(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 106055679 A
(43)申请公布日 2016.10.26

(21)申请号 201610397148.2

(22)申请日 2016.06.02

(71)申请人 南京航空航天大学
地址 211106 江苏省南京市江宁区将军大道29号

(72)发明人 秦小麟 史太齐 刘亮 王宁 夏斌

(51)Int.Cl.
G06F 17/30(2006.01)

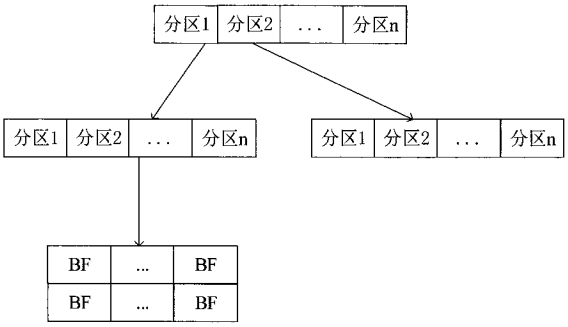
权利要求书1页 说明书5页 附图2页

(54)发明名称

一种多层次缓存感知型索引方法

(57)摘要

本发明涉及一种数据库多层次缓存感知型索引机制，属于计算机数据库中的数据库高效查询技术领域。本发明通过对不同层次存储介质的行为特点和物理特性的研究，将索引结构划分为两个层次，并对不同层次根据存储介质的不同进行特别优化。该索引树的内部结点与B+树的内部结点类似，内部结点针对主存-缓存层次，根据缓存存取机制，采用内部结点分区的方式提高结点扇出度。对内部结点分区内部关键字采用压缩编码的方式，提高索引内部节点对主存空间的利用率，提高缓存命中率。索引树的叶结点是针对磁盘-主存层次，采用概率论中的布隆过滤器技术，在内存中建立存储关键字的页面的布隆过滤器，并组织成索引叶节点，减少查询过程中的磁盘交换次数。



CN 106055679 A

1.一种多层次缓存感知型索引方法，其特征在于包括如下步骤：

(1)多层次缓存感知型索引内部结点在B^+-树内部结点基础上进行修改，首先对内部结点分区，将内部结点分成占用相同主存空间的分区；

(2)索引内部结点分区之后，对分区中的关键字进行编码。在分区中存放的关键字不是关键字原始数值而是编码之后的数值，编码可以提高索引结构对主存空间的利用率；

(3)对于索引叶节点，采用概率论中的布隆过滤器技术，对存放关键字的页面映射成一个布隆过滤器。布隆过滤器中存储着该页面的成员信息，查找关键字时，首先查找布隆过滤器，确定关键字是否在该过滤器对应的页面，如果在，再查找对应的页面。

2.如权利要求1的一种多层次缓存感知型索引方法，其特征要对索引结构内部结点分区，包括：

首先，根据缓存块的大小对内部结点进行分区，将内部结点划分为与缓存块大小相同的分区。对内部结点分区之后，每个分区中携带有部分关键字信息，为了方便定位到每个分区，在内部结点头部增加内部结点分区信息索引。内部结点头部的分区信息索引中存放的关键字可以帮助快速定位到特定分区。在查找关键字时，首先搜索内部结点分区信息索引，根据分区信息索引定位到特定的分区，然后在该分区中查找待查找关键字。

3.如权利要求2所述的一种多层次缓存感知型索引方法，其特征在于在对内部结点分区时，还包括：

内部结点在分区之后，对分区中的关键字进行编码。分区中的关键字并不是原始数值，而是经过编码之后的数值，对每个分区中的关键字进行编码，将整型关键字转化为字节数组。由于整型数值按照数值大小可以用不同长度的字节数组进行存储，为了便于查找编码之后的关键字，在分区头部增加分区头部索引信息，存储不同长度的字节数组在分区中的起始地址。在分区中查找关键字时，此时待查找关键字已经经过编码，因此可以直接根据分区头部索引信息定位到分区中的特定位置，并从该位置开始继续查找关键字。

