(19)中华人民共和国国家知识产权局

(12)发明专利

(10)授权公告号 CN 102253820 B
(45)授权公告日 2013.03.20

(21)申请号 201110162619.9

(22)申请日 2011.06.16

(73)专利权人 华中科技大学
地址 430074 湖北省武汉市洪山区珞喻路1037号

(72)发明人 周可 魏建生 张攀峰 李春花 王桦

(74)专利代理机构 华中科技大学专利中心 42201
代理人 李智

(51)Int.Cl.
G06F 5/06(2006.01)

(56)对比文件
CN 102082575 A,2011.06.01, 全文.
CN 101963982 A,2011.02.02, 全文.
袁志坚,等.. 典型Bloom 过滤器的研究及其数据流应用.《计算机工程》.2009, 第35卷(第7期), 第5～7页.
Kai Cheng,et al..Time-Decaying Bloom Filters for Data Streams with Skewed Distributions.《15th International Workshop on Research Issues in Data Engineering: Stream Data Mining and Applications》.2005,
王树鹏。. 重复数据删除技术的发展及应用.《中兴通讯技术》.2010, 第16卷(第5期), 第9～14页.

审查员 刘芬

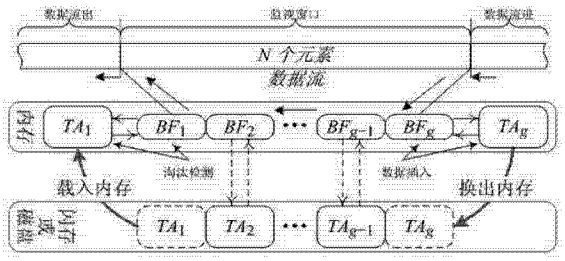
权利要求书 1 页 说明书 5 页 附图 2 页

(54)发明名称

一种流式重复数据检测方法

(57)摘要

本发明提供了一种流式重复数据检测方法,通过构建一个计时型布隆过滤器阵列TBFA,在滑动窗口模型内灵活高效地检测重复数据。TBFA由多个同构的计时型布隆过滤器TBF构成,每个TBF包含一个布隆过滤器和一个分离的用于保存时间戳的计时器组,整个TBFA以一种循环先入先出队列的方式工作,在记录新元素的同时,淘汰已经移出数据流监控窗口的旧元素。本发明在滑动窗口模型下工作,对元素的监测可以精确到一个元素,从而使基于本发明的统计结果具有稳定性,另外TBFA中的部分计时器组可以被卸载到磁盘中,从而减少内存开销。理论分析和实验数据表明,DCBA在加载10%以内数据内容到内存的情况下,能够保持95%以上的查询效率,从而使本发明在空间效率和可扩展性上优于现有技术方案。



CN 102253820 B

1. 一种流式重复数据检测方法，涉及由多个计时型布隆过滤器 TBF 在逻辑上构成的循环先入先出队列，将逻辑上从队首到队尾的 TBF 依次计为 TBF_1，TBF_2，…，TBF_g，g 为 TBF 的个数，每个 TBF 包含一个位向量 BV 和一个计时器组 TA，同时与其它 TBF 共享 k 个哈希函数和一个全局基础时钟；其中，位向量用于插入数据元素，计时器组用于记录插入数据元素时的时间戳，该检测方法具体为：

（1）插入数据元素 x：采用所述 k 个哈希函数分别对数据元素 x 进行计算得到 k 个哈希值 $h_1(x)$，$h_2(x)$，…，$h_k(x)$，将处于队尾的 TBF_g 所含位向量中偏移量为 $h_1(x)$，$h_2(x)$，…，$h_k(x)$ 的 k 个位的值分别置 1，同时将基础时钟的当前时间戳写入 TBF_g 所含计时器组中对应的 k 个计时器；若 TBF_g 此时已装满数据元素，则清空处于队首的 TBF_1 并将其置为队尾；

