(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 102968467 A
(43)申请公布日 2013.03.13

(21)申请号 201210447165.4

(22)申请日 2012.11.10

(71)申请人 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 曹强 谢长生 黄国强 慎涵

(74)专利代理机构 华中科技大学专利中心 42201
代理人 朱仁玲

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

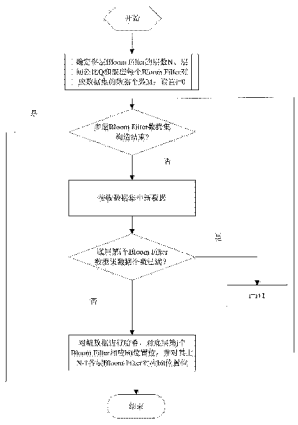
权利要求书 2 页 说明书 7 页 附图 3 页

(54)发明名称

一种多层Bloom Filter的优化方法及查询方法

(57)摘要

本发明公开了一种多层Bloom Filter的优化方法，包括以下步骤：根据总数据集的大小S确定Bloom Filter的层数N、第一层Bloom Filter个数Q以及底层每个Bloom Filter对应数据集的数据个数M，并设置计数器i=0，判断多层Bloom Filter的构造是否完毕，若未完毕，则接收新数据，判断底层第i个Bloom Filter对应数据集的数据个数是否等于M，若不等于，则使用哈希函数组对新数据进行哈希计算以得到哈希值，将底层Bloom Filter第i个Bloom Filter中与哈希值对应的比特位置1，并对底层Bloom Filter上面的各N-1层进行置位。本发明的方法使比特位查询操作更为简便，大大减少了磁盘访问的次数，有效减少了对多层Bloom Filter的查询时间。



CN 102968467 A

1. 一种多层 Bloom Filter 的优化方法，其特征在于，包括以下步骤：

(1) 根据总数据集的大小 S 确定 Bloom Filter 的层数 N、第一层 BloomFilter 个数 Q 以及底层每个Bloom Filter对应数据集的数据个数M，并设置计数器i=0；其中，多层Bloom Filter 各层 Bloom Filter 个数是以第一层 BloomFilter 个数 Q 为首项，公比同为 Q 的等比数列，且满足 $Q^N \times M \geq S$，Q 是磁盘扇区容量的整数倍，各层 Bloom Filter 包含的比特位总数相等；

(2) 判断多层 Bloom Filter 的构造是否完毕，若完毕则过程结束，否则进入步骤(3)；

(3) 接收新数据；

(4)判断底层第 i 个 Bloom Filter 对应数据集的数据个数是否等于 M，若等于，则进入步骤(5)，否则进入步骤(6)；

(5) 设置 i=i+1；

(6) 使用哈希函数组对新数据进行哈希计算以得到哈希值，将底层 Bloom Filter 第 i 个Bloom Filter 中与哈希值对应的比特位置 1，并对底层 Bloom Filter 上面的各 N-1 层进行置位，然后返回步骤(2)。

2. 根据权利要求 1 所述的优化方法，其特征在于，步骤(6) 中对底层 Bloom Filter 上面的各 N-1 层进行置位的操作包括：

(a)在第 N 层的第 i 个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间，将该连续空间中属于第 i 个 Bloom Filter 的比特位置 1；

(b) 第 N 层第 i 个 Bloom Filter 对应于第 N-1 层的第 $\left\lceil \frac{i}{Q} \right\rceil$ 个 Bloom Filter，在第 N-1 层第 $\left\lceil \frac{i}{Q} \right\rceil$ 个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间，将该连续空间中属于第 $\left\lceil \frac{i}{Q} \right\rceil$ 个 Bloom Filter 的比特位置 1，$\lceil\ \rceil$ 表示向上取整；

(c) 第 N-1 层的第 $\left\lceil \frac{i}{Q} \right\rceil$ 个 Bloom Filter 对应于第 N-2 层的第 $\left\lceil \frac{i}{Q^2} \right\rceil$ 个 Bloom Filter，在第 N-2 层该第 $\left\lceil \frac{i}{Q^2} \right\rceil$ 个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间，将该连续空间中属于第 $\left\lceil \frac{i}{Q^2} \right\rceil$ 个 Bloom Filter 的比特位置 1；

(d) 重复(a) 至(c)，直到将第一层 Bloom Filter 相应连续空间的相应比特位置 1 为止。

3. 一种多层 Bloom Filter 的查询方法，其特征在于，包括以下步骤：

(1) 根据总数据集的大小 S 确定 Bloom Filter 的层数 N、第一层 BloomFilter 个数 Q 以及底层每个 Bloom Filter 对应数据集的数据个数 M，并设置计数器 i=0；

(2) 判断多层 Bloom Filter 的构造是否完毕，若完毕则进入步骤(7)，否则进入步骤(3)；

(3) 接收新数据；

(4)判断底层第 i 个 Bloom Filter 对应数据集的数据个数是否等于 M,若等于,则进入步骤(5),否则进入步骤(6);

(5)设置 i=i+1;

(6)使用哈希函数组对新数据进行哈希计算以得到哈希值,将底层 Bloom Filter 第 i 个 Bloom Filter 中与哈希值对应的比特位置 1,并对底层 Bloom Filter 上面的各 N-1 层进行置位,然后返回步骤(2);

(7)初始化计数器 j = 1;

(8)使用与步骤(6)中相同的哈希函数组对待查询数据进行哈希运算以得到哈希值;

