ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.20, No.1, January 2009, pp.96−108
doi: 10.3724/SP.J.1001.2009.03458

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

布鲁姆过滤器查询算法*

谢 鲲[1+], 文吉刚[1], 张大方[2], 谢高岗[3]

[1](湖南大学 计算机与通信学院,湖南 长沙 410082)

[2](湖南大学 软件学院,湖南 长沙 410082)

[3](中国科学院 计算技术研究所 网络与普适计算研究部,北京 100190)

Bloom Filter Query Algorithm

XIE Kun[1+], WEN Ji-Gang[1], ZHANG Da-Fang[2], XIE Gao-Gang[3]

[1](College of Computer and Communication, Hu'nan University, Changsha 410082, China)

[2](School of Software, Hu'nan University, Changsha 410082, China)

[3](Networking and Ubiquitous Computing Department, Institute of Computing Technology, The Chinese Academy of Sciences, Beijing 100190, China)

+ Corresponding author: E-mail: xiekun@hnu.cn

Xie K, Wen JG, Zhang DF, Xie GG. Bloom filter query algorithm. *Journal of Software*, 2009,20(1):96−108.
http://www.jos.org.cn/1000-9825/3458.htm

Abstract: This paper surveys the mathematics behind Bloom filters, some important variations and network-related applications of Bloom filters. The current researches show that although Bloom filters start drawing significant attention from the academic community and there has been considerable progress, there are still many unknown dimensions to be explorered. The research trends of Bloom filter algorithm are foreseen in the end.

Key words: Bloom filter; computer network; distributed computing; data set membership query

摘 要: 从理论和应用两方面系统地综述了布鲁姆过滤器查询算法迄今为止的主要研究成果,分析了目前布鲁姆过滤器查询算法的研究现状,最后展望了布鲁姆过滤器查询算法未来可能的研究方向.

关键词: 布鲁姆过滤器,计算机网络,分布式计算,集合从属查询

中图法分类号: TP393 文献标识码: A

随着计算技术的飞速发展,数据库、网络和其他应用中的数据集合规模呈几何增长.日益增长的网络信息空间给网络资源存储、管理、定位、交互带来了前所未有的挑战.信息的表示和查找是计算机网络和大多数计算机应用程序的核心,是进行资源查找、定位、访问和交换的关键,是网络和分布式系统资源共享最基本的操作.当数据集合变得越来越大,集合的表示和访问就越来越困难,如何表示海量资源信息,如何快速、高效地在海量信息中查找所需的资源成为国、内外学术界的研究热点.因此,在高速发展的网络和计算机系统环境下,研究简洁的结构表示资源信息,设计与之对应的高效查询算法查找定位资源信息,成为提升网络软件体系结构,进行

* Supported by the National Natural Science Foundation of China under Grant Nos.90718008, 90604015 (国家自然科学基金)
Received 2007-09-12; Accepted 2008-08-07

高效的大规模数据管理的关键,亦成为多种网络应用得以最终推广的基础.

作为一种精简的信息表示方案,布鲁姆过滤器(Bloom filter)[1]能够满足高速网络发展中高效资源交互和查找需求.布鲁姆过滤器对集合采用一个位串表示,并支持元素的哈希查找.既能表示集合,支持集合元素查询,又能有效地过滤掉不属于集合的元素.其算法结构的实质是将集合中的元素通过 k 个哈希函数映射到位串向量中.与传统的哈希存储表不同,布鲁姆过滤器中哈希表退化成一个位串向量 V,一个元素仅占用几个比特位.传统的树型查询算法和哈希查询算法存储空间与元素自身大小和集合规模直接相关,而布鲁姆过滤器查询算法所需存储空间与元素自身大小和集合规模无关,仅与元素映射到向量的位数相关.使用 m bit 的 V 向量可以表示 n 个元素的集合,每个元素平均只需要 m/n 比特位,极大地节约了存储空间.布鲁姆过滤器存在假阳性误判(即属于集合中的元素而误判为不属于集合中)[1,2].因此,布鲁姆过滤器是一种允许存在一定误判的情况下,存储空间节俭的哈希结构,是在查询准确率和存储代价之间的折衷.

由于哈希查找的常数时间和存储空间开销较小,布鲁姆过滤器查询算法具有很好的实用价值[3,4].自算法首次提出以来,已广泛应用于各种计算机系统中,用以表达庞大数据集合、提高查找效率.最初的应用主要集中在数据库操作、字典查询和文件操作[5−9]方面.最近 10 年,随着网络研究的发展以及新的覆盖网和 P2P 网络应用技术的涌现,出现了布鲁姆过滤器查询算法研究高潮,应用于网络的领域和方式越来越多[3,10]:包括覆盖网和 P2P 节点协作交互[11−13]、资源路由[14]、数据帧路由标签[15]、网络测量管理[16−24]、网络入侵检测[25−27]、Ad hoc 网络中信息共享[28]和传感器网络数据过滤和路由[29,30]等网络应用领域.据 Brooder 和 Mitzenmacher 预言[3],目前这些布鲁姆过滤器查询算法在网络上的应用还只是冰山一角,随着布鲁姆过滤器查询算法越来越多地被研究人员所认识和重视,它将在现代计算机网络和一些新的学术领域得到更为广泛的应用.