（2）检测数据元素 x 是否为重复数据：

（21）在 TBF_1，TBF_2，…，TBF_{g-1} 中查询是否存在一个 TBF，其偏移量为 $h_1(x)$，$h_2(x)$，…，$h_k(x)$ 的 k 个位的值全为 1，若存在，则说明该 TBF 插入过数据元素 x，进入步骤（22），否则，x 不为重复元素；

（22）判断插入过数据元素 x 的 TBF 是否是 TBF_1，若不是 TBF_1，则表明 x 为重复元素，若是 TBF_1，则进入步骤（23）；

（23）查询 TBF_1 插入 x 时的时间戳是否小于 TBF_g 插入 x 的时间戳，若小于，则表明 x 逻辑上已被 TBF_1 删除，x 不为重复元素；否则，表明 x 为重复元素。

一种流式重复数据检测方法

技术领域

[0001] 本发明属于计算机数据传输及存储系统，具体涉及一种数据流中的重复数据删除方法。

背景技术

[0002] 互联网的扩展使得数据信息呈几何级数爆炸性增长，图灵奖得主吉姆．格雷（Jim Gray）指出：网络环境下每 18 个月新增的数据量等于有史以来数据量的总和。数字图书馆、电子商务、医学影像、生物工程、科学计算、虚拟现实、数字化地球、网站多媒体等应用的不断发展，对建立高性能、高可靠的海量信息存储系统提出了需求，未来的存储系统其规模将达到 PB 级甚至 EB 级。海量的数据的传输与存储对网络系统及存储设备及服务器系统提出了非常高的要求，另一方面虚拟技术及云存储及各种网络应用的发展，使得大数据的流动变成了经常的事情。大量数据的转移、上载、下载给网络造成了沉重的负担，降低了用户的网络体验。同时大数据量的流动也加剧了存储系统的开销。但在这些数据流中实际上存在大量的数据冗余，即数据流中存在许多重复的数据块，这些冗余及重复的数据占据着大量的网络带宽和存储空间。而这些重复数据完全可以通过重复数据删除技术进行清洗，以达到节约存储空间提高带宽利用率的目的。

[0003] 关于重复数据删除技术，近几年来，国际上对重复数据的检测与删除进行了大量的研究并提出了几种重复数据检测方法。主要有：完全文件检测 (whole file detection，简称 WFD)、固定块 (fixed-sized chunking，简称 FSC) 检测技术、基于内容的块检测技术 (content-defined chunning，简称 CDC)、滑动块 (sliding block) 技术、shingle 检测技术、bloom filter 检测技术、模式匹配的检测技术等。这些重复数据检测技术尽管方法不同，但目的都是为了检测出存储系统中不同文件或数据对象间存在的相同数据块。流式重复数据检测，目前已知的有三种方法，(分段窗口模型（Landmark window model）、跳跃窗口模型（Jumping window model）、滑动窗口模型（Sliding window model）。分段窗口模型是按照等时或等长条件将数据流分成多个段，在每次检查重复数据时，只需放置一段于内存中。该方法的主要缺点是无法同时检测各段之间的重复数据。跳跃窗口模型是先定义一个能容纳 N 个数据段的滑动窗口，然后将数据流分成很多小段；每次从窗口的一端跳入一个数据段进入窗口，同时从窗口的另一端淘汰一个数据段；窗口内的所有数据段之间可以进行重复数据检测。该方法的主要缺点，是数据流不够流畅，同时无法精确分析重复数据检测的结果。滑动窗口模型仅仅维持最近的 N 个元素，当一个新元素到达时，同时淘汰到期的旧元素。该方法的主要缺点是当 N 的取值过大时检测的开销不可接受。