(9)从第一层的 Q 个 Bloom Filter 所对应的所有连续地址空间中选取与步骤(8)所得的哈希值对应的连续地址空间,对这些连续地址空间做按位相与运算,进入步骤(10);

(10)判断该与运算结果中的比特位是否全为 0,若是,说明待查询数据不存在,过程结束,否则进入步骤(11);

(11)判断 j 是否等于层数 N,若等于,进入步骤(14),否则进入步骤(12);

(12)对于每组与运算结果中值为 1 的每一个比特位,选取其所属的 Bloom Filter 对应的第 j+1 层的 Q 个 Bloom Filter 组成一组查询 Bloom Filter,置 j = j+1;

(13)对于第 j 层的每一组查询 Bloom Filter,从该组查询 Bloom Filter 对应的所有连续地址空间中选取与步骤(8)所得的哈希值对应的连续地址空间,对这些连续地址空间做按位相与运算,然后返回步骤(11);

(14)在各组与运算结果中值为 1 的比特位所属的 Bloom Filter 对应的数据集中查询数据,过程结束。

4. 根据权利要求 3 所述的查询方法,其特征在于,步骤(6)中对底层 Bloom Filter 上面的各 N-1 层进行置位的操作包括:

(a)在第 N 层的第 i 个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间,将该连续空间中属于第 i 个 Bloom Filter 的比特位置 1;

(b)第 N 层第 i 个 Bloom Filter 对应于第 N-1 层的第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter,在第 N-1 层第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间,将该连续空间中属于第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 的比特位置 1,$\lceil \rceil$表示向上取整;

(c)第 N-1 层的第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 对应于第 N-2 层的第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter,在第 N-2 层该第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间,将该连续空间中属于第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter 的比特位置 1;

(d)重复(a)至(c),直到将第一层 Bloom Filter 相应连续空间的相应比特位置 1 为止。

一种多层 Bloom Filter 的优化方法及查询方法

技术领域

[0001] 本发明属于计算机存储领域，更具体地，涉及一种多层 Bloom Filter 的优化方法及查询方法。

背景技术

[0002] Bloom filter 是由 Howard Bloom 在 1970 年提出的二进制向量数据结构，可用来快速判定一个元素是否存在于一个集合中。相较于散列、树等方法，Bloom Filter 可保证待查询数据集在存放时的空间局部性。随着待查询数据集的增长，数据集可被分割为若干个相同小容量的数据集，分别对应一个 Bloom Filter。由于被查询数据要依次查询每个 Bloom Filter，直至找到该数据或者查询结束，多个 Bloom Filter 的查询时间大大增加。为了加速海量数据集的查询过程，多层 Bloom Filter 被引进。当上层 Bloom Filter 的判定元素不存在，其对应下层的 Bloom Filter 即可不再查询，减少了 BloomFilter 查询次数。

[0003] 图 2(a)为三层 Bloom Filter 的结构组织，每层 Bloom Filter 包含的二进制比特位总数相等。第 i 层(1 ≤ i<3，i 为正整数)每个 Bloom Filter 对应于 i+1 层的 2 个 Bloom Filter。

[0004] 对一个哈希值进行查询时，先分别判定其在第一层的各 Bloom Filter 对应的比特位是否为 1，若为 1，查询命中，则该 Bloom Filter 对应的下层 BloomFilter 要继续查询。如图 1 所述，第一层 2 个 Bloom Filter 对应比特位值均为 1，则需要查询第一层这 2 个 Bloom Filter 在第二层中对应的所有 BloomFilter。对于没有命中的 Bloom Filter，该哈希值不存在于其对应的数据集中，其对应于下层的 Bloom Filter 不用继续查询。

[0005] 查询第二层中相应的 Bloom Filter，若要查询 Bloom Filter 的比特位值为 1，查询命中，则该 Bloom Filter 对应的下层 Bloom Filter 要继续查询。如图 1，第二层中第 2 个 Bloom Filter 命中，则要继续查询该 Bloom Filter 对应的第三层中的 Bloom Filter。对于没有命中的 Bloom Filter，其对应于下层的 Bloom Filter 不用继续查询。图 1 中第 1、第 3 和第 4 个 Bloom Filter 对应于第三层的 Bloom Filter 即不用查询。

[0006] 在底层 Bloom Filter 查询中，当要查询 Bloom Filter 相应的比特位值为 1，命中，则表示该哈希值可能存在于该 Bloom Filter 对应的数据集中，取该数据集进行查询。如图 1 所示，第三层第 3 个 Bloom Filter 命中，即取其对应的数据集查询该哈希值是否存在。对于没有命中的底层 Bloom Filter，其对于数据集不用查询。图 1 中除了第三层第 3 个 Bloom Filter 对应的数据集外，其他数据集均不用被查询。

[0007] 多层 Bloom Filter 将被查询哈希值定位到不同的数据集，大大减少了数据的查询的次数，减少查询开销。

[0008] 然而，对于海量数据集，对多层 Bloom Filter 查询次数会很大，BloomFilter 的查询变成一个瓶颈，甚至当 Bloom Filter 规模超过内存容量时，会产生大量的磁盘访问(Input/Output，简称 IO)，这直接造成元素查询的时间超过我们可承受范围。

