基于最少使用频次的LFU缓存淘汰算法

概念分析

LFU(Least Frequently Used)即最近最不常用.看名字就知道是个基于访问频次的一种算法。以前写过几篇关于用python实现lru算法的模块,有兴趣的朋友可以看看。 LRU是基于时间的,会将时间上最不常访问的数据给淘汰,在算法表现上是放到列表的顶部;LFU为将频率上最不常访问的数据淘汰.既然是基于频率的,就需要有存储每个数据访问的次数.从存储空间上,较LRU会多出一些持有计数的空间.

LFU算法认为"如果数据过去访问频率很高,那么将来被访问的频率也很高".其实我个人对于缓存用的最多的是 Lru和Fifo,LFU更加偏向于随机性比较大的场景。

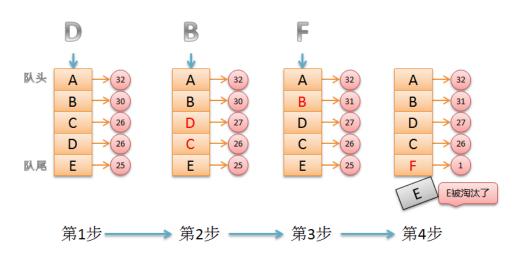
我擦,最近爬虫很是凶猛呀标记下文章的原文http://xiaorui.cc (http://xiaorui.cc)

被爬虫啦! 更多python开发及运维研发文章

请移步到 http://xiaorui.cc

下面的图我是从别人那边偷来的....别打我

LFU淘汰过程



下面简单讲解一下:

- 1. 假设我们的lfu最大的存储空间控制为5个,此时访问D,D现在的访问频率计数是26;
- 2. 访问D后,D的频率+1,也就是27了。 此时需要调整缓存池数据需要重新排序,D和C交换;
- 3. 访问B,B的频率+1,由于A的频率仍然比B大,所以不需要调整;
- 4. 当新数据F插入缓存池之前,由于已经空间满了,需要干掉一个!因为E的频率最低,故淘汰E,将F插入缓存池,缓存池重新排序,F放到队尾,

3.优略分析

【命中率】

命中率方面还是要看你的应用的场景 大多数的场景下,一旦访问内容发生较大变化,LFU需要用更长的时间来适应,因为他是往队列的下面塞入,如果一定时间内对于新数据的访问频 次量不够的话,后面再继续的不断的推入新的数据,根据Lfu的算法,被干掉的肯定是那些刚刚被推进来的,还没有被多次访问的那堆数据了。(历史的频率记录会是这些污染数据保持较 长的一段时间) ,这也是为啥大多数人用Lru的原因。

【复杂度】

需要维护所有的访问记录的频率数据结构,实现较LRU复杂.

【存储成本】

需要维护所有的访问记录的频率数据结构.

【缺陷】

仅仅从最近访问频率上考虑淘汰算法.可能会淘汰一些仍有价值的单元.内存和性能消耗较高.

下面的是用python实现Lfu缓存算法的装饰器。 脚本里面增加了随机的访问的概念统计,Lfu非常适合这种随机性比较大,跟时间没太多关系的缓存。代码我已经推送到了pypi里面,有

兴趣的朋友可以直接安装

```
Python

1 
2 pylfuńsył:
3 pip install pylfu
```

```
Python
1
2 import collections
3 import functools
4 from itertools import ifilterfalse
5 from heapq import nsmallest
6 from operator import itemgetter
7 class Counter(dict):
        'Mapping where default values are zero'
9
       def __missing__(self, key):
10
           return 0
11 def lfu cache(maxsize=100):
      '''Least-frequenty-used cache decorator.
12
13
       Arguments to the cached function must be hashable.
14
15
       Cache performance statistics stored in f.hits and f.misses.
       Clear the cache with f.clear().
16
17
       http://en.wikipedia.org/wiki/Least_Frequently_Used
18
19
20
       def decorating_function(user_function):
21
           cache = {}
                                           # mapping of args to results
22
           use_count = Counter()
                                           # times each key has been accessed
23
           kwd_mark = object()
                                           # separate positional and keyword args
24
25
           @functools.wraps(user_function)
26
           def wrapper(*args, **kwds):
27
               key = args
28
               if kwds:
29
                   key += (kwd_mark,) + tuple(sorted(kwds.items()))
30
               use_count[key] += 1
31
32
               # get cache entry or compute if not found
33
34
                   result = cache[key]
35
                   wrapper.hits += 1
36
               except KeyError:
37
                   result = user_function(*args, **kwds)
                   cache[key] = result
38
39
                   wrapper.misses += 1
40
41
                   # purge least frequently used cache entry
42
                   if len(cache) > maxsize:
43
                       for key, _ in nsmallest(maxsize // 10,
44
                                               use_count.iteritems(),
45
                                               key=itemgetter(1)):
46
                           del cache[key], use_count[key]
47
48
               return result
49
50
           def clear():
51
               cache.clear()
52
               use_count.clear()
53
               wrapper.hits = wrapper.misses = 0
54
55
           wrapper.hits = wrapper.misses = 0
56
           wrapper.clear = clear
57
           return wrapper
58
       return decorating_function
59
60
```

```
61 if __name__ == '__main__':
63
       @lfu_cache(maxsize=20)
64
       def f(x, y):
65
           return 3*x+y
66
67
      domain = range(5)
       from random import choice
68
69
      for i in range(1000):
70
           r = f(choice(domain), choice(domain))
71
72
      print(f.hits, f.misses)
```