发明内容

[0004] 本发明的目的在于提出一种高效精准的流式重复数据检测方法，在减小内存开销的同时提高了查询效率和精准度。

[0005] 一种流式重复数据检测方法，涉及由多个计时型布隆过滤器 TBF 在逻辑上构成的

循环先入先出队列，将逻辑上从队首到队尾的 TBF 依次计为 TBF_1，TBF_2，…，TBF_g，g 为 TBF 的个数，每个 TBF 包含一个位向量 BV 和一个计时器组 TA，同时与其它 TBF 共享 k 个哈希函数和一个全局基础时钟；其中，位向量用于插入数据元素，计时器组用于记录插入数据元素时的时间戳，该检测方法具体为：

[0006] （1）插入数据元素 x：采用所述 k 个哈希函数分别对数据元素 x 进行计算得到 k 个哈希值 $h_1(x), h_2(x), \cdots, h_k(x)$，将处于队尾的 TBF_g 所含位向量中偏移量为 $h_1(x)$，$h_2(x), \cdots, h_k(x)$ 的 k 个位的值分别置 1，同时将基础时钟的当前时间戳写入 TBF_g 所含计时器组中对应的 k 个计时器；若 TBF_g 此时已装满数据元素，则清空处于队首的 TBF_1 并将其置为队尾；

[0007] （2）检测数据元素 x 是否为重复数据：

[0008] （21）在 $TBF_1, TBF_2, \cdots, TBF_{g-1}$ 中查询是否存在一个 TBF，其偏移量为 $h_1(x), h_2(x), \cdots, h_k(x)$ 的 k 个位的值全为 1，若存在，则说明该 TBF 插入过数据元素 x，进入步骤（22），否则，x 不为重复元素；

[0009] （22）判断插入过数据元素 x 的 TBF 是否是 TBF_1，若不是 TBF_1，则表明 x 为重复元素，若是 TBF_1，则进入步骤（23）；

[0010] （23）查询 TBF_1 插入 x 时的时间戳是否小于 TBF_g 插入 x 的时间戳，若小于，则表明 x 逻辑上已被 TBF_1 删除，x 不为重复元素；否则，表明 x 为重复元素。

[0011] 本发明的技术效果体现在：本发明通过构建一个计时型布隆过滤器阵列（Timing Bloom Filter Array，TBFA），在滑动窗口模型内灵活高效地检测重复数据。一个 TBFA 由多个同构的计时型布隆过滤器（Timing Bloom Filter，TBF）构成。而每个 TBF 包含一个布隆过滤器和一个分离的用于保存时间戳的计时器组。整个 TBFA 以一种循环先入先出队列（First-In First-Out，FIFO）的方式工作，在记录新元素的同时，淘汰已经移出数据流监控窗口的旧元素。该发明在滑动窗口模型下工作，对元素的监测可以精确到一个元素，从而使基于该发明的统计结果具有稳定性。TBFA 中的部分计时器组可以被卸载到磁盘中，从而减少内存开销。理论分析和实验数据表明，DCBA 在加载 10% 以内数据内容到内存的情况下，能够保持 95% 以上的查询效率，从而使该发明在空间效率和可扩展性上优于已有解决方案。

附图说明

[0012] 图 1 为布隆过滤初始状态示意图；

[0013] 图 2 为布隆过滤插入 x1 和 x2 后的状态示意图；

[0014] 图 3 为布隆过滤的检验示意图；

[0015] 图 4 为本发明单节点布隆过滤重删检测示意图；

[0016] 图 5 为本发明多节点布隆过滤重删检测示意图；

[0017] 图 6 为布隆过滤队列示意图。

具体实施方式

[0018] 本发明是针对数据流利用布隆过滤检测技术检测重复数据。在描述发明方案前先简要介绍一下布隆过滤（bloom filter）的工作原理。

[0019] Bloom Filter 是一种空间效率很高的随机数据结构，它利用位数组很简洁地表示

一个集合，并能判断一个元素是否属于这个集合。Bloom Filter 的这种高效是有一定代价的：在判断一个元素是否属于某个集合时，有可能会把不属于这个集合的元素误认为属于这个集合（false positive）。因此，Bloom Filter 不适合那些“零错误”的应用场合。而在能容忍低错误率的应用场合下，Bloom Filter 通过极少的错误换取了存储空间的极大节省。

