一种改进的 Dynamic Count Filter 实现方法*

岳未然 赵 辉 徐 龙

(四川大学计算机学院 成都 610064)

摘要:布隆过滤器常用来快速判断给定元素是否在一个集合中,动态计数过滤器是布隆过滤器的一种改进。本文针对当前动态计数过滤器处理数据溢出时,新建以及重建溢出过滤器向量时间开销大的问题,提出了一种基于布隆过滤器向量的改进实现方法。该方法采用多个布隆过滤器向量替代溢出过滤器向量,以避免溢出过滤器的建立,同时也避免了其重建时进行的数据拷贝。实验结果表明,该方法较动态计数过滤器和动态计数布隆过滤器缩减了处理数据溢出所需的时间,大大提升过滤器操作效率,并且较动态计数布隆过滤器节省了内存空间。

关键词:计数器,布隆过滤器,计数布隆过滤器,动态计数过滤器,动态计数布隆过滤器,布隆过滤器向量,溢出过滤器向量,多维动态计数过滤器

An Improved Dynamic Count Filter Realization Method

YUE Weiran, ZHAO Hui, XU Long

(Department of Computer, Sichuan University, Chengdu, 610064, China)

Abstract: Bloom filters are often used to charge if a given element is in a set quickly, and the dynamic count filter is an improvement over a Bloom filter. In this paper, an improved method based on Bloom Filter Vector is proposed to deal with the problem that the time cost of new and reconstructed Overflow Filter Vector is large when the data overflow is processed by Dynamic Count Filter. The method replaces the Overflow Filter Vector with multiple Bloom Filter Vectors to avoid the Overflow Filter and avoid the duplication of the old data. The experimental results show that this method is better than dynamic count filter and dynamic count bloom filter on reducing the processing time required for data overflow, enhancing the filter operation efficiency greatly, and also save more memory space than the dynamic count bloom filter.

Keywords: Counter, Bloom Filter, Counter Bloom Filter, Dynamic Count Filter, Bloom Filter Vector, Overflow Filter Vector, Dimension Dynamic Count Filter

1 概述

布隆过滤器(Bloom Filter, BF)[1,2]以其提出者 Burton Howard Bloom 命名。它由一个布隆过滤器向量(Bloom Filter Vector, BFV)和一系列哈希函数组成,可以用来判断一个元素是否在一个集合中。由于使用了哈希函数,它的时间效率要远高于其他一些查询算法,适合应用在能容忍低错误率的场合。BF 被广泛应用于分布式数据库查询[3]、P2P 网络[4,5]、网络高速缓存查询[6,7]等等。

近几年,BF 的研究又取得了新的进展,文献[8]对其进行综述,主要有计数布隆过滤器(Counter Bloom Filter, CBF),动态计数过滤器(Dynamic Count Filter, DCF)。文献[9]将 DCF 应用于网络流抽样。文献[10]

本文于 2017 - 01 - 18 收到,2017 - 03 - 23 收到修改稿。

* 国家重点研发计划(2016yfb0800604,2016yfb0800605),国家自然科学基金项目(61572334)。

提出动态计数布隆过滤器(Dynamic Count Bloom Filter,DCBF)。CBF 由一个计数布隆过滤器向量(Count Bloom Filter Vector,CBFV)和一系列哈希函数组成,相比于 BF 增加了元素的删除功能。但由于 CBFV 位数固定,当其无法满足最大计数时会发生溢出。DCF 在 CBF 基础上,增加溢出过滤器向量(Overflow Filter Vector,OFV),克服了 CBF 计数器位数固定导致的溢出问题。但是由于 OFV 位数多,其新建时间较长,另外,当前 OFV 因无法满足最大计数而重建时,旧数据的拷贝会造成巨大时间开销,进而导致 DCF 的平均操作效率低下。文献[11]针对目前 DCF 中存在的问题提出解决方法 DROFV(Delay Rebuild Overflow Filter Vector,DROFV),该方法将 OFV 的重建推迟至 DCF 操作空闲期,避免了计数值临时降低导致的 OFV 反复重建,同时也降低重建 OFV 对 DCF 操作造成的影响,但仍存在 DCF 平均操作效率低下问题。DCBF 方法具有一组 CBF,该方法检测数据对应计数器是否达到最大值,达到则新建 CBF 的方式,避免了 OFV 的重建,从而节省大量时间,但多次新建 CBF 仍造成一定的时间开销;同时由于该组 CBF 互相独立,计数器最大值随位数呈线性增长,并且存储元素时需同时记录其相应 CBF 信息,造成 CBF 无法轻易删除,因此较 DCF 占用更大内存空间。