如果大家觉得文章对你有些作用! 帮忙点击广告. 一来能刺激我写博客的欲望, 二来好维护云主机的费用. 如果想赏钱,可以用微信扫描下面的二维码. 另外再次标注博客原地址 xiaorui.cc (http://xiaorui.cc) 感谢!



 $\bot - \$ (\text{http://xiaorui.cc/2015/04/19/python-ordereddict\%e5\%ae\%9e\%e7\%8e\%b0\%e6\%9c\%89expire\%e5\%92\%8cmax\%e7\%9a\%84\%e9\%98\%9f\%e5\%88\%97\%e5\%92\%8cmax\%e7\%pamax\%e7\%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax\%e7%pamax%e7%$

别倒贴,找人DDOS你!

原文地址:

http://xiaorui.cc/2015/04/20/%e5%9f%ba%e4%ba%8e%e9%a2%91%e6%ac%a1%e7%9a%84%e7%bc%93%e5%ad%98%e6%b7%98%e6%b1%b0%e7%ae%97%e6%b3%95%e4%b9%8blfu/(http://xiaorui.cc/2015/04/20/%e5%9f%ba%e4%ba%8e%e9%a2%91%e6%ac%a1%e7%9a%84%e7%bc%93%e5%ad%98%e6%b7%98%e6%b1%b0%e7%ae%97%e6%b3%95%e4%b9%8blfu/) 转载时必须以链接形式注明原始出处及本声明。

公告

有事可以发邮件, rfyiamcool@163.com

我的另一个博客,偏向运维方面的~ops.xiaorui.cc (http://ops.xiaorui.cc)

我个人的github地址是, github.com/rfyiamcool (http://github.com/rfyiamcool)

我给pypi提交过的项目,[话说曾经被删过一些] pypi地址 (https://pypi.python.org/pypi?%3Aaction=search&term=fengyun)

博客使用阿里云主机服务,大家可以使用我的优惠推荐码购买主机... 九折优惠哦 pmcp6m

对Python及运维开发感兴趣的朋友可以加QQ群 : 478476595







最近访客

标签

ansible (http://xiaorui.cc/tag/ansible/) ansible api (http://xiaorui.cc/tag/ansible-api/) docker (http://xiaorui.cc/tag/docker/) docker api (http://xiaorui.cc/tag/docker-api/) elasticsearch (http://xiaorui.cc/tag/elasticsearch/) elk (http://xiaorui.cc/tag/elk/) epoll (http://xiaorui.cc/tag/epoll/) etcd (http://xiaorui.cc/tag/etcd/) gevent (http://xiaorui.cc/tag/gevent/) golang (http://xiaorui.cc/tag/golang-2/) grafana (http://xiaorui.cc/tag/grafana/) happybase (http://xiaorui.cc/tag/happybase/)