4.一种多层次缓存感知型索引方法，其特征在于，包括以下步骤：

除了对索引内部结点分区和对分区中的关键字编码之外，索引还对叶节点进行修改，对存放关键字的页面构建布隆过滤器。每个关键字实际存放在磁盘的页面上，构建叶节点时，对存放关键字的页面进行编号，同时为每个页面创建多个hash函数，通过不同的hash函数为每一个页面构建一个布隆过滤器。布隆过滤器存放着与之对应的页面成员信息，当查找的叶节点时，对叶节点中的每一个布隆过滤器进行查找，检测待查找关键字是否存在于布隆过滤器对应的页面，如果存在，再将页面从磁盘读入主存，然后对该页面进行查找。在一个叶结点中查找时，采用并行的方式查找叶结点中的所有过滤器。

一种多层次缓存感知型索引方法

技术领域

[0001] 本发明涉及一种多层次缓存感知型索引方法，属于计算机空间数据库中的高效查询技术领域。

背景技术

[0002] 专利[1]则对内存页面进行了优化，提高了主存和磁盘层次的效率，对于主存-缓存层次则没有涉及到。文献[5]提出的B+树索引方法在传统数据库中占据着重要的地位。为了提升B^+-树的缓存感知能力，Jun Rao在文献[2]中提出了它的变种CSB^+树(Cache Sensitive B^+-Trees)。CSB^+树的更新操作类似于B^+树，与B^+树不同的是，CSB+树的每一个结点只保留了很少的指针。通过减少结点中指针的数量，相同的缓存空间可以保留更多的关键字，从而表现出更加优秀的性能。文献[3]提出的T-树索引方法，在总体性能上表现优良，自问世以来就被大部分主存数据库所采用。但是T-树的缓存感知能力不如B^+树。在对T-树进行搜索时，首先比较结点中的最大值和最小值，然后再决定搜索左右子树。当T-树的一个结点被放在缓存中时，CPU访问其中的最大值和最小值，而缓存块中剩余的关键字不再被访问。由此可以看出T-树的缓存空间利用率非常低下。T-树已经无法适应处理器速度和主存访问速度发展不平衡的状况。Ig-hoon Lee等人在文献[4]中提出CST-树索引方法，通过建立结点组(Node group)和数据结点(Datanode)，将T-树结点中的最大值和最小值提取出来和结点中剩余的关键字分别存放的方式，增强被频繁访问数据的局部性特性。同时，专利[2]提出的技术，采用硬件预取技术，可以扩大树结点的容量，提高查询效率，但是依赖于计算机自身的硬件资源。

[0003] 上文中提到的专利引用自如下专利：

[0004] [1]Hsu L R,O′connor J M.System and Method for Page-Conscious GPU Instruction:U.S.Patent 20,160,055,005[P].2016-2-25.

[0005] [2]Bolger R M,Corrao A,Hamilton R A,et al.Pre-fetching items in a virtual universe based on avatar communications:U.S.Patent 8,990,306[P].2015-3-24.

[0006] 上文中提到的文献来源于如下的期刊：

[0007] [1]Silva J,Sklyarov V,Skliarova I.Comparison of On-chip Communications in Zynq-7000All Programmable Sys-tems-on-Chip[J].Embedded Systems Letters,IEEE,2015,7(1):31-34.

[0008] [2]Kocberber O,Grot B,Picorel J,et al.Meet the walkers:Accelerating index traversals for in-memory data-bases[C]//Proceedings of the 46th Annual IEEE/ACM In-ternational Symposium on Microarchitecture.ACM,2013:468-479.

[0009] [3]Ding C,Li P.Cache-conscious memory manage-ment[C]//Proceedings of the ACM SIGPLAN Workshop on Memory System Performance and Correctness.2014.

[0010] [4]Gray J.Tape is dead,disk is tape,flash is disk,RAM local-ity is

king[J].Gong Show Presentation at CIDR,2007.

[0011] [5]Lee I,Shim J,Lee S,et al.CST-trees:cache sensitive t-trees[M]// Advances in Databases:Concepts,Systems and Applications.Springer Berlin Heidelberg,2007:398-409.