[0020] 下面我们具体来看Bloom Filter是如何用位数组表示集合的。初始状态时，Bloom Filter 是一个包含 m 位的位数组，每一位都置为 0。见图 1。

[0021] 为了表达 $S=\{x_1, x_2, \ldots, x_n\}$ 这样一个 n 个元素的集合，Bloom Filter 使用 k 个相互独立的哈希函数（Hash Function），它们分别将集合中的每个元素映射到 {1,...,m} 的范围中。对任意一个元素 x，第 i 个哈希函数映射的位置 $h_i(x)$ 就会被置为 1（i=1,2...,k）。如果一个位置多次被置为 1，那么只有第一次会起作用，后面几次将没有任何效果。在图 2 中，k=3，且有两个哈希函数选中同一个位置（从左边数第 8 位）。

[0022] 在判断 y 是否属于这个集合时，我们对 y 应用 k 个哈希函数，如果所有 $h_i(y)$ 的位置都是 1（i=1,2...,k），那么我们就认为 y 是集合中的元素，否则就认为 y 不是集合中的元素。图 3 中 y1 就不是集合中的元素。y2 或者属于这个集合，或者刚好是一个“假阳性（false positive）”。

[0023] 前面提到，Bloom Filter 在判断一个元素是否属于它表示的集合时会有一定的错误率（假阳性率，false positive rate），下面就来估下计错误率的大小。在估计之前为了简化模型，假设 k，n ＜ m 且各个哈希函数是完全随机的。当集合 $S=\{x_1, x_2, \ldots, x_n\}$ 的所有元素都被 k 个哈希函数映射到 m 位的位数组中时，这个位数组中某一位还是 0 的概率是：

[0024] $$p' = \left(1 - \frac{1}{m}\right)^{kn} \approx e^{-kn/m}$$

[0025] 本发明是针对数据流，利用bloom filte技术检测数据流中重复数据的技术，通过采用布隆过滤队列及与之关联的计数器数组，能有效地检测数据流中的重复数据，同时提高内存的使用率，具体的设计如下：

[0026] 计时型布隆过滤器阵列的数据结构如图 4 所示。计时型布隆过滤器阵列（Timing Bloom FilterArray，TBFA）在滑动窗口模型下以循环先入先出队列方式工作。滑动窗口模型将大数据流抽象为无限长度的时序队列，并通过一个固定长度的窗口监控距离当前时间点最近的 N 个已知元素（N 的大小与布隆过滤器队列的容量相关，它的值与 TBFA 的容量相等）随着数据的更新和流动，窗口以一个元素为单位向前滑动，在记录一个新元素的同时淘汰一个旧元素，保持尺寸不变。数据元素可以是数据块，或者是文件构成。如果是由数据块构成，则数据流需要事先通过一些分块算法进行分块，这可以由滑动指纹算法（Rabin）或同步算法（Rsync）等来完成。

[0027] 一个计时型布隆过滤器阵列包含 g（1 ＜ g ＜ N）个计时型布隆过滤器（TimingBloom Filter，TBF），在逻辑上构成一个循环先入先出队列，将逻辑上位于队首的 TBF 计为 TBF1，并依次标记每个 TBF 直到队尾的 TBFg。每个 TBF 包含一个位向量（Bit Vector，BV）和一个计时器组（Timer Array，TA），同时与其它 TBF 共享 k 个哈希函数，其中位向量和哈希函数组的工作原理与传统的布隆过滤器（Bloom Filter，BF）的工作原理相同，位向量用于记录插入的元素，而计时器数组用于记录元素插入时的时间戳。若滑动窗口