Influxdb (http://xiaorui.cc/tag/influxdb/) InfluxDB 性能 (http://xiaorui.cc/tag/influxdb-%e6%80%a7%e8%83%bd/) kibana (http://xiaorui.cc/tag/kibana/) kibana4 (http://xiaorui.cc/tag/kibana4) logstash (http://xiaorui.cc/tag/logstash/) mysql (http://xiaorui.cc/tag/mysql/) peewee (http://xiaorui.cc/tag/peewee/) peewee 源码 (http://xiaorui.cc/tag/peewee-%e6%ba%90%e7%a0%81/) python (http://xiaorui.cc/tag/python/) python docker (http://xiaorui.cc/tag/python-docker/) python elasticsearch (http://xiaorui.cc/tag/python-elasticsearch/) python epoll (http://xiaorui.cc/tag/python-epoll/) python etcd (http://xiaorui.cc/tag/python-etcd/) python gevent (http://xiaorui.cc/tag/python-gevent/) python happybase (http://xiaorui.cc/tag/python-happybase/) python hbase (http://xiaorui.cc/tag/python-hbase/) python influxdb (http://xiaorui.cc/tag/python-influxdb/) python mongodb (http://xiaorui.cc/tag/python-mongodb/) python multiprocessing (http://xiaorui.cc/tag/python-multiprocessing/) python peewee (http://xiaorui.cc/tag/python-peewee/) python redis (http://xiaorui.cc/tag/python-redis/) python requests (http://xiaorui.cc/tag/python-requests/) python rq (http://xiaorui.cc/tag/python-rq/) python thrift (http://xiaorui.cc/tag/python-thrift/) python zabbix (http://xiaorui.cc/tag/python-zabbix/) redis (http://xiaorui.cc/tag/redis/) redis cluster (http://xiaorui.cc/tag/redis-cluster/) saltstack (http://xiaorui.cc/tag/saltstack/) saltstack api (http://xiaorui.cc/tag/saltstack-api/) tokudb (http://xiaorui.cc/tag/tokudb/) zabbix (http://xiaorui.cc/tag/zabbix/) zabbix二次开发 (http://xiaorui.cc/tag/zabbix%e4%ba%8c%e6%ac%a1%e5%bc%80%e5%8f%91/) zabbix开发 (http://xiaorui.cc/tag/zabbix%e5%bc%80%e5%8f%91/)

♪ 最热文章

解决mac osx下pip安装ipython权限的问题 (http://xiaorui.cc/2016/03/27/%e8%a7%a3%e5%86%b3mac-osx%e4%b8%8bpip%e5%ae%89%e8%a3%85ipython%e6%9d%83%e9%99%90%e7%9a%84%e9%97%ae%e9%a2%98/)	9
公司内部的python分享《美妙的多进程管理》 (http://xiaorui.cc/2016/04/06/%e5%85%ac%e5%8f%b8%e5%86%85%e9%83%a8%e7%9a%84python%e5%88%86%e4%ba%ab%e3%80%8a%e7%be%8e%e5%a6%99%e7%9a%84%e5%a4%9a%e8%bf%9b%e7%a8%8b%e7%ae%a1%e7%90%86%	e3%80%8b/)
沈灿的新书《Ansible自动化运维·技术与最佳实践(http://xiaorui.cc/2016/05/19/%e6%b2%88%e7%81%bf%e7%9a%84%e6%96%b0%e4%b9%a6- %e3%80%8aansible%e8%87%aa%e5%8a%a8%e5%8c%96%e8%bf%90%e7%bb%b4%e6%8a%80%e6%9c%af%e4%b8%8e%e6%9c%80%e4%bd%b3%e5%ae%9e%e8%b7%b5%e3%80%8b/)	6
心路历程之我的阅读书单 (http://xiaorui.cc/2016/05/17/%e5%bf%83%e8%b7%af%e5%8e%86%e7%a8%8b%e4%b9%8b%e6%88%91%e7%9a%84%e9%98%85%e8%af%bb%e4%b9%a6%e5%8d%95/)	5
技术分享《分布式一致性raft算法实现原理》(http://xiaorui.cc/2016/07/08/%e6%8a%80%e6%9c%af%e5%88%86%e4%ba%ab- %e3%80%8a%e5%88%86%e5%b8%83%e5%bc%8f%e4%b8%80%e8%87%b4%e6%80%a7%e7%ae%97%e6%b3%95%e5%ae%9e%e7%8e%b0%e5%8e%9f%e7%90%86%e3%80%8b/)	5
详解python调用redis lua内嵌脚本的高级用法 (http://xiaorui.cc/2016/03/27/%e8%af%a6%e8%a7%a3python%e8%b0%83%e7%94%a8redis-lua%e5%86%85%e5%b5%8c%e8%84%9a%e6%9c%ac%e7%9a%84%e9%ab%98%e7%ba%a7%e7%94%a8%e6%b3%95/)	5
探索mysql和redis主从复制的实现原理 (http://xiaorui.cc/2016/05/15/%e6%8e%a2%e7%b4%a2mysql%e5%92%8credis%e4%b8%bb%e4%bb%8e%e5%a4%8d%e5%88%b6%e7%9a%84%e5%ae%9e%e7%8e%b0%e5%8e%9f%e7%90%86/)	4
打造mvc框架之python template模板实现原理 (http://xiaorui.cc/2016/04/12/%e6%89%93%e9%80%a0mvc%e6%a1%86%e6%9e%b6%e4%b9%8bpython-template%e6%a8%a1%e6%9d%bf%e5%ae%9e%e7%8e%b0%e5%8e%9f%e7%90%86/)	4

Copyright © 2013. 峰云就她了,专注与运维自动化平台. EMEZ === (http://www.cnzz.com/stat/website.php?web_id=1253526843)