发明内容

[0012] 本发明为解决的技术问题：

[0013] 本发明的目的是提出一种多层次缓存感知型索引方法，以提升主存索引对缓存行为的感知能力，并针对不同层次的储存介质进行优化，使得索引方法能够适应计算机存储介质多层次的应用场景。通过本发明提出的方法，可以使得主存索引方法能够在大数据时代，主存无法存储所有数据而调用磁盘时，也能保证索引方法的效率。同时，针对缓存行为特点，提高索引结构对存储空间的利用率，进一步提升索引结构的缓存感知能力，以实现更高效的数据库主存索引方法。

[0014] 本发明为解决其技术问题采用如下技术方案：

[0015] 一种多层次缓存感知型索引方法，包括如下步骤：

[0016] (1)多层次缓存感知型索引内部结点在B+-树内部结点基础上进行修改，首先对内部结点分区，将内部结点分成占用相同主存空间的分区；

[0017] (2)索引内部结点分区之后，对分区中的关键字进行编码。在分区中存放的关键字不是关键字原始数值而是编码之后的数值，编码可以提高索引结构对主存空间的利用率；

[0018] (3)对于索引叶节点，采用概率论中的布隆过滤器技术，对存放关键字的页面映射成一个布隆过滤器。布隆过滤器中存储着该页面的成员信息，查找关键字时，首先查找布隆过滤器，确定关键字是否在该过滤器对应的页面，如果在，再查找对应的页面。

[0019] 其中对索引结构内部结点分区，包括：

[0020] (1)根据缓存块的大小对内部结点进行分区，将内部结点划分为与缓存块大小相同的分区；

[0021] (2)对内部结点分区之后，每个分区中携带有部分关键字信息，为了方便定位到每个分区，在内部结点头部增加内部结点分区信息索引；

[0022] (3)内部结点头部的分区信息索引中存放的关键字可以帮助快速定位到特定分区。在查找关键字时，首先搜索内部结点分区信息索引，根据分区信息索引定位到特定的分区，然后在该分区中查找待查找关键字；

[0023] 在对内部结点分区并编码时，还包括：

[0024] (1)压缩编码：内部结点在分区之后，对分区中的关键字进行编码。分区中的关键字并不是原始数值，而是经过编码之后的数值，对每个分区中的关键字进行编码，将整型关键字转化为字节数组；

[0025] (2)分区头部索引信息区：由于整型数值按照数值大小可以用不同长度的字节数组进行存储，为了便于查找编码之后的关键字，在分区头部增加分区头部索引信息，存储不同长度的字节数组在分区中的起始地址；

[0026] (3)在分区中查找关键字时，此时待查找关键字已经经过编码，因此可以直接根据分区头部索引信息定位到分区中的特定位置，并从该位置开始继续查找关键字。

[0027] 叶结点中的布隆过滤器技术包括如下步骤：

[0028] (1)每个关键字实际存放在磁盘的页面上，构建叶节点时，对存放关键字的页面进行编号；

[0029] (2)为每个页面创建多个hash函数，通过不同的hash函数为每一个页面构建一个布隆过滤器。布隆过滤器存放着与之对应的磁盘页面中存放的关键字成员信息，可以通过布隆过滤器快速的判定关键字key是否保存在对应的磁盘页面中；

[0030] (3)在叶结点中查找时，采用多线程技术，并行的检查叶结点中的所有布隆过滤器，判断关键字key是否在布隆过滤器中。如果存在，再将对应的磁盘页面调入主存，并查找页面。

[0031] 本发明采用以上技术方案与现有技术相比，具有以下有益效果：

[0032] (1)本发明基于CSB+树索引结构，提出对内部结点进行分区，可以扩大内部结点的容量，使得一个结点不再局限于一个缓存块大小，可以横跨多个缓存块大小的存储空间而不会增加查找时的缓存失配次数，降低了索引树的高度，加快了从根节点到索引树叶节点的查询速度。