针对 DCF 以及 DCBF 目前存在的问题,本文提出一种改进的实现方法——多维动态计数过滤器(Dimension Dynamic Count Filter,DDCF)。该方法采用多个 BFV 替代 OFV,以避免建立 OFV,同时也避免重建 OFV 时的数据拷贝,相较于 DCF 大大提升了发生溢出时的处理速率,提升了平均操作效率。相较于 DCBF,该方法避免了多次新建 CBF,节省了部分时间,同时由于其 CBF 和多个 BFV 之间为联合关系,计数器最大值与位数关系为指数增长,并且可以随需实时删除计数器最高位 BFV,因此节省了大量内存空间。

2　DCF 的原理

DCF 由 CBFV 和 OFV 两部分组成。其数据结构如图 1 所示。

两个向量的长度均为 m,其中 CBFV 计数器位数固定为 x($x \in N$,N 为自然数集合),OFV 位数根据溢出情况动态调整,假设为 y($y \in N$,N 为自然数集合)。图 1 中右一列为计数器值 VALUE,该值由两个向量中相应位连接起来的二进制数决定,例如第五个计数器中的值是 1026,二进制数由 OFV 中的 100 和 CBFV 中的 00000010 组成。

当前 DCF 能够存储的最大值 $MAX = 2^{(x+y)} - 1$,并且 MAX 随位数增加呈指数级增长。在集合中添加元素时,如果某个计数器的最大 VALUE > MAX,则需对 OFV 进行位拓展,步骤为①新建 OFVNEW,位数为 y + 1;②拷贝 OFV 数据至 OFVNEW;③销毁 OFV;④命名 OFVNEW 为 OFV。

在删除元素时,如果发现计数器中最大 $VALUE < 2^{(x+y)} - 1$,则说明当前 OFV 位数过高,造成了内存浪费,应减小 OFV 位数,步骤为①新建 OFVNEW,位数为 y - 1,如果 y - 1 = 0,结束;②拷贝 OFV 数据至 OFVNEW;③销毁 OFV;④命名 OFVNEW 为 OFV。

由上述分析可知,DCF 存在两方面问题。第一,OFV 的长度 m 通常较大,新建 OFV 时间较长。第二,拷贝旧数据会导致巨大的时间开销。

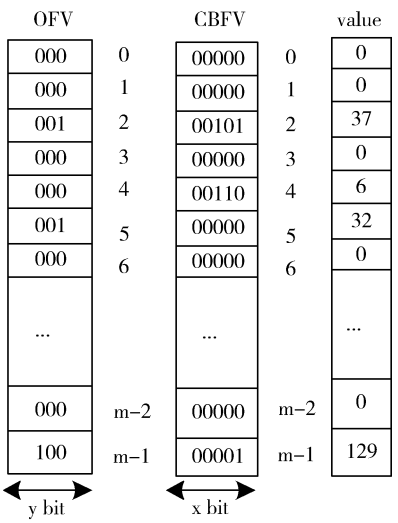

图 1　DCF 工作原理图

3　DCBF 的原理

DCBF 由一组 CBF 组成,其数据结构如图 2 所示。

图 2 中 CBF(0 - N)计数器为 x 位($x \in N$,N 为自然数集合),每个 CBF 计数器最大值 $MAX = 2^x - 1$,因此 DCBF 计数器的最大值为 N * MAX,其随位数增加呈线性增长。设每个计数器当前值为 VALUE(0 - N)。

使用 DCBF 在集合中添加元素时首先经过哈希运算映射到第一个 CBF(图 2 中为 CBF0)中,如果其对应

位置均大于 0,则说明该数据已经存在;如果其对应的位置不全大于 0,则该元素为新元素,此时应根据其哈希值依次查找 CBF 对应向量中计数器的值,如果此时计数器值未达到最大,则停止查找,将对应 CBF 中的计数器全部加 1;如果当前所有 CBF 中其对应计数器都已经达到最大值,则再新增一个 CBF,将其映射到新 CBF 中,最终 CBF 个数为 N。

