(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 113050894 A
(43)申请公布日 2021.06.29

(21)申请号 202110424188.2

(22)申请日 2021.04.20

(71)申请人 南京理工大学
地址 210094 江苏省南京市孝陵卫200号
申请人 江苏省农业科学院

(72)发明人 王吟吟 杨余旺 柯亚琪 陈霆希 曹宏鑫 葛道阔

(74)专利代理机构 北京盛凡智荣知识产权代理有限公司 11616
代理人 范国刚

(51)Int.Cl.
G06F 3/06(2006.01)
G06N 3/00(2006.01)

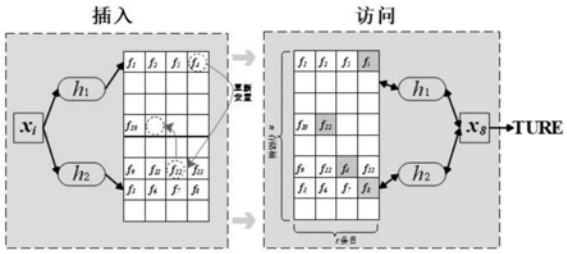
权利要求书1页 说明书5页 附图2页

(54)发明名称

一种基于布谷鸟算法的农业光谱混合存储系统缓存替换算法

(57)摘要

本发明公开了一种基于布谷鸟算法的农业光谱数据缓存替换算法。近年来海量增长得农业光谱数据数据的对存储系统带来巨大挑战。单一的存储介质如HDD(Hard Disk Drive,HDD)或SSD(Solid State Drive,SSD)由于其固有物理特性限制,并不能满足实际需求,将SSD和HDD混合使用的存储架构是一种可行的解决方案,这样管理农业光谱数据的混合存储的缓存替换策略就成为存储性能提升的关键。我们推出一种基于计数布谷鸟过滤器热探测方案(Count Cuckoo Filter Hot-probe Method,CCF)具有非常好的空间和时间效率,且支持删除功能。通过将CCF和自适应两级LRU(Least Recently Used,LRU)相结合,形成CCF-LRU缓存替换策略,该策略利用CCF识别热数据,利用自适应两级LRU管理缓存。实验结果表明,结合热探针方案的缓存替换策略与传统策略相比,能显著提高缓存命中率。CCF-LRU相比其他结合热探针方案的缓存替换策略,时间复杂度和空间复杂度更小,命中率更高。



CN 113050894 A

1.本专利对在谷鸟过算法的基础上提出了一种计数布谷鸟过滤器热探测方案(Count Cuckoo Filter Hot-probe Method,CCF)。

2.本专利设计了一种能自适应调整的自适应两级LRU缓存替换算法。

3.如权利要求1、2,通过将CCF和自适应两级LRU(Least Recently Used,LRU)相结合,形成CCF-LRU缓存替换策略,该策略利用CCF识别热数据。利用自适应两级LRU管理缓存。

一种基于布谷鸟算法的农业光谱混合存储系统缓存替换算法

技术领域

[0001] 本发明涉及农业光谱数据混合存储系统,通用基于布谷鸟算法,结合LRU缓存替换算法,提出一种基于计数布谷鸟过滤器热探测方案的混合存储缓存替换策略,已降低时间复杂度和空间复杂度,提高农业光谱混合存储系统缓存命中率。

背景技术

[0002] 随着5G、大数据、物联网农业等新技术的快速发展和应用,数据产生的规模和速度呈现指数式增长。海量增长得数据给存储系统带来巨大挑战,作为数据主要存储介质HDD(Hard Disk Drive,HDD)由于其固有的物理特性限制,性能提升已经遇到瓶颈。SSD(Solid State Drive,SSD)是一种新型存储介质,其读写性能大大优于HDD,但SSD的单位成本要远远高于 HDD。用SSD完全替换HDD的成本开销是十分昂贵。因此,采用SSD和HDD构建混合存储系统,是一种兼顾性能与成本的优秀方案。