[0033] (2)本发明对索引内部结点中的关键字进行编码，将关键字编码之后可以降低储存关键字耗费的存储空间，使得同样大小的结点容量可以存储更多的关键字。编码之后，提高了索引结构对主存空间的利用率，加快了查询效率，提高了缓存命中率。

[0034] (3)本发明针对磁盘速度远远落后于主存速度的现状，采用布隆过滤器技术将磁盘页面映射成为布隆过滤器，并将存储一定范围关键字的页面所对应的布隆过滤器组织成索引结构的叶节点。通过布隆过滤器可以快速确认对应的磁盘页面中是否存放关键字key，从而降低了磁盘I/O交换次数，提高了索引方法的性能。

附图说明

[0035] 图1是索引方法原理示意图。整棵树类似于B+树结构，内部结点与传统B+树内部结点不同之处在于将内部结点在逻辑上划分成大小相同的连续分区。每一个分区存储的关键字在物理上是连续、有序的。每个分区中的关键字对应的子结点组合在一起，并用指针指向组合结点的首地址。叶结点中BF代表一个布隆过滤器，如图所示，叶结点是由多个布隆过滤器组合而成，每一个布隆过滤器对应于一个磁盘物理页面。

[0036] 图2是索引内部结点分区结构示意图。图中H_k是内部结点头部索引区中的关键字，用以快速定位到内部结点特定分区。假设key_k代表分区k中的关键字序列，key_{k+1}代表分区k+1中的关键字序列，则H_k满足：$key_k < H_k < key_{k+1}$。

[0037] 图3是索引结点分区内部数据结构示意图。图中a、b、c分别代表经过编码之后的关键字的长度为1字节、2字节、4字节的关键字个数。图3表示的是内部结点中，每一个分区内部的数据结构，经过编码之后，关键字根据与该分区中最大关键字数值之间的差值的大小而占用不同大小的空间。假设key代表某一分区中的关键字，该分区中的最大关键字为key_n，则差值为$D(x)=key_n-key$。该关键字经过编码之后所占用的字节个数为：$space(key)=\left\lceil \frac{\log_2^{D(x)}}{8} \right\rceil$。

[0038] 图4是布隆过滤器示意图。图中BF代表一个布隆过滤器，每一个布隆过滤器的数据

结构是一个位数组，位数组中每一位对应着一个hash函数。一个布隆过滤器对应着一个磁盘页面，布隆过滤器包含着这个磁盘页面中关键字的信息，可以通过检索磁盘页面对应的布隆过滤器，判断待查找的关键字key是否在磁盘页面中。如果不存在，则不需要再检索磁盘页面，从而避免一次磁盘I/O交换。如果存在，再将磁盘页面读入主存，并扫面页面查找关键字。

具体实施方式

[0039] 下面结合附图对本发明创造做进一步详细说明。

[0040] 对于传统主存数据库索引方法，采用指针消除和指令预取机制提高索引方法对缓存行为的感知能力，这些做法没有考虑索引结构对存储空间的影响，面对大量数据记录时，索引本身需要消耗大量的主存空间，搜索指定关键字会造成多次缓存失配。本发明采用编码的方式对索引结构中存放的关键字进行编码。编码可以降低关键字对存储空间的消耗，使得同样大小的存储空间可以存放更多的关键字。在查找关键字的过程中，需要对关键字进行编码和解码，这一部分是有中央处理器(CPU)负责的，通过编码的方式，将制约索引方法的部分因素转移到更加高效的CPU上，可以充分利用已有计算机硬件资源的潜力和性能。

[0041] 同时对结点进行分区。分区可以使得结点内部的查找限制在一个特定的范围——分区内部进行，不需要搜索全部结点信息。同时，根据缓存的行为特点再加上指令预取机制，将分区容量设定为缓存块大小的倍数，可以防止在分区内部查找时引起缓存失配。将结点分区之后，一个结点可以包含很多个容量相等的分区，每一个分区与传统索引方法中的结点相等，这样可以使得结点的容量在逻辑和物理层面上得到扩大。增加结点容量，会使得包含同样数量的索引结构的高度降低，从而在查找关键字过程中，从根结点到叶结点的查找速度加快，提升定位到最终叶结点的效率。同时，索引结构高度的降低，可以减少因结点转移而造成的缓存失配次数，提高索引方法对缓存行为的感知能力。