使用 DCBF 删除元素时需找到该元素对应的 CBF,然后根据哈希算法找到其映射的位置,将相应计数器值减 1。由此可知,在保存元素时需同时保存其对应 CBF 信息。

由以上分析可知,DCBF 存在三方面问题。第一,由于 CBF 的长度以及计数器位数通常较大,多次新建 CBF 时间较长。第二,由于 DCBF 计数器最大值与总位数呈线性增长关系,随着元素的存入和更多 CBF 的建立,较 DCF 需要更多内存空间。第三,新的 CBF 建立后,即使其前面 CBF 中相应计数器的值小于 MAX,也无法删除新 CBF,造成内存空间的浪费。

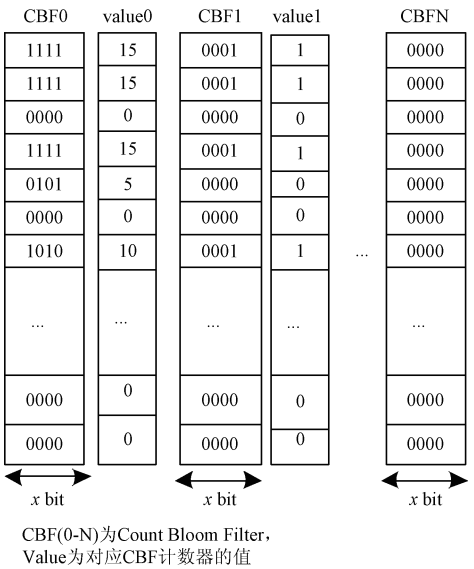

图 2 DCBF 工作原理图

4 DDCF 的原理

单维多位的 OFV 与单位 BFV 均由位串组成,新建时间与位串大小成正相关,而多位 OFV 较单位 BFV 大数倍,因此 DCF 中新建 OFV 对过滤器操作效率存在一定影响。OFV 重建时需将旧 OFV 中数据拷贝至新 OFV 中,由于 OFV 长度通常较大,因此旧数据拷贝对过滤器操作影响较大。

本文提出方法 DDCF 将 DCF 中单维多位 OFV 替换为多维单位 BFV,新建 BFV 较新建 OFV 快,同时处理溢出时仅针对计数器最高位 BFV 进行增加或删除,避免了旧数据的拷贝,从而提高了操作效率。其数据结构如图 3 所示。

DDCF 的 CBFV 部分与 DCF 相同,OFV 替换为 y 个 BFV,图 3 中左侧向量代表计数器高位。计数器的值 VALUE 由所有 BFV 与 CBFV 中相应位的二进制值联合决定。

当前 DDCF 能够存储的最大值 $MAX = 2^{(x+y)} - 1$。集合中添加元素时,如果某个计数器的最大 VALUE > MAX,则需在左侧增加 BFV。删除元素时,如果计数器中最大 $VALUE < 2^{(x+y)} - 1$,则销毁最左侧 BFV。

由以上分析可知,过滤器发生溢出时,新建 BFV 较新建 OFV 所需时间短,从而能够提升处理速度。同时,DDCF 对计数器位数进行拓展与减少时,避免了 OFV 旧数据的拷贝,节省大量时间。

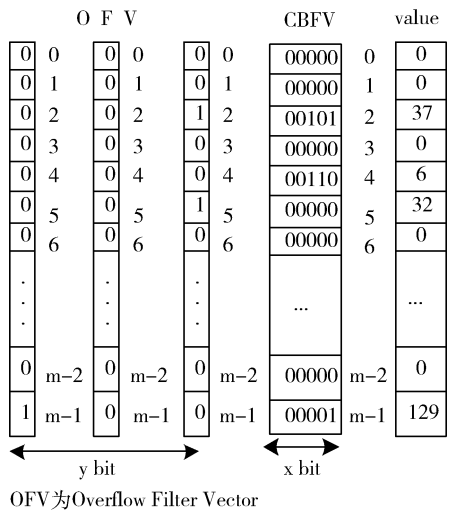

图 3 DDCF 工作原理图

5 对比实验

本实验的目的是验证本文提出的 DDCF 实现了 DCF 的全部功能,并分别测试 DCF、DROFV、DCBF 和 DDCF 在运行过程中所需时间及空间。

5.1 实验环境及方法

实验环境如表 1 所示。实验分别测试初始化时间、不重复数据平均插入时间、不重复数据平均删除时间、重复数据平均插入时间、重复数据平均删除时间、不重数据占用内存、不重数据占用内存。