发明内容

[0009] 针对现有技术的缺陷，本发明的目的在于提供一种多层 Bloom Filter 的优化方法，其能够加快元素的查询过程。

[0010] 为实现上述目的，本发明提供了一种多层 Bloom Filter 的优化方法，包括以下步骤：

[0011] (1)根据总数据集的大小 S 确定 Bloom Filter 的层数 N、第一层 BloomFilter 个数 Q 以及底层每个 Bloom Filter 对应数据集的数据个数 M，并设置计数器 i=0；其中，多层 Bloom Filter 各层 Bloom Filter 个数是以第一层 BloomFilter 个数 Q 为首项，公比同为 Q 的等比数列，且满足 $Q^N \times M \geq S$，Q 是磁盘扇区容量的整数倍，各层 Bloom Filter 包含的比特位总数相等；

[0012] (2)判断多层 Bloom Filter 的构造是否完毕，若完毕则过程结束，否则进入步骤(3)；

[0013] (3)接收新数据；

[0014] (4)判断底层第 i 个 Bloom Filter 对应数据集的数据个数是否等于 M，若等于，则进入步骤(5)，否则进入步骤(6)；

[0015] (5)设置 i=i+1；

[0016] (6)使用哈希函数组对新数据进行哈希计算以得到哈希值，将底层 Bloom Filter 第 i 个 Bloom Filter 中与哈希值对应的比特位置 1，并对底层 Bloom Filter 上面的各 N-1 层进行置位，然后返回步骤(2)。

[0017] 步骤(6)中对底层 Bloom Filter 上面的各 N-1 层进行置位的操作包括：

[0018] (a)在第 N 层的第 i 个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间，将该连续空间中属于第 i 个 Bloom Filter 的比特位置 1；

[0019] (b)第 N 层第 i 个 Bloom Filter 对应于第 N-1 层的第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter，在第 N-1 层第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间，将该连续空间中属于第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 的比特位置 1，$\lceil \ \rceil$表示向上取整；

[0020] (c)第 N-1 层的第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 对应于第 N-2 层的第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter，在第 N-2 层该第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间，将该连续空间中属于第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter 的比特位置 1；

[0021] (d)重复(a)至(c)，直到将第一层 Bloom Filter 相应连续空间的相应比特位置 1 为止。

[0022] 通过本发明所构思的以上技术方案，与现有技术相比，本发明具有以下的有益效果：

[0023] 由于采用了比特位组织方式以及步骤(1)至(6)的优化方法，多层 Bloom Filter

在不增加存储空间的基础上，相应比特位查询操作更为简便，大大减少了磁盘访问的次数，有效减少了对多层 Bloom Filter 的查询时间。

[0024] 本发明的另一目的在于提供一种多层Bloom Filter的查询方法，其能够加快元素的查询过程。

[0025] 为实现上述目的，本发明提供了一种多层 Bloom Filter 的查询方法，包括以下步骤：

[0026] (1)根据总数据集的大小 S 确定 Bloom Filter 的层数 N、第一层 BloomFilter 个数 Q 以及底层每个 Bloom Filter 对应数据集的数据个数 M，并设置计数器 i=0；

[0027] (2)判断多层 Bloom Filter 的构造是否完毕，若完毕则进入步骤(7)，否则进入步骤(3)；

[0028] (3)接收新数据；

[0029] (4)判断底层第 i 个 Bloom Filter 对应数据集的数据个数是否等于 M，若等于，则进入步骤(5)，否则进入步骤(6)；

[0030] (5)设置 i=i+1；

[0031] (6)使用哈希函数组对新数据进行哈希计算以得到哈希值，将底层 Bloom Filter 第 i 个 Bloom Filter 中与哈希值对应的比特位置 1，并对底层 Bloom Filter 上面的各 N-1 层进行置位，然后返回步骤(2)；

[0032] (7)初始化计数器 j = 1；

[0033] (8)使用与步骤(6)中相同的哈希函数组对待查询数据进行哈希运算以得到哈希值；

[0034] (9)从第一层的 Q 个 Bloom Filter 所对应的所有连续地址空间中选取与步骤(8)所得的哈希值对应的连续地址空间，对这些连续地址空间做按位相与运算，进入步骤(10)；

[0035] (10)判断该与运算结果中的比特位是否全为 0，若是，说明待查询数据不存在，过程结束，否则进入步骤(11)；

[0036] (11)判断 j 是否等于层数 N，若等于，进入步骤(14)，否则进入步骤(12)；

[0037] (12)对于每组与运算结果中值为 1 的每一个比特位，选取其所属的 Bloom Filter 对应的第 j+1 层的 Q 个 Bloom Filter 组成一组查询 Bloom Filter，置 j = j+1；

[0038] (13)对于第 j 层的每一组查询 Bloom Filter，从该组查询 Bloom Filter 对应的所有连续地址空间中选取与步骤(8)所得的哈希值对应的连续地址空间，对这些连续地址空间做按位相与运算，然后返回步骤(11)；

[0039] (14)在各组与运算结果中值为 1 的比特位所属的 Bloom Filter 对应的数据集中查询数据，过程结束。

[0040] 步骤(6)中对底层 Bloom Filter 上面的各 N-1 层进行置位的操作包括：

[0041] (a)在第 N 层的第 i 个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间，将该连续空间中属于第 i 个 Bloom Filter 的比特位置 1；

[0042] (b)第 N 层第 i 个 Bloom Filter 对应于第 N-1 层的第 $\left\lceil \frac{i}{Q} \right\rceil$ 个 Bloom Filter，在第 N-1 层第 $\left\lceil \frac{i}{Q} \right\rceil$ 个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间，将该

连续空间中属于第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 的比特位置 1，$\lceil \rceil$表示向上取整；