尺寸为N,则每个TBF的设计容量为N/(g-1),即可记录N/(g-1)个元素和其时间戳信息。规定新元素总是插入位 于队尾的TBFg,每当插入一个新元素则相应地从队首的TBF1中淘汰一个最旧的元素,当TBFg充满时TBF1则会被置空,然后队列循环移动一个单元,TBF1从队首转移到队尾用于记录新元素并被重新标记为TBFg。从而TBF1和TBFg所记录的有效元素总数保持为N/(g-1),考虑中间g-2个充满的TBF,则整个TBFA记录的有效元素总数为N。

[0028] 在物理数据组织上,TBFA把g个TBF的位向量与他们所关联的计时器组分离存储。具体来说,g个TBF具有同构性,即它们具有相同的位向量长度、计时器组长度且共享同一组哈希函数。g个位向量中具有相同偏移量的位单元被存放到连续的内存空间中,见图6所示,因此偏移量相同的位单元能够被同时访问,这种数据布局能够允许上层应用同时查询g个位向量,以判断待查询的元素是否被其中某个位向量所记录,其查询效率远高于传统的顺序检测方法。另一方面,每个位向量被关联一个长度相等的计时器组,用于存储被记录元素的时间戳。TBFA允许TBF2,...,TBFg-1对应的g-2个计时器组被卸载到硬盘,从而很大程度上减少内存开销。

[0029] TBFA的数据结构同时适用于单节点和分布式环境。见图5所示,在具有r个节点的分布式环境下,可将构成TBFA的g个TBF分为r组,每组s个TBF由一个节点存储和维护,其中r×s=g。在每个节点中,s个TBF仍然按照前段所述的方法优化内存数据布局以提高查询效率。此外,存储节点之间需要保持计时同步,当维护TBFg的节点插入一个新元素时,维护TBF1的节点将相应地删除一个旧元素,所有g个TBF仍然在逻辑上以循环先入先出队列的方式工作。

[0030] 计时型布隆过滤器阵列的重复数据检测原理。计时型布隆过滤器阵列(Timing Bloom FilterArray,TBFA)通过插入、删除和查询三种操作支持滑动窗口模型下的流式重复数据检测。具体说明如下:

[0031] 元素的插入方法。如前所述,TBFA所包含的所有TBF是同构的,新元素总是被插入到逻辑上位于队尾的TBFg。TBFg包含一个长度为m的位向量(布隆过滤器)和一个由m个计时器构成的计时器组(Timer Array,TA),同时与其它TBF共享一个基础时钟(Base Clock,BC)和一组哈希函数h1,h2,..,hk。基础时钟在[0,1,...,N/(g-1)-1]的范围内循环计时并以0为一个计时周期的开始。初始阶段,基础时钟和布隆过滤器的各个位都被置0;当插入一个新元素x时,TBFg的布隆过滤器中偏移量为h1(x),h2(x),...,hk(x)的k个位被置1,用以记录x;另一方面,TBFg的计时器组中偏移量为h1(x),h2(x),...,hk(x)的k个计时器被设置为基础时钟的当前值。完成上述操作后,基础时钟自增到下一个计时点。

[0032] 元素的删除方法。TBFA采用被动删除策略淘汰移出滑动窗口的旧元素。具 体来说,当新元素插入TBFg触发基础时钟自增操作后后,TBF1中时间戳小于基础时钟当前值的元素被认为已经失效。理论上在TBF1与TBF_g中元素的和为N/(g-1)个,随着TBFg满载,TBF1中的所有元素将会失效从而其数据结构会被重新初始化,同时基础时钟进入下一个循环计时周期,这时可以把TBF_1摘下来,插入到TBF_g的前面。这样形成一个循环队列。所有TBF在逻辑上前移动了一个单元,形成新的先入先出队列。这种被动删除元素的方法可以有效减少删除元素的内存访问频度,提高整个TBFA的查询效率。

[0033] 元素的查询方法。查询元素x的重复性时,首先通过哈希函数组计算x的k个映射值h1(x),h2(x),…,hk(x)。对于一个具体的TBF,当且仅当其布隆过滤器中偏移量