[0042] 随着索引数据量的增大，索引结构中的数据不适合全部装载在主存中，需要将一部分数据存放在磁盘上，在需要的时候再取到主存中。磁盘的访问速度远远落后于主存的访问速度，因此在查找过程中，如果频繁读写磁盘，会造成索引性能的下降。本发明创造采用统计学中的布隆过滤器技术对索引叶结点加以改造，可以减少查找过程中的I/O交换次数。首先对存放关键字的磁盘页面进行编号，然后对每一个页面构造一个布隆过滤器BF，BF中包含着对应页面的成员信息，可以根据BF判断待查找的关键字是否存在于对应的磁盘页面中。BF是一个m位的位数组，占用很少量的主存空间，而与BF对应的页面是一个很大的存储空间，通常为4KB～5KB，通过m位的布隆过滤器可以很快的判断出关键字是否在对应的页面，可以减少对磁盘的访问次数，从而提高查找的效率。

[0043] 下面通过说明书附图对各实施例对本发明进行说明。

[0044] 1)实施例一

[0045] 本发明的实施例一介绍了查找关键字时如何在内部结点查找并定位到叶结点的方法，多层次缓存感知型索引方法的内部结点的结构如图2所示，内部节点中分区的数据结构如图3所示。包括：

[0046] A、读取索引树根结点头部分区信息索引，根据内部结点头部的分区信息索引定位到相应分区；

[0047] B、对待查找的关键字编码；

[0048] C、定位到相应分区之后，读取分区头部的编码信息索引，根据索引信息定位到分区的相应区域；

[0049] D、在分区的特定区域查找关键字，并与编码后的待查找的关键字对比，然后定位到相应的子结点；

[0050] E、重复以上过程，直至查找到叶结点，然后在叶结点中继续查找关键字。

[0051] F、读取叶结点中存储的关键字key的范围[min_key，max_key]，如果待查找关键字属于该范围执行步骤G。否则执行步骤I；

[0052] G、在当前叶结点中，查找所有的布隆过滤器BF，判断该BF是否包含有key的信息

[0053] H、将含有key的布隆过滤器对应的页面全部读取到主存中，并对每一个页面进行查找，并返回对应的记录，结束；

[0054] I、返回关键字key不在存在。

[0055] 2)实施例二

[0056] 本发明的实施例三介绍了在叶结点中插入关键字所在磁盘页面的方法。具体步骤如下所示：

[0057] A、如果叶结点中包含的关键字数量小于阈值，执行步骤B。否则，执行步骤E

[0058] B、确定存放关键字key的页面编号pid；

[0059] C、如果$key \notin [\min_key, \max_key]$，扩展min_key或者max_key并增加结点中的关键字数量标志：

[0060] D、根据pid-min_pid得到叶结点中对应于页面的布隆过滤器，并将key插入到当前叶结点

[0061] 的布隆过滤器中，结束。

[0062] E、对于要分裂的叶结点N，创建两个新结点N_1、N_2；

[0063] F、设定N_1包含的关键字key的范围：$\left[N.\min\_key, \frac{N.\min\_key + N.\max\_key}{2}\right]$；

[0064] G、设定N_2包含的关键字key的范围：$\left[\frac{N.\min\_key + N.\max\_key}{2}, N.\max\_key\right]$；

[0065] H、对于N中的关键字key，如果key在N_1的范围内，则根据key所在页面pid更新N_1的pid范围，同时将N_1中对应的布隆过滤器中的标志位置1。否则，则更新的N_2范围，并将N_2中对应的布隆过滤器的标志位置1；

