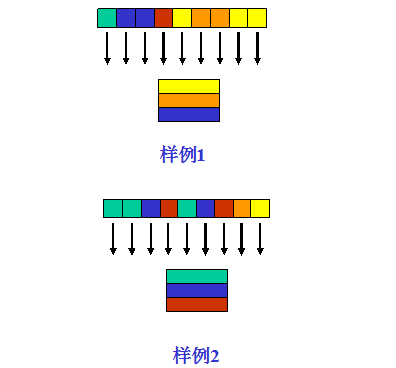
#### 思路：

结合LFU\*和LFU - Aging

### 4、Window - LFU算法

#### 思路：

Window - LFU并不记录所有数据的访问历史，而只是记录过去一段时间内的访问历史



样例1：黄色访问3次，蓝色和橘色都是两次，橘色更新，因此缓存黄色、橘色、蓝色三个数据块

样例2：绿色访问3次，蓝色两次，暗红两次，蓝色更新，因此缓存绿色、蓝色、暗红三个数据块

|  |  |
| --- | --- |
| 对比点 | 对比 |
| 命中率 | Window - LFU/LFU - Aging > LFU *- Aging > LFU > LFU* |
| 复杂度 | LFU - Aging > LFU> LFU *- Aging >Window - LFU > LFU* |
| 代价 | LFU - Aging > LFU > Window - LFU > LFU *- Aging > LFU* |

## 二、LRU算法的改进

淘汰最长时间未被使用的页面。

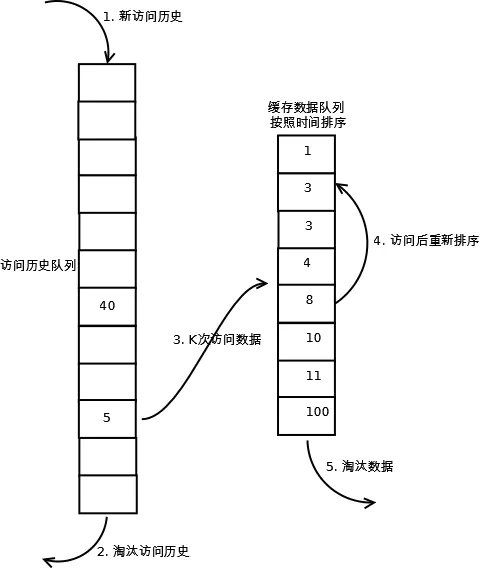
存在问题：

* 对冷数据突发性访问抵抗能力差，可能会因此淘汰掉热的文件。热点页面在偶然一个时间节点被其他大量仅访问了一次的页面所取代则造成浪费。
* 对于大量数据的循环访问抵抗能力查，极端情况下可能会出现命中率0%。（如：循环访问）
* 不能按照数据的访问概率进行淘汰。

### 1、LRU - K算法

#### [思路](https://zhuanlan.zhihu.com/p/348838247)：

相比于传统的LRU就是LRU - 1，仅访问了一次就能代替别人，**K次访问**才能有替换资格。最后第K次的访问距离。访问距离 ↑，时间间隔 ↑ ，被替换 ↑ 。



LRU-2，只有当数据的访问次数达到2次的时候，才将数据放入缓存。当需要淘汰数据时，LRU-2会淘汰第2次访问时间距当前时间最大的数据。

#### 实现步骤：

(1) 数据第一次被访问，加入到访问历史列表；

(2) 如果数据在访问历史列表里后没有达到K次访问，则按照一定规则（FIFO，LRU）淘汰；

(3) 当访问历史队列中的数据访问次数达到K次后，将数据索引从历史队列删除，将数据移到缓存队列中，并缓存此数据，缓存队列重新按照时间排序；

(4) 缓存数据队列中被再次访问后，重新排序；

(5) 需要淘汰数据时，淘汰缓存队列中排在末尾的数据，即：淘汰“倒数第K次访问离现在最久”的数据。

#### [仍然存在的问题和改进:](https://blog.csdn.net/Pun_C/article/details/50920469)

1、针对**关联访问**的问题：块被首次访问之后，紧接着的短时间内会有数次访问。（如：数据库中同一事务内的select和update会多次扫描相同的块）

提出参数：Correlated References Period（块首次访问后的一段时间）为了避免关联访问的干扰造成对块的错误判断，在这段时间内的多次访问只算作一次访问。只有这段时间后块再次被访问，才算第二次被访问。

2、针对**无记忆性**的问题：块被替换出cache后，可能很快地再次被访问，由于之前访问记录已丢弃，这样只算作首次访问，之后又很快被替换出cache后，又再次被访问，这样又只会算作首次访问，如此下来，虽然块被频繁访问，属于hot块，但由于替换出cache后没有保留访问信息，导致错误判断

提出参数：Reference Retained Information Period（对于替换出cache后的块会继续保留访问信息一段时间）

#### LRU - K其他的问题：

1. 由于优先级队列的排序操作需要额外的O(logN)的时间复杂度，N为P的大小。
2. A1，P和A2的大小都必须按照实际情况进行配置取最优比例，才能发挥最优性能。
3. 块的访问频率变化响应较慢。这是因为P的比较是按照历史的最后第K次访问距离进行比较。如果块A在P中的时候倒数第K次的距离较少，但经过较长时间才有新的访问，重新更新访问距离后，才会被快速替换出cache。

### 2、2Q算法

#### 思路：

该算法类似于LRU-2，不同点在于2Q将LRU-2算法中的访问历史队列（不是缓存数据的）改为一个FIFO缓存队列，即：2Q算法有**两个缓存队列**，一个是FIFO队列A1，一个是LRU队列Am。

#### 问题：

A1和Am各自所占cache的比例是关键。

如果A1太小，则检测是否hot块的时间太短，很可能需要较长时间才把hot块加入到Am中。但如果A1太大，则A1会占了原本所属Am的cache，hot块的数量就会减少，会影响cache命中率。

【待补充改进和缺点】

### 3、LIRS (Low Inter-reference Regency Set)

【待看】