[0043] (c)第 N-1 层的第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 对应于第 N-2 层的第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter，在第 N-2 层该第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间，将该连续空间中属于第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter 的比特位置 1；

[0044] (d)重复(a)至(c)，直到将第一层 Bloom Filter 相应连续空间的相应比特位置 1 为止。

[0045] 通过本发明所构思的以上技术方案，与现有技术相比，本发明具有以下的有益效果：由于采用了比特位组织方式以及步骤(1)至(14)的查询方法，一个哈希值对应同层 Q 个 Bloom Filter 的比特位存在于同一连续地址空间，通过对少数连续空间的查询即可实现对进行多层 Bloom Filter 的查询，显著地加快了查询速度。

附图说明

[0046] 图 1 是现有技术中多层 Bloom Filter 的组织结构图。

[0047] 图 2（a）是现有技术中 Bloom Filter 的比特位组织方式。

[0048] 图 2（b）是本发明 Bloom Filter 的比特位组织方式。

[0049] 图 3 是本发明多层 Bloom Filter 的优化方法的流程图。

[0050] 图 4 是本发明多层 Bloom Filter 的查询方法的流程图。

[0051] 图 5 是本发明多层 Bloom Filter 的优化方法的实例图。

具体实施方式

[0052] 为了使本发明的目的、技术方案及优点更加清楚明白，以下结合附图及实施例，对本发明进行进一步详细说明。应当理解，此处所描述的具体实施例仅仅用以解释本发明，并不用于限定本发明。

[0053] 图 2（a）给出现有 Bloom Filter 的比特位组织方式，在现有的多层 BloomFilter 中，上层一个 Bloom Filter 对应下层 W 个 Bloom Filter（W 为人为设定的正整数），每个 Bloom Filter 中所有的比特位在物理地址空间中是连续的。

[0054] 图 2（b）为本发明 Bloom Filter 的比特位组织方式，本发明多层 BloomFilter 的构建中，第一层 Q 个 Bloom Filter 的相同位置的比特位放在磁盘同一连续地址空间中，第 j（j ＝ 1,…,N-1）层的第 m 个 Bloom Filter 对应于第 j+1 层的 Q 个 Bloom Filter，所有 Q 个 Bloom Filter 相同位置的比特位放在磁盘同一连续地址空间中，第 j 层的第 m 个 Bloom Filter 的比特位数与其对应的第 j+1 层的 Q 个 Bloom Filter 的比特位总数相等。连续地址空间大小即为 Q bit，第 k 个连续地址空间中第 m（1 ≤ m ≤ Q）个 bit 属于相应第 m 个 Bloom Filter，其值即为相应第 m 个 Bloom Filter 的第 k 个比特位的值；对于关联的 Q 个 Bloom Filter，一个哈希值对应于一个连续空间，其中 Q、j、k、m、N 均为正整数。

[0055] 如图 3 所示，本发明多层 Bloom Filter 的优化方法包括以下步骤：

[0056] (1)根据总数据集的大小 S 确定 Bloom Filter 的层数 N、第一层 BloomFilter 个数 Q 以及底层每个 Bloom Filter 对应数据集的数据个数 M,并设置计数器 i=0 ;其中,多层 Bloom Filter 各层 Bloom Filter 个数是以第一层 BloomFilter 个数 Q 为首项,公比同为 Q 的等比数列,且满足 $Q^N \times M \geq S$, Q 是磁盘扇区容量的整数倍,各层 Bloom Filter 包含的比特位总数相等 ;

[0057] (2)判断多层 Bloom Filter 的构造是否完毕,即当前的多层 Bloom Filter 已是否包含总数据集中所有数据,若完毕则过程结束,否则进入步骤(3);

[0058] (3)接收新数据 ;

[0059] (4)判断底层第 i 个 Bloom Filter 对应数据集的数据个数是否等于 M,若等于,则进入步骤(5),否则进入步骤(6);

[0060] (5)设置 i=i+1 ;

[0061] (6)使用哈希函数组对新数据进行哈希计算以得到哈希值,将底层 Bloom Filter 第 i 个 Bloom Filter 中与哈希值对应的比特位置 1,并对底层 Bloom Filter 上面的各 N-1 层进行置位,然后返回步骤(2);

[0062] 其中,对底层 Bloom Filter 上面的各 N-1 层进行置位的操作包括 :

[0063] (a)在底层即第 N 层的第 i 个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间,将该连续空间中属于第 i 个 Bloom Filter 的比特位置 1 ;

[0064] (b)第 N 层第 i 个 Bloom Filter 对应于第 N-1 层的第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter,在第 N-1 层第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间,将该连续空间中属于第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 的比特位置 1,$\lceil \ \rceil$表示向上取整 ;

[0065] (c)第 N-1 层的第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 对应于第 N-2 层的第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter,在第 N-2 层该第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间,将该连续空间中属于第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter 的比特位置 1 ;

[0066] (d)重复(a)至(c),直到将第一层 Bloom Filter 相应连续空间的相应比特位置 1 为止。

[0067] 如图 4 所示,本发明多层 Bloom Filter 的查询方法包括以下步骤 :

[0068] (1)根据总数据集的大小 S 确定 Bloom Filter 的层数 N、第一层 BloomFilter 个数 Q 以及底层每个 Bloom Filter 对应数据集的数据个数 M,并设置计数器 i=0 ;