[0003] SSD-HDD混合存储系统一般将SSD作为HDD的缓存,利用SSD的高性能和HDD低成本大容量的特点,为用户提供高性能、大容量的存储系统。由于SSD价格较高,所以SSD存储空间有限,这样管理混合存储的缓存替换策略就成为性能提升的关键。性能较好的混合存储缓存替换策略,都会考虑数据的访问频率,以确定这个数据是冷数据还是热数据,根据冷、热不同替换代价,设计最优的缓存替换策略。因此,判定一个数据冷、热的热探测(或冷探测) 是核心技术。

[0004] 本专利基于布谷鸟过滤器,提出一种计数布谷鸟过滤器热探测方案。并将CCF和自适应两级LRU相结合,形成CCF-LRU缓存替换策略。该策略利用CCF识别热数据,利用自适应两级LRU管理缓存。实验结果表明,结合热探针方案的缓存替换策略与传统策略相比,能显著提高缓存命中率。CCF-LRU相比其他结合热探针方案的缓存替换策略,时间复杂度和空间复杂度更小,命中率更高。

发明内容

[0005] 本专利描述的农业光谱混合存储系统,SSD是作为介于HDD和内存之间的缓存,SSD存储的是HDD中的数据副本,图6描述系统主要组件之间的关系和数据交互路径。黑色箭头表示数据流动路径,当内存发出读请求时,CCF-LRU判断SSD是否命中,如果命中则从SSD中读取数据,如未命中则从HDD中读取数据,并复制一个副本给SSD。当内存发出写请求时,则先写入SSD,然后延迟写回HDD。

[0006] 本专利将CCF和自适应两级LRU相结合,形成CCF-LRU策略,该策略利用CCF识别热数据,利用自适应两级LRU管理缓存。如图4所示,CCF-LRU维护冷和热两级LRU链,热链用于存储热数据,冷链用于存储冷数据。热链的尾端和冷链的首端相连,热链中被驱逐的数据,插入冷链首端。

[0007] 当有访问请求时,判断是否命中。如命中,如图5(a)所示,则返回数据LAB地址,并通过CCF判断命中的是热数据还是冷数据,如果是热数据则放入热链首端,如果是冷数据则

放入冷链首端。如未命中，如图5(b)所示，则从外存读取数据放入LRU链中，再返回数据。首次读取数据默认是冷数据，插入冷LRU链的首端，如果此时LRU链已满，则通过CCF判断冷链尾端数据，如果是冷数据，则直接驱逐，如果是热数据则重新插入到热链的首端，再重复上一步操作，直到有数据被驱逐。CCF-LRU缓存替换策略见算法5。

[0008]

Algorithm 5:CCF-LRU

Input: Access request of x
Output: LBA address of x

```
If LRU list has Access request of x then
    Judging whether x is hot data or cold data by CCF
    If x is hot data then
        Insert x into the head end of Hot list `
    Else
        Insert x into the head end of Cold list `
Else
    do
        Read the tail end of Cold list y
        Judging whether y is hot data or cold data by CCF
        If y is hot data then
            Insert y into the head end of Hot list `
    while(y is cold data)
    Discard y
return LBA address of x
```

[0009] 两级LRU缓存替换策略的链表长度一般是固定，或由人工调整，很难适应不同的工作场景和变化的工作负载。大量实践表明，两个适当的链表长度比例可以提高命中率，因此自适应调整链表长度是很有意义的。

[0010] 在CCF-LRU中，两个链表的长度在工作过程中能自适应调整，LRU链表总长度等于热链表和冷链表之和$L=L_{hot}+L_{cold}$，L_{hot}在0.2L到0.8L范围内调整。初始状态$L_{hot}=0.2L$、$L_{cold}=0.8L$，每经过Q次访问后，比较热链队尾和冷链队头0.01L长度的链表中热数据块的个数，如果热链热数据块多，则冷链队头0.01L长度的链表归并到热链队尾，如果冷链热数据块多，则热链队尾0.01L长度的链表归并到冷链队头，如果数据块相等，则不操作。从而实现自适应调整两个链表长度。自适应两级LRU如图6所示