5.2　实验结果分析

过滤器初始化时间与其大小有关，本实验设置过滤器集合为 1000000，错误率为 0.01，CBFV 与 CBF 均为 4 位。

由表 2 可知，DCF、DROFV、DCBF 和 DDCF 初始化耗时基本相同。这是由于过滤器均初始化 CBFV 或者 CBF，实验设置它们的位数与长度相同。

表 1　实验环境

CPU	Intel Core(TM) 2.67 GHz
内存	4.0GB
OS	Windows 7
IDE	pycharm
语言	python

表 2　初始化时间

	初始化平均时间
DCF	0.184s
DROFV	0.184s
DCBF	0.180s
DDCF	0.179s

由于不重复数据较难导致计数器的溢出，不涉及溢出处理。本实验分别测试连续插入 1、5000、10000 条数据所需时间。同样也分别测试删除 1、5000、10000 条数据所需时间。

依据表 3 和表 4 可知，数据为不重复数据时，DCF、DROFV、DCBF、DDCF 插入与删除操作的时间效率基本相同。这是由于不重复数据的插入没有造成计数器的溢出，不涉及溢出处理。

表 3　不重复数据平均插入时间

	插入 10000 不重数据平均时间	插入 5000 不重数据平均时间	插入 1 条数据平均时间
DCF	34.692s	17.358s	0.00350s
DROFV	34.692s	17.358s	0.00350s
DCBF	34.396s	17.250	0.00350s
DDCF	34.392s	17.252s	0.00351s

表 4　不重复数据平均删除时间

	删除 10000 条不同数据平均时间	删除 5000 条不同数据平均时间	删除 1 条数据平均时间
DCF	41.232s	20.921s	0.004s
DROFV	41.232s	20.921s	0.004s
DCBF	42.172	20.899	0.004s
DDCF	42.106s	20.844s	0.004s

插入重复数据较易导致计数器的溢出，因此实验通过重复数据插入时间来实现对溢出处理的测试。由于实验初始化 CBFV 和 CBF 均为 4 位，因此对于 DCF、DROFV 和 DDCF 而言，溢出发生在插入第 16，32，64 条数据时，对于 DCBF 来说溢出发生在插入第 16，31，46 条数据时。

删除重复数据将清除 DCF、DROFV 和 DDCF 计数器高位部分，清除发生在删除第 64，32，16 条数据时。对于 DCBF 来说，测试删除 46，31，16 条数据时所需时间。

依据表 5 与表 6 可知，当数据为重复数据时，DDCF 相对于 DCF 与 DROFV 的插入与删除操作时间效率有明显提高。

表 5　重复数据平均插入时间

	插入 16 条重复数据	插入 32(31)条重复数据	插入 64(46)条重复数据
DCF	0.121s	214.938s	522.938s
DROFV	0.121s	214.938s	522.938s
DCBF	0.211s	0.531s	0.803s
DDCF	0.107s	0.222s	0.521s

表 6　重复数据平均删除时间

	64 条重复删除至 63	32 条重复删除至 31	16 条重复删除至 15
DCF	313.687s	157.562s	0.0079s
DROFV	313.687s	157.562s	0.0079s
DDCF	0.011s	0.009s	0.008s
	46 条重复删除至 45	31 条重复删除至 30	16 条重复删除至 15
DCBF	0.015s	0.015s	0.015s

插入 16 条重复数据时，由于 CBFV 计数器最大值为 15，会产生溢出，需新建 OFV，因为此时无旧 OFV，所以无需旧数据的拷贝，但 OFV 位数较 BFV 大，所以消耗时间略大。同理可知，删除第 16 条重复数据时，