[0069] (2)判断多层 Bloom Filter 的构造是否完毕,即当前的多层 Bloom Filter 已是否包含总数据集中所有数据,若完毕则进入步骤(7),否则进入步骤(3);

[0070] (3)接收新数据 ;

[0071] (4)判断底层第 i 个 Bloom Filter 对应数据集的数据个数是否等于 M,若等于,则进入步骤(5),否则进入步骤(6);

[0072] (5)设置 i=i+1；

[0073] (6)使用哈希函数组对新数据进行哈希计算以得到哈希值，将底层 Bloom Filter 第 i 个 Bloom Filter 中与哈希值对应的比特位置 1，并对底层 Bloom Filter 上面的各 N-1 层进行置位，然后返回步骤(2)；

[0074] 其中，对底层 Bloom Filter 上面的各 N-1 层进行置位的操作包括：

[0075] (a)在底层即第 N 层的第 i 个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间，将该连续空间中属于第 i 个 Bloom Filter 的比特位置 1；

[0076] (b)第 N 层第 i 个 Bloom Filter 对应于第 N-1 层的第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter，在第 N-1 层第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间，将该连续空间中属于第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 的比特位置 1，$\lceil \ \rceil$表示向上取整；

[0077] (c)第 N-1 层的第$\left\lceil \frac{i}{Q} \right\rceil$个 Bloom Filter 对应于第 N-2 层的第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter，在第 N-2 层该第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter 对应的所有连续空间中选取与哈希值对应的连续空间，将该连续空间中属于第$\left\lceil \frac{i}{Q^2} \right\rceil$个 Bloom Filter 的比特位置 1；

[0078] (d)重复(a)至(c)，直到将第一层 Bloom Filter 相应连续空间的相应比特位置 1 为止。

[0079] (7)初始化计数器 j = 1；

[0080] (8)使用与步骤(6)中相同的哈希函数组对待查询数据进行哈希运算以得到哈希值；

[0081] (9)从第一层的 Q 个 Bloom Filter 所对应的所有连续地址空间中选取与步骤(8)所得的哈希值对应的连续地址空间，对这些连续地址空间做按位相与运算，进入步骤(10)；

[0082] (10)判断该与运算结果中的比特位是否全为 0，若是，说明待查询数据不存在，过程结束，否则进入步骤(11)；

[0083] (11)判断 j 是否等于层数 N，若等于，进入步骤(14)，否则进入步骤(12)；

[0084] (12)对于每组与运算结果中值为 1 的每一个比特位，选取其所属的 Bloom Filter 对应的第 j+1 层的 Q 个 Bloom Filter 组成一组查询 Bloom Filter，置 j = j+1；

[0085] (13)对于第 j 层的每一组查询 Bloom Filter，从该组查询 Bloom Filter 对应的所有连续地址空间中选取与步骤(8)所得的哈希值对应的连续地址空间，对这些连续地址空间做按位相与运算，然后返回步骤(11)；

[0086] (14)在各组与运算结果中值为 1 的比特位所属的 Bloom Filter 对应的数据集中查询数据，过程结束。实例：

[0087] 对于存储容量为 512TB 的海量重复数据删除系统，假设它基于块级重删，块大小为 4KB，每个块对应一个指纹，指纹个数共有 2^{37} 个，每个指纹 20 个字节，加上其他元数据信息，一个指纹项需要 32 个字节，共有 4TB 大小的指纹库；其在内存中是放不下的；当一个新

的数据块到来时，需要判定它是否和已存储的数据重复，即该数据块指纹和已有的指纹是否相同；

[0088] 为了加快指纹查找过程，本发明引入了多层 Bloom Filter，假定 BloomFilter 的错误率为万分之一，取 10 个哈希函数，相应每层 Bloom Filter 大小高达为 320GB，两层即为 640GB，其在内存中也是放不下的，需要放在磁盘中，其查询即会造成磁盘访问；

[0089] 根据公式 $Q^N \times M \geq S$，建立两层 Bloom Filter，第一层有 2^{15} 个 BloomFilter，因为公比为 2^{15}，第二层有 2^{30} 个 Bloom Filter，第二层即底层每个 Bloom Filter 对应 2^7 个指纹，即 $Q=2^{15}$，$N=2$，$M = 2^7$，$S=2^{37}$ 个，满足公式；

[0090] 按照本发明的 Bloom Filter 构造方式，连续地址空间大小为 2^{15}bit 即 4KB；

[0091] 如图 5，假定新指纹经过 10 个哈希函数得到 3 个不同的哈希值 1，2，10。

[0092] 三个哈希值对应于第一层 Bloom Filter 中第 1、第 2、第 10 个连续地址空间，我们取这 3 个相应的 4KB 连续地址空间，做与运算。

[0093] 三个连续空间中第 1 个 bit 分别为 1、1、0，相与结果为 0；第 2 个 bit 分别为 0、0、0，相与结果为 0；第 3 个比特位分别为 1、1、1，相与结果为 1；其他各位相与结果均为 0。

[0094] 连续空间与结果中的第 3 个 bit 属于第一层第 3 个 Bloom Filter，值为 1 表示其所属 Bloom Filter 查询命中，因为 Bloom Filter 为 2 层，需要查询该 Bloom Filter 对应的下一层即第二层的 2^{15} 个 Bloom Filter。

[0095] 根据哈希值，取第二层相应 Bloom Filter 的第 1、第 2、第 10 个连续地址空间，取这 3 个相应的 4KB 连续地址空间，将相应的空间做与运算，三个连续空间中第 1 个 bit 分别为 1、1、0，相与结果为 0；第 1 个 bit 分别为 1、1、1，相与结果为 1；其他各位相与结果均为 0。