为 h1(x),h2(x),…,hk(x) 的 k 个位单元全为 1 时，认为 x 已经被该 TBF 所记录。由于 g 个 TBF 的布隆过滤器具有优化的内存数据布局，TBFA 可以同时查询 x 在所有 TBF 中的存在性，从而判断 x 是否是当前滑动窗口中的重复元素。具体做法见图 6 所示，取出偏移量分别为 h1(x),h2(x),…,hk(x) 的 k 处向量，即 {$BitVector_{h1(x)}$，$BitVector_{h2(x)}$，……，$BitVector_{hk(x)}$}，对这 k 个向量按位进行“与”运算，得到的结果，如果除最后一位外（最后位必为 1，它代表刚插入的元素），其它的某位为“1”则表示该位对应的布隆过滤中有个元素与刚插入的数据内容重复，即为查找到的重复元素；如果计算得到的结果所有位都为“0”则表示没有找到重复元素。

[0034] 特别地，当 x 出现在 TBF1 中时，需要进一步检测 x 的时间戳以判断其是否已经失效，并根据检测结果对最终判断作必要修正。由于 TBF2,…,TBFg-1 的计时器组在查询期间不被访问，他们可以被卸载到硬盘中以减少内存开销，当各 TBF 构成的先入先出队列循环时，再重新加载必要的计时器组。

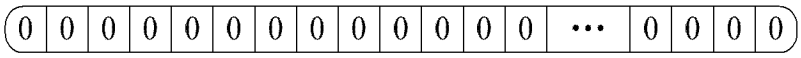

0 0 0 0 0 0 0 0 0 0 0 0 0 ··· 0 0 0 0

图 1

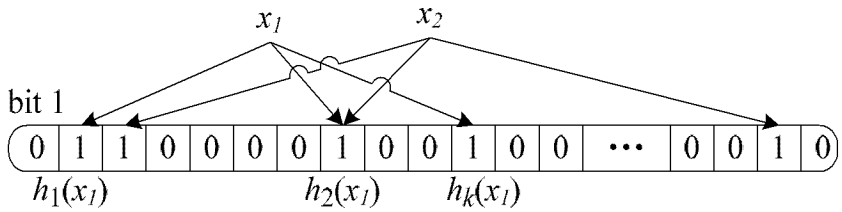

x_1
x_2
bit 1
0 1 1 0 0 0 0 1 0 0 1 0 0 ··· 0 0 1 0
$h_1(x_1)$
$h_2(x_1)$
$h_k(x_1)$