DCF、DROFV 与 DDCF 时间耗费基本相同。

插入 32 条,64 条数据时,DCF 与 DROFV 均需进行 OFV 的新建以及旧数据的拷贝操作,时间开销十分严重,且由表 5 可看出,插入重复数据越多,需拷贝旧数据越多,所需时间越长。而对于 DDCF,由于没有旧数据的拷贝,插入重复数据所需时间较少。同理,删除第 32 条,64 条重复数据时 DCF 与 DDCF 时间开销相差也较大。

依据表 5 与表 6 可知,当数据为重复数据时,DDCF 相对于 DCBF 较快。这是由于处理溢出问题时两者均没有拷贝旧数据操作,但是由于 DDCF 对 BFV 进行操作,DCBF 对 CBF 进行操作,而 BFV 计数器位数少,占用内存较小,因此对 BFV 的操作较 CBF 操作速度快。

插入不重复数据不涉及计数器溢出处理。本实验测试插入 1 条、5000 条、10000 条不重数据后过滤器所占内存大小。

插入重复数据导致计数器溢出,对于 DCF,DROFV,DDCF 来说溢出发生在插入 16 条,32 条,64 条重复数据时,而对于 DCBF 来说发生在插入 16 条,31 条,46 条数据时。

表 7 不重复数据占用内存

	插入 10000 条不重数据	插入 5000 条不重数据	插入 1 条数据
DCF	4.582MIB	4.582MIB	4.582MIB
DROFV	4.582MIB	4.582MIB	4.582MIB
DCBF	4.582MIB	4.582MIB	4.582MIB
DDCF	4.582MIB	4.582MIB	4.582MIB

表 8 重复数据占用内存

	插入 16 条重复数据	插入 32(31)条重复数据	插入 64(46)条重复数据
DCF	5.727MIB	7.305MIB	8.433MIB
DROFV	5.727MIB	7.305MIB	8.433MIB
DCBF	9.164MIB	14.046MIB	19.432MIB
DDCF	5.727MIB	7.305MIB	8.433MIB

依据表 7 和表 8 可知,当插入不重复数据时,不同过滤器占用内存均相同,这是由于插入不重复数据不涉及溢出处理,所测值均为初始化 CBFV 或 CBF 所占内存。当插入重复数据时,由于 DCF、DROFV 和 DDCF 每增加一个 OFV,计数器增加一位,其最大值增大为当前最大值的一倍,即随计数器位数增加呈指数级增长;而 DCBF 每增加一个 CBF,计数器增加 4 位,计数器最大值增大固定值 15,随位数增加呈线性增长。因此,插入较多重复数据后,DCF,DROFV 和 DDCF 过滤器占用内存相同,并且较 DCBF 小。

实验结论:本文所提 DDCF 较 DCF、DROFV 和 DCBF 在不影响不重复数据的插入与删除操作时间效率的情况下,优化了对溢出情况的处理,提高了溢出处理速率,并且较 DCBF 节省内存空间。

6 结束语

本文通过分析 DCF 和 DCBF 工作原理,指出 DCF 在新建、重建 OFV 时存在的时间开销严重问题和 DCBF 新建 CBF 速度慢、占用内存空间大问题,提出了一种 DCF 的改进实现方法。该方法采用多个 BFV 替代 OFV,以避免建立 OFV,同时也避免了重建 OFV 时的数据拷贝。本文还通过实验验证了在重复数据较多时本文所提出的 DDCF 插入以及删除操作较 DCF、DROFV 和 DCBF 降低了处理时间,同时也较 DCBF 节省了较大内存空间。

参 考 文 献

[1] 肖明忠,代亚非. Bloom Filter 及其应用综述[J]. 计算机科学,2004,31(4):180 - 183

[2] 张每平,许力. 无线多媒体传感器网络共享密钥发现协议设计与实现[J]. 微计算机应用,2009,30(7):19 - 24

[3] Mullin J K. Optimal semijions for distributed database systems[J]. IEEE Transactions on Software Engineering,1990,16(5):558 - 560