[0096] 该层已经是最后一层 Bloom Filter，读取该命中 Bloom Filter 对应的数据集，即第二层第 $2\times2^{15}+2$ 个 Bloom Filter 所对应的数据集。

[0097] 若该多层 Bloom Filter 的比特位全部存储在磁盘，该查询访问磁盘的总数即为 6 次；

[0098] 如果按照传统方式，要在第一层 2^{15} 个 Bloom Filter 中，每个 Bloom Filter 相应的 3 个比特位进行查询，这样做了 3×2^{15} 个比特位查询和至少 2^{15} 个 512 字节数据的磁盘访问，第二层做了同样 3×2^{15} 个比特位查询和至少 2^{15} 个 512 字节数据的磁盘访问，共有至少 2^{16} 次的磁盘访问；

[0099] 现有 Bloom Filter 磁盘访问次数为优化后磁盘的访问次数约 2^{13} 倍。

[0100] 本领域的技术人员容易理解，以上所述仅为本发明的较佳实施例而已，并不用以限制本发明，凡在本发明的精神和原则之内所作的任何修改、等同替换和改进等，均应包含在本发明的保护范围之内。

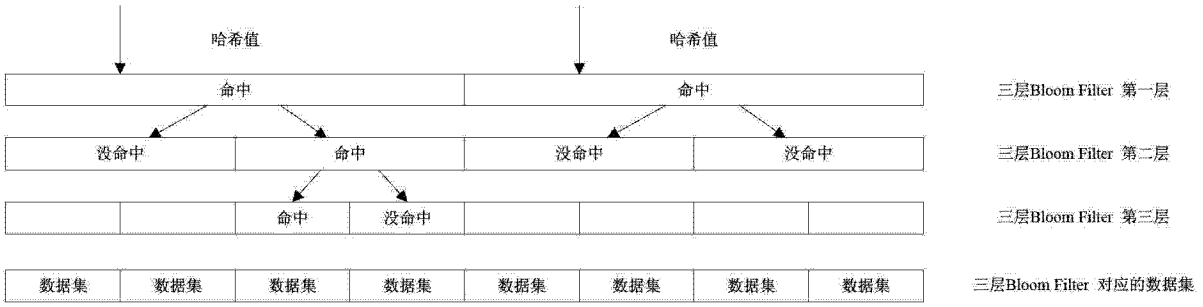

哈希值
哈希值
命中
命中
三层Bloom Filter 第一层
没命中
命中
没命中
没命中
三层Bloom Filter 第二层
命中
没命中
三层Bloom Filter 第三层
数据集
数据集
数据集
数据集
数据集
数据集
数据集
数据集
三层Bloom Filter 对应的数据集

图 1

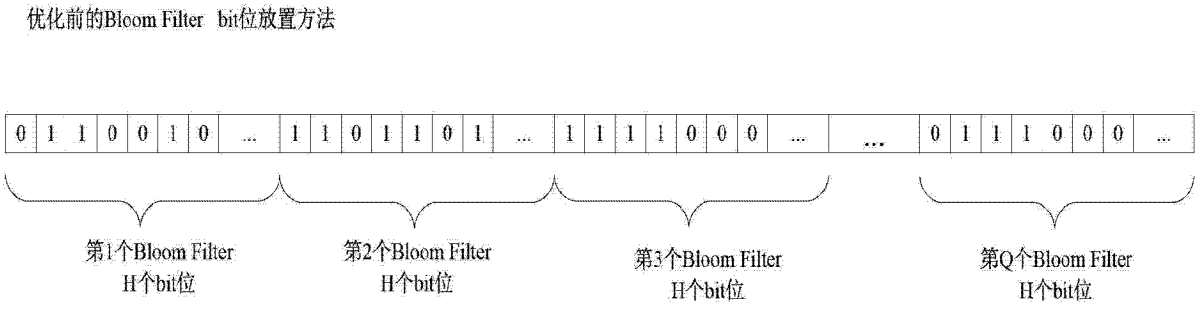

优化前的Bloom Filter bit位放置方法
第1个Bloom Filter
H个bit位
第2个Bloom Filter
H个bit位
第3个Bloom Filter
H个bit位
第Q个Bloom Filter
H个bit位

图 2(a)

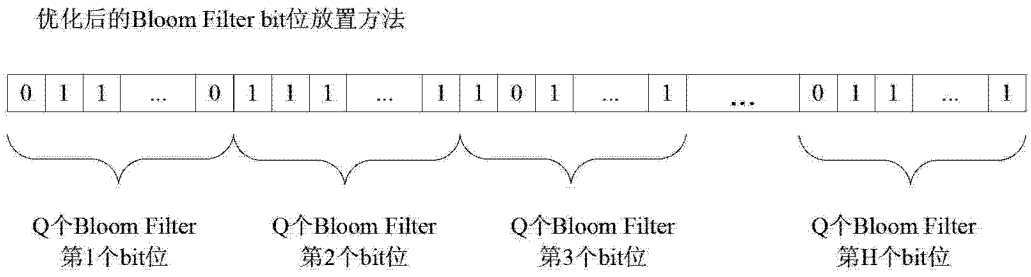

优化后的Bloom Filter bit位放置方法
Q个Bloom Filter
第1个bit位
Q个Bloom Filter
第2个bit位
Q个Bloom Filter
第3个bit位
Q个Bloom Filter
第H个bit位