[0066] I、跳转到A。

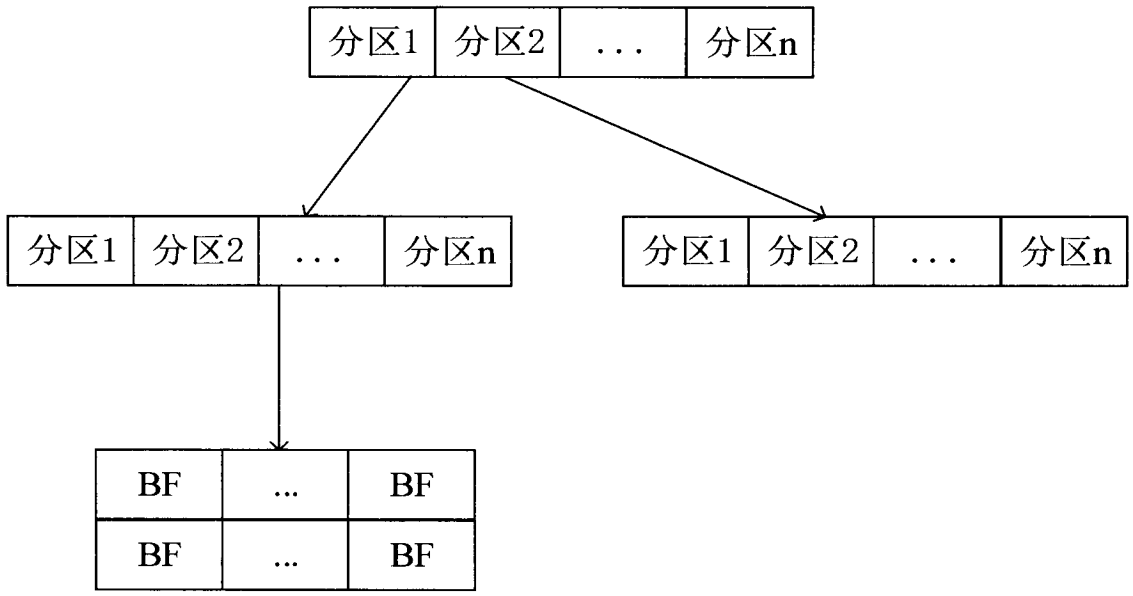

分区1
分区2
...
分区n
分区1
分区2
...
分区n
分区1
分区2
...
分区n
BF
...
BF
BF
...
BF

图1

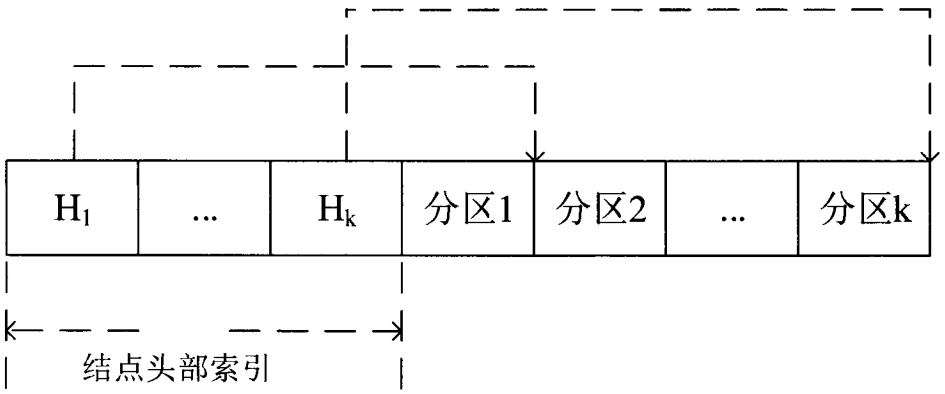