[4] Ahmed R, Boutaba R. Plexus: a scalable peer - to - peer protocol enabling efficient subset search[J]. ACM Transactions on Net-

working,2009,17(1):130 – 143
[5] 张一鸣,卢锡城,郑倩冰等. 一种面向大规模 P2P 系统的快速搜索算法[J]. 软件学报,2008,19(6):1473 – 1480
[6] Fan L, Cao P, Almeida J, et al. Summary cache:a scalable wide – area Web cache sharing protocol[J]. ACM Transactions on Networking,2000,8(3):281 – 293
[7] 郭浩然,田野. 信息中心网络对抗伪局部性缓存攻击方法研究[J]. 网络新媒体技术,2016,5(6):7 – 11
[8] 刘元珍. Bloom Filter 及其在网络中的应用综述[J]. 计算机应用与软件,2013,30(9):219 – 220
[9] 王宜青,陈庶樵,张震. 基于动态计数型过滤器的网络流公平抽样机制[J]. 计算机应用与软件,2014,31(11):139 – 142
[10] 孟金凤,高仲合. 基于 DCBF 的流抽样测量算法[J]. 计算机工程与应用,2015,51(17):92 – 95. [11] jiaomeng. Dynamic-CountFilter[EB/OL]. [2007,3,28]. http://blog. csdn. net/jiaomeng/article/details/1543751

作者简介

岳未然,男,1992 年生,硕士研究生,主要研究方向是网络与信息安全。
徐龙,男,1990 年生,硕士研究生,主要研究方向是网络与信息安全。
通信作者:赵辉

(上接第 63 页)

5　结束语

数字信号处理中,经常会遇到线性卷积与循环卷积的计算问题。本文采用反向思维,从有限长序列的线性卷积、周期卷积和循环卷积之间的关系出发,分析了不同点数的循环卷积与线性卷积之间的对应关系。提出了一种利用线性卷积乘法竖式法计算循环卷积的快速便捷的新算法,计算过程简单,且不易出错,适用于任意点数的循环卷积的计算。本文中给出了算法总流程图,结合不同点数,经过 MATLAB 软件仿真实现,验证了新算法的正确性。本文中的分析扩展了有限长序列循环卷积的计算方法,能够很好地处理解决实际问题。

参　考　文　献

[1] Won Y. Yang, Tae G. Chang, Ik H. Song, et al. Signals and Systems with MATLAB:(影印版)[M]. 北京:电子工业出版社,2012. 13 – 20
[2] 赵鸿图,刘艳辉. 循环卷积的时域与频域算法研究[J]. 计算机工程与设计,2014,35(5):1678.
[3] Philips L,Parr M,Riskin EA. Signals,Systems,ans Transforms[M]. Fourth Edition,NJ,Pearson Education Inc,2008. 213 – 227
[4] 程佩青. 数字信号处理教程(第四版)[M]. 北京:清华大学出版社,2013. 150 – 154
[5] 陈辉金,黄喜军. 圆周卷积的竖式法求解分析[J]. 电气电子教学学报,2012,34(3):19 – 20
[6] 李昌利. “数字信号处理”中循环卷积的简单计算方法[J]. 电气电子教学学报,2012,34(6):31 – 33
[7] 高西全,丁玉美,阔永红. 数字信号处理 – 原理、实现及应用(第二版)[M]. 西安:西安电子科技大学出版社,2010. 71 – 72
[8] 刘国良. 基于 Matlab 的离散卷积[J]. 现代电子技术,2009,32(5):125 – 126
[9] Vinary K. Ingle. Digital signal processing using Matlab:(影印版)[M]. 北京:可惜出版社,2003. 125 – 131
[10] 胡广书. 数字信号处理理论、算法与实现(第三版)[M]. 北京:清华大学出版社,2012. 48,130 – 131
[11] Alan V Oppenhim,Ronald W Schafer, Discrete – time signal process[M]. Prentice – Hall,Inc,2009. 548 – 550
[12] 徐守时. 信号与系统(第 2 版)[M]. 北京:清华大学出版社,2016. 96 – 97
[13] 王益艳. 离散序列线性卷积和循环卷积的关系[J]. 四川文理学院学报,2015,25(5):32 – 33

作者简介

毕春艳,女,1984 年 1 月出生,硕士研究生,讲师,研究方向为信号处理、数字图像处理和通信系统。
徐晋,男,1981 年出生,硕士研究生, 讲师,研究方向为微电子,信号处理与嵌入式系统。