附图说明

[0011] 图1:布谷鸟过滤器结构示意图。

[0012] 图2:计数布谷鸟过滤器热探针法模型示意图。

[0013] 图3：LRU缓存替换策略示意图。

[0014] 图4 :CCF-LRU缓存替换策略两级LRU链示意图。

[0015] 图5:CCF-LRU缓存替换策略示意图。

[0016] 图6:自适应两级LRUU缓存替换策略示意图。

[0017] 图7:农业光谱混合存储系统概述。

具体实施方式

[0018] 1、布谷鸟过滤器

[0019] 布谷鸟算法的布谷鸟过滤器由哈希表由n个存储桶组成，每个存储桶可以存储b个条目。每个项目先通过哈希计算得到一个j位的指纹f，公式1，再由哈希函数$h_1(x)$和$h_2(x)$确定两个候选的存储桶索引，公式2-3。

[0020] $f=fingerprint(x)$ (1)

[0021] $h_1(x)=hash(x)$ (2)

[0022] $$h_2(x)=h_1(x)\oplus hash(f) \quad (3)$$

[0023] 公式3中的异或运算确保$h_1(x)$也可以通过$h_2(x)$和指纹f异或得到。换言之，如果有两个桶a和b，桶a中存储有指纹f，可以通过桶a的索引η_a和指纹f异或得到另外一个桶的索引η_b，见公式4

[0024] $$\eta_b=\eta_a\oplus f \quad (4)$$

[0025] CF利用以上特性完成插入、查询和删除操作。图1左半部分演示了有8个桶，每个桶有 4个条目的CF(n=8，b=4)。当要插入新元素x_i时，CF通过公式1-3计算两个候选的存储桶索引η和指纹f_i，如果候选桶中有空的条目，则写入指纹f_i，如果两个桶都满，CF随机重置一个桶中的条目，如图2中为f_4，将指纹f_i写入条目。受害者f_4根据公式4计算另外一个候选桶的索引η_b，如果候选桶有空条目则写入指纹f_4，如果候选桶也没有条目，则CF再个候选桶中的条目，如图2中为f_{12}，写入指纹f_4，受害者f_{12}重复以上过程，直到所有的指纹都找到自己的条目或重复次数大于最大踢出数，最后一个受害者指纹被抛弃为止，插入操作完成。

[0026] CF的查询和删除操作较为简单，如图2右半部分，查询元素x_8，CF通过公式1-3计算两个候选的存储桶索引和指纹f_8，在候选的存储桶中查询是否有指纹f_8，有则返回Ture，没有则返回False。删除操作在查询操作基础上，返回Ture则执行删除指纹f操作，返回False不操作。

[0027] 2、LRU缓存替换策略

[0028] LRU是一种常用的缓存替换策略，如图3选择最近最久未使用的页面予以淘汰。该策略使用一个链表保存数据。新访问的数据的插入到链表首端。链表中数据被访问，则将数据移到链表首端。当链表满的时候，将链表尾端的数据丢弃。LRU缓存替换策略实现简单，命中率高，但没有考虑访问频率，且偶发性的、周期性的批量操作会导致命中率急剧下降，缓存污染情况比较严重。

[0029] 3、基于计数布谷鸟过滤器热探测方案

[0030] 本专利提出一种基于计数布谷鸟过滤器热探测方案，该方案在CF哈希表的基础上，维护一个计数表，计数表和哈希表一样，由n个桶组成，每个桶有b个条目。每个条目存储一个 N位计数器，当哈希表中的条目发生时，对应的计数表也发生相应变化。每经过Q次访问，计数器值折半(计数器向右移1位)，如图2。