图 2(b)

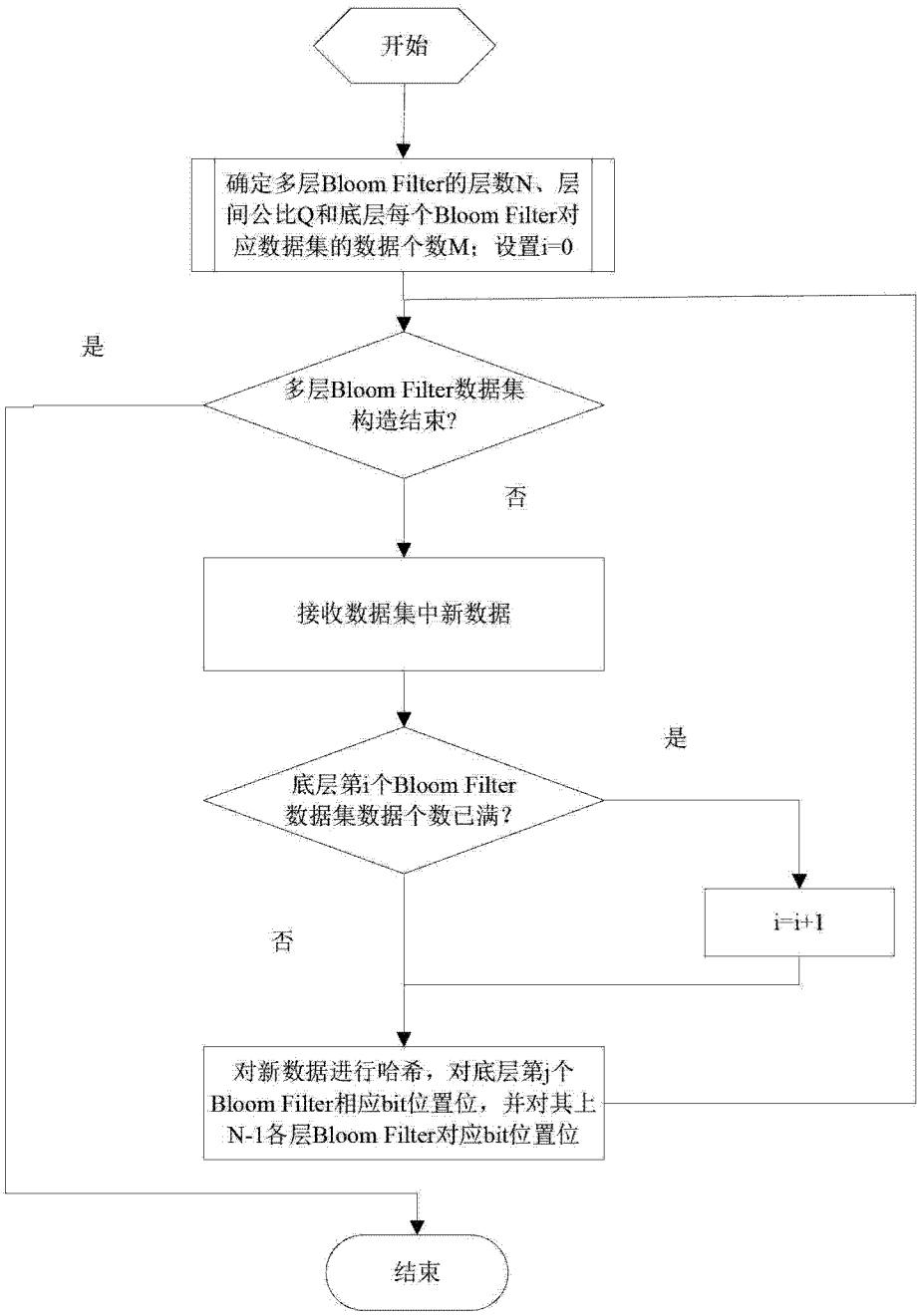

开始
确定多层Bloom Filter的层数N、层间公比Q和底层每个Bloom Filter对应数据集的数据个数M；设置i=0
是
多层Bloom Filter数据集构造结束?
否
接收数据集中新数据
是
底层第i个Bloom Filter数据集数据个数已满？
i=i+1
否
对新数据进行哈希，对底层第j个Bloom Filter相应bit位置位，并对其上N-1各层Bloom Filter对应bit位置位
结束