图 2

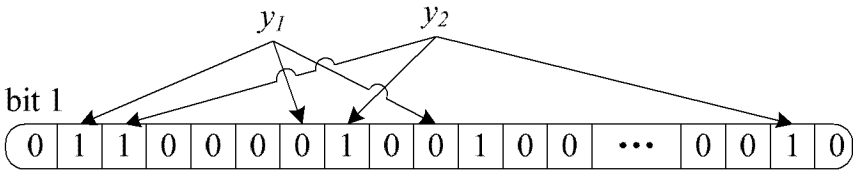

y_1
y_2
bit 1
0 1 1 0 0 0 0 1 0 0 1 0 0 ··· 0 0 1 0

图 3

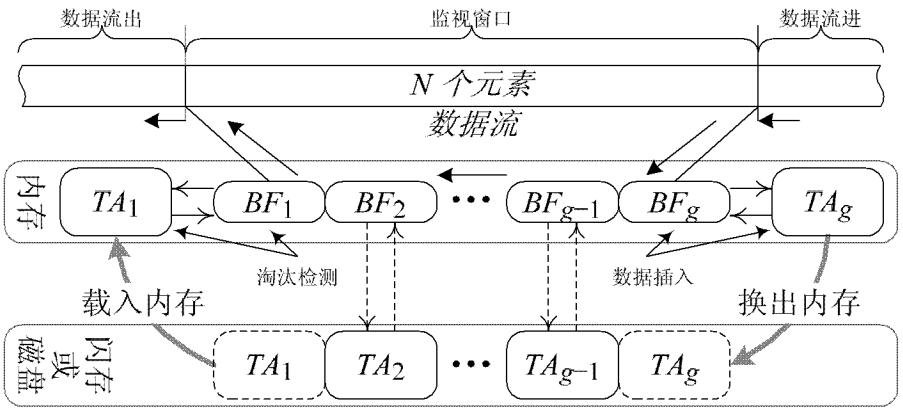

数据流出
监视窗口
数据流进
N 个元素
数据流
内存
TA_1
BF_1
BF_2
BF_{g-1}
BF_g
TA_g
淘汰检测
数据插入
载入内存
换出内存
闪存或磁盘
TA_1
TA_2
TA_{g-1}
TA_g

图 4

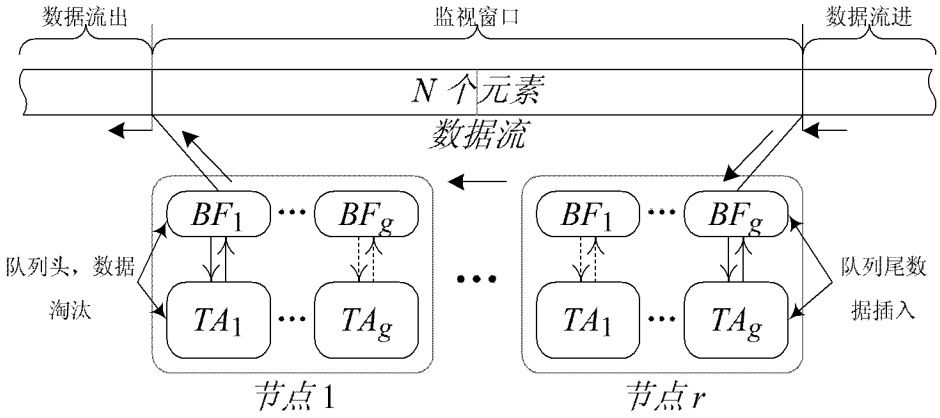

数据流出
监视窗口
数据流进
N 个元素
数据流
BF1
BFg
TA1
TAg
队列头，数据
淘汰
队列尾数
据插入
节点 1
节点 r

图 5

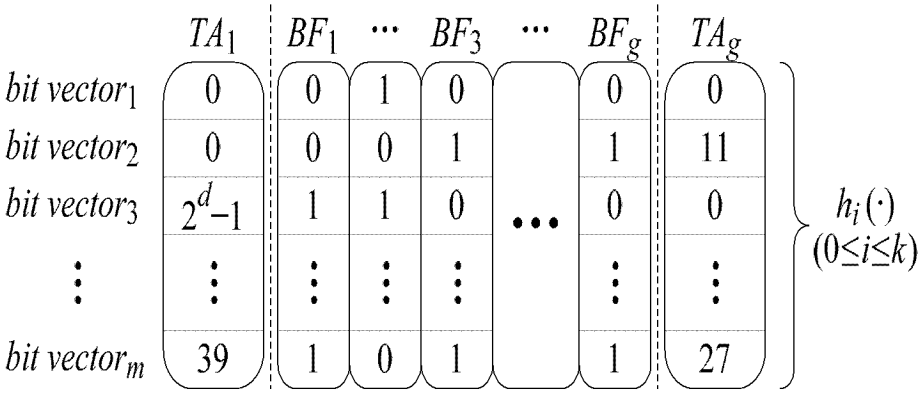

TA1
BF1
BF3
BFg
TAg
bit vector1
bit vector2
bit vector3
bit vectorm
0
0
1
0
0
0
0
0
0
1
11
2d−1
1
1
0
0
0
39
1
0
1
1
27
hi(·)
(0≤i≤k)

图 6