[0031] 插入：当有插入请求时，先计算请求的LBA对应的指纹f和两个候选桶索引，判断两个候选桶内有无空条目，如有，则写入指纹f，相应的计数器设为1，若没有，则随机选择一个受害指纹，将受害指纹和相应计数器暂存，再驱逐受害指纹和相应计数器，写入指纹f，相应的计数器设为1。被驱逐的受害指纹，利用公式4，计算备选桶索引，再判断备选桶有无空条目，如没有，空条目，则再次执行上一步骤操作，直到备选桶有空条目，或者重复执行次数大

于最大踢出数，抛弃受害指纹和相应的计数器，如有空条目，判断备选桶里有无受害指纹，如没有，则写入受害指纹和相应的计数器，如有受害指纹，则抛弃受害指纹和相应的计数器。如算法1所示。

Algorithm 1: Insert(x)

```
1: f = fingerprint(x)
2: i_1 = hash(x)
3: i_2 = i_1 ⊕ hash(f)
8:   If bucket[i_1] or bucket[i_2] has an empty entry then
9:     add f to that bucket
10:    Get the index ζ of f
11:    Find the count table by index ζ and add 1
12:    return Done
13:  Else
14:    i=randomly pick i_1 or i_2 // must relocate existing items
15:    for (n = 0; n < MaxNumKicks; n++) do
16:      If bucket[i_1] or bucket[i_2] has f then
17:        Discard f
18:        return Done
19:      randomly select an entry e from bucket[i]
20:      swap f and the fingerprint stored in entry e
21:      the same swap the data in the count table
22:      i = i ⊕ hash(f)
23:      If bucket[i] has an empty entry then
24:        add f to that bucket[i]
25:        Get the index ζ of f
26:        Find the count table by index ζ and add to that bucket[i]
27:        return Done
28:      return Failure // Hashtable is considered full;
```

[0032]

[0033] 访问：当有访问请求时，先计算请求的LBA对应的指纹f和两个候选桶索引，然后判断两个候选桶中有无指纹f。若有，则相应的计数器累加1。若没有，进行插入操作，如算法2所示。

Algorithm 2: Access(x)

```
1: f = fingerprint(x)
2: i_1 = hash(x)
3: i_2 = i_1 ⊕ hash(f)
3: If bucket[i_1] or bucket[i_2] has f then
4:   Get the index ζ of f
5:   Find the count table by index ζ and add 1
6:   return Done
7: Else
7:   Insert(x)
```

[0034]

[0035] 删除：当有删除请求时，先计算请求的LBA对应的指纹f和两个候选桶索引，判断两个候选桶中有无指纹f，若有则删除指纹f和对应计数器，若无则不操作，如算法3所示。

[0036]

Algorithm 3: Delete (x)

$f = fingerprint(x)$
$i_1 = hash(x)$
$i_2 = i_1 \oplus hash(f)$
If bucket[i_1] or bucket[i_2] has f **then**
    Delete f from this bucket
        Get the index ζ of f
        Find the count table by index ζ and reset to 0
        return True
return False

[0037]

[0038] 冷、热判定：当有判定请求时，先计算请求的LBA对应的指纹f和两个候选桶索引，判断指纹f是否在两个候选桶中。如果不在，则LBA为首次访问，直接判定为冷数据；如果在，则查找对应的计数表中的数值。左侧K 位数值大于0，即该LAB地址访问次数不小于$2^{N-K}-1$时，判定为热数据，否则判定为冷数据。如算法4所示。

Algorithm 4: Decide(x)

Input: LBA address of x
Output: Hot or Cold
$f = fingerprint(x)$
$i_1 = hash(x)$
$i_2 = i_1 \oplus hash(f)$
If bucket[i_1] or bucket[i_2] has f **then**
    Get the index ζ of f
    g is the first K most significant bits of count table by index ζ
     If $g > 0$ then
       return Hot
Else
     return Cold

[0039]