图 3

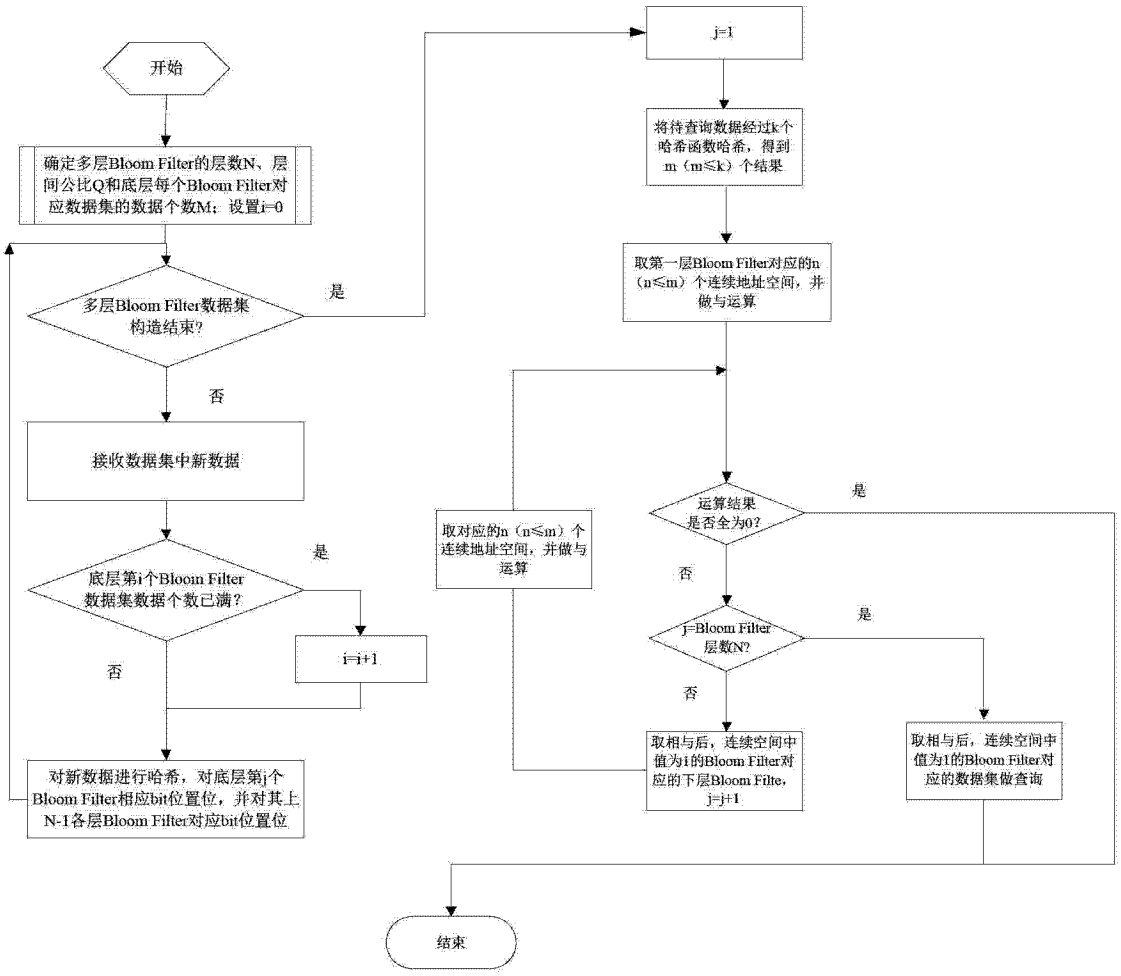

开始
确定多层Bloom Filter的层数N、层间公比Q和底层每个Bloom Filter对应数据集的数据个数M；设置i=0
多层Bloom Filter数据集构造结束?
是
否
接收数据集中新数据
底层第i个Bloom Filter数据集数据个数已满?
是
i=i+1
否
对新数据进行哈希，对底层第j个Bloom Filter相应bit位置位，并对其上N-1各层Bloom Filter对应bit位置位
j=1
将待查询数据经过k个哈希函数哈希，得到m（m≤k）个结果
取第一层Bloom Filter对应的n（n≤m）个连续地址空间，并做与运算
运算结果是否全为0?
是
否
j=Bloom Filter层数N?
是
否
取对应的n（n≤m）个连续地址空间，并做与运算
取相与后，连续空间中值为1的Bloom Filter对应的下层Bloom Filte，j=j+1
取相与后，连续空间中值为1的Bloom Filter对应的数据集做查询
结束

图 4

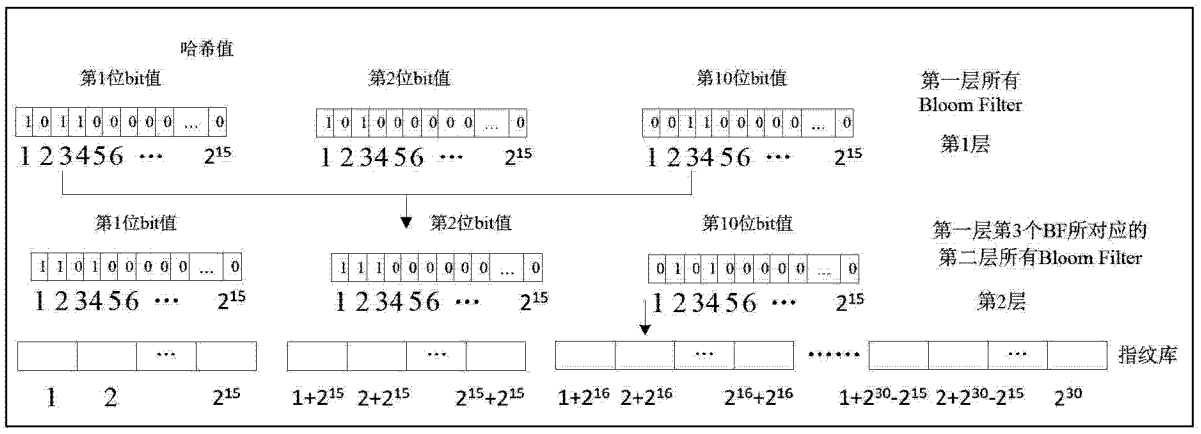

哈希值
第1位bit值
第2位bit值
第10位bit值
第一层所有Bloom Filter
第1层
第1位bit值
第2位bit值
第10位bit值
第一层第3个BF所对应的第二层所有Bloom Filter
第2层
指纹库

图 5