H1
...
Hk
分区1
分区2
...
分区k
结点头部索引

图2

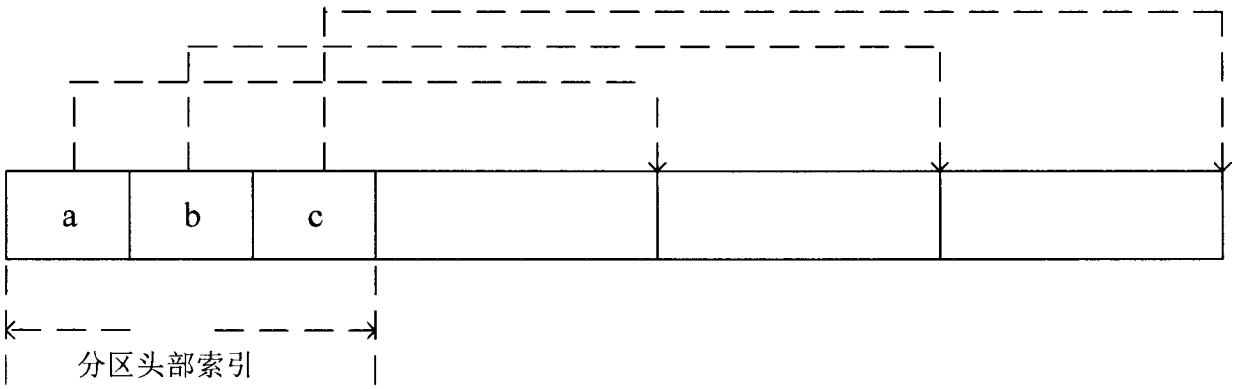

a
b
c
分区头部索引

图3

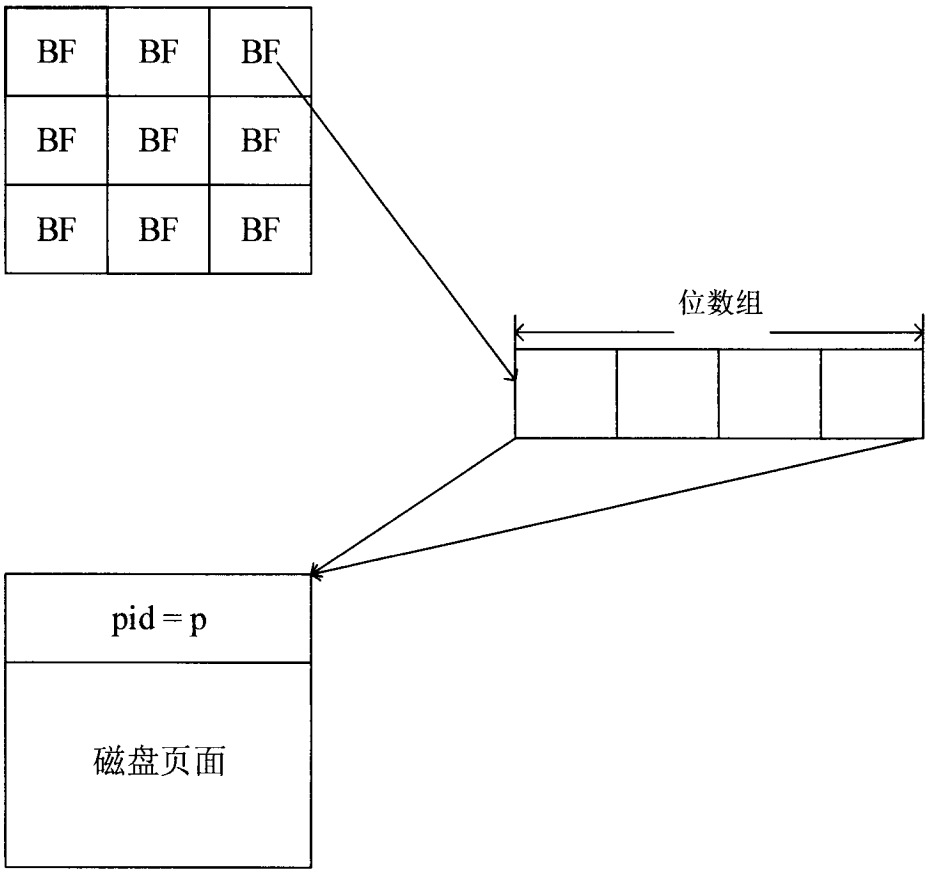


图4