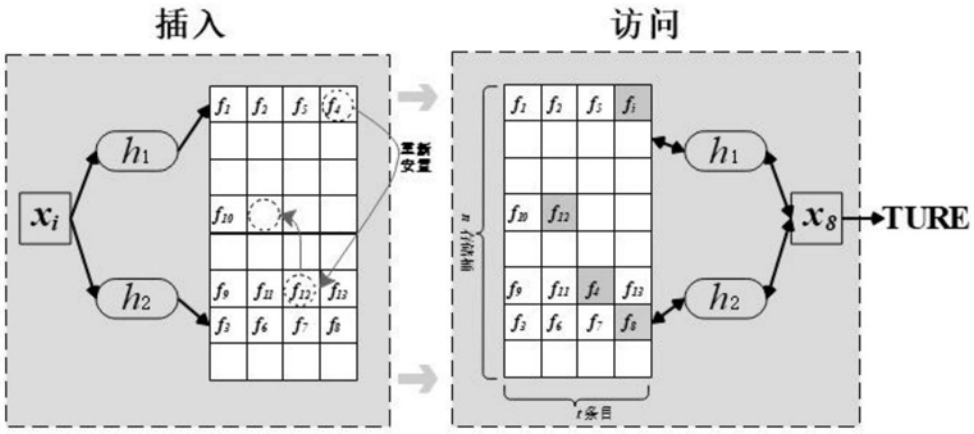

插入
访问
x_i
h_1
h_2
重新安置
x_8
TURE

图1

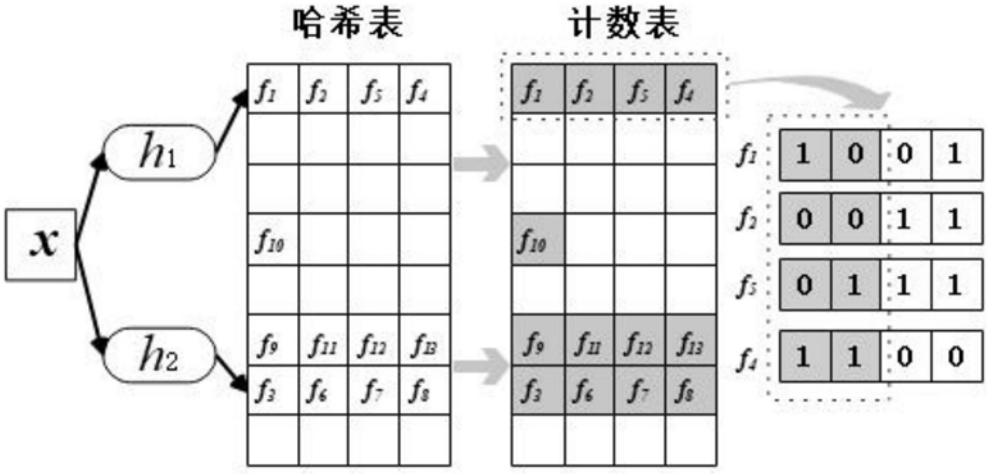

哈希表
计数表
x
h_1
h_2
1 0 0 1
0 0 1 1
0 1 1 1
1 1 0 0

图2

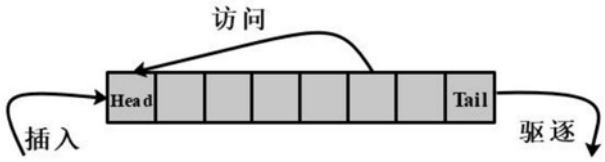

访问
Head
Tail
插入
驱逐

图3

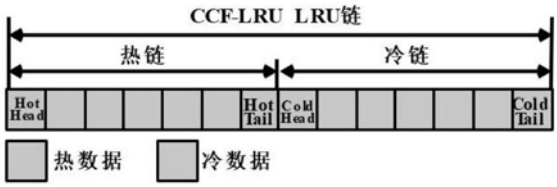

CCF-LRU LRU链
热链
冷链
Hot Head
Hot Tail
Cold Head
Cold Tail
热数据
冷数据

图4

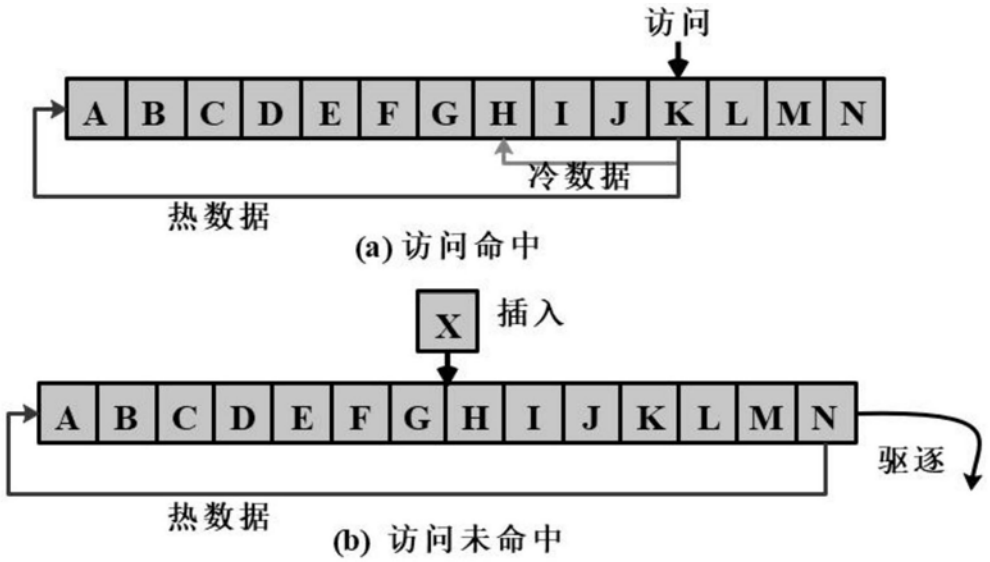

访问
A B C D E F G H I J K L M N
冷数据
热数据
(a) 访问命中
X
插入
A B C D E F G H I J K L M N
驱逐
热数据
(b) 访问未命中

图5

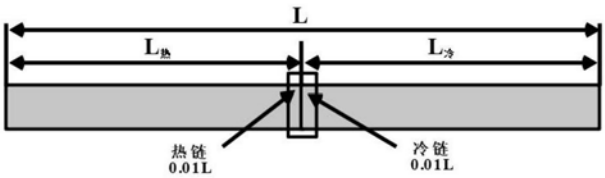

L
L热
L冷
热链
0.01L
冷链
0.01L

图6

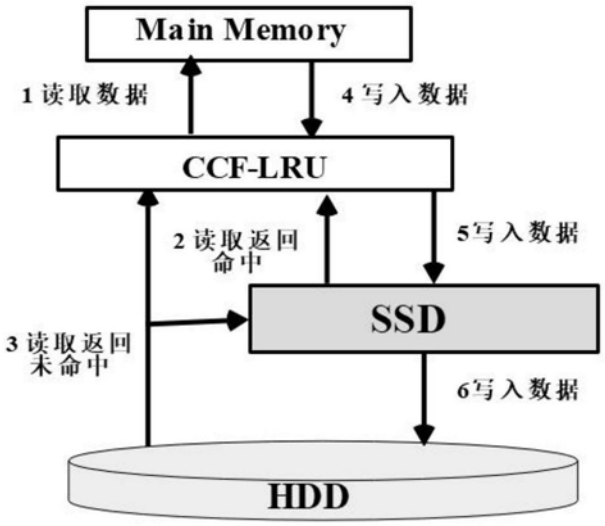

Main Memory
1 读取数据
4 写入数据
CCF-LRU
2 读取返回
命中
5写入数据
SSD
3 读取返回
未命中
6写入数据
HDD

图7