布鲁姆过滤器查询算法的理论研究在最近几年备受关注.从目前公开发表的国内外文献来看,出现了多种标准布鲁姆过滤器(standard Bloom filter)查询算法[1]的改进算法,包括:计数式布鲁姆过滤器(counting Bloom filter)查询算法[31,32],压缩布鲁姆过滤器(compressed Bloom filter)查询算法[2],光谱布鲁姆过滤器(spectral Bloom filter)查询算法[33],拆分型布鲁姆过滤器(split Bloom filter)查询算法[34],动态布鲁姆过滤器(dynamic Bloom filter)查询算法[35],可扩展布鲁姆过滤器(scalable Bloom filter)查询算法[36],多维布鲁姆过滤器查询算法[35,37]和分档布鲁姆过滤器(basket Bloom filter)查询算法[38]等.这些布鲁姆过滤器扩展算法,为布鲁姆过滤器查询算法的研究发展做出了大量贡献,但是由于前期受算法领域的关注度不够,该算法研究正处于上升阶段,还存在许多有趣的理论问题有待解决.

本文的目的就是尝试对当前与该专题有关的研究成果进行汇总.本文第 1 节概述标准布鲁姆过滤器查询算法原理.第 2 节介绍布鲁姆过滤器查询算法的研究现状.第 3 节介绍布鲁姆过滤器典型应用模式和应用分类.第 4 节总结并展望布鲁姆过滤器可能的研究方向.本文围绕布鲁姆过滤器的原理、扩展和应用进行探讨,我们希望通过本文的工作,给分布式交互查询的研究者、网络协议设计者,尤其是正在进入相关研究领域的人员一些启发和帮助.

1 布鲁姆过滤器查询算法基本原理

1.1 标准布鲁姆过滤器查询算法基本操作

标准布鲁姆过滤器查询算法具体描述如下,如图 1 所示.数据集合 $S=\{s_1,s_2,\ldots,s_n\}$ 共有 n 个元素通过 k 个哈希函数 $h_1,h_2,\ldots,h_k$ 映射到长度为 m 的位串向量 V 中.通常,布鲁姆过滤器表示的汇总(summary)信息就是向量 V,用 BF 表示.每一个哈希函数相互独立且函数的取值范围为 $\{0,1,2,\ldots,m-1\}$.

元素插入操作就是将元素插入到集合,完成元素到布鲁姆过滤器的表示;元素的查询操作就是利用布鲁姆判断元素是否在集合中.布鲁姆过滤器在使用前必须初始化,即将 V 向量的各位初始化为 0.集合中元素的插入过程就是元素到向量的映射过程.

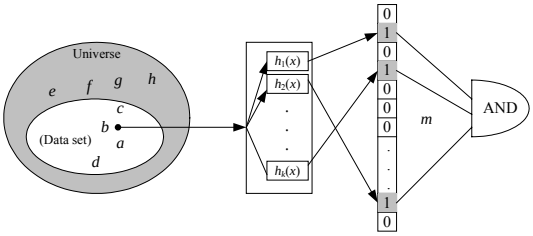

Fig.1 Standard Bloom filter

图 1 标准布鲁姆过滤器原理

- 元素插入操作,“Insert x into S.”,如图 2 所示.

1) 计算元素 x 的 k 个哈希地址,即 $h_1(x),h_2(x),\ldots,h_k(x)$;

2) 将布鲁姆过滤器对应的 k 个位置置位,即 $BF[h_1(x)]=BF[h_2(x)]=BF[h_3(x)]=\ldots=BF[h_k(x)]=1$.

- 元素查询操作,“Query if $x\in S$?”,与插入操作相反,仍分为两个步骤:

1) 计算 x 相应的 k 个哈希地址,即 $h_1(x),h_2(x),\ldots,h_k(x)$;

2) 检查布鲁姆过滤器向量这 k 个对应位置:$BF[h_1(x)],BF[h_2(x)],BF[h_3(x)],\ldots,BF[h_k(x)]$是否都为 1.它们只要有一个是 0,$x$ 必不在集合中.如果全为 1,则 x 可能在集合中,但不一定.此时可能出现所谓假阳性误判,即将不属于集合的元素误判为属于集合中.如图 3 所示,元素 x 不属于集合,但是属于集合的元素 a,b,c,d 将 x 的映射位置置位,导致误判 x 在集合中.

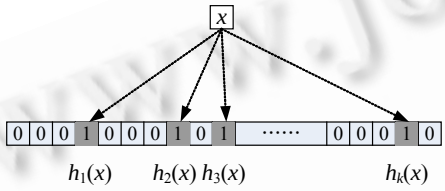

Fig.2 Insertion of element Bloom filter

图 2 标准布鲁姆过滤器元素插入

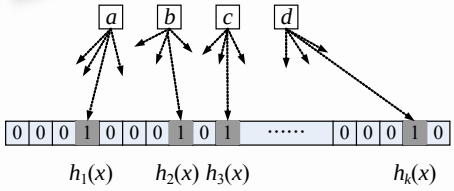

Fig.3 False positive of Bloom filter

图 3 布鲁姆过滤器假阳性

可以看出,布鲁姆过滤器结构的实质是将集合中的元素通过 k 个哈希函数映射到位串向量中,对于集合中每一个元素只需要保存几个比特位.布鲁姆过滤器作为一种支持集合查询的数据结构,在达到其高效、简洁地表示集合效果的同时,却存在某元素不属于数据集合而被指称属于该数据集合的可能性,即假阳性误判,而不存在假阴性误判[2](属于集合中的元素而误判为不属于集合中).

从图 3 可以看出,布鲁姆过滤器查询算法存在查询假阳性误判,因此,在实际应用中,必须对查询误判率进行评估,设计合适的布鲁姆过滤器,降低查询误判对应用的影响.

我们用 p 表示向量 V 中任一 bit 位为 0 的概率,那么任一 bit 位为 1 的概率为 p–1.假设哈希函数取值服从均匀分布,则当集合中所有元素都映射完毕后,V 向量任意一位为 0 的概率:

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

当不属于集合的元素误判断属于集合时,元素在 V 向量的对应位置都必须为 1.即元素的误判率为[2]

$$f^{BF}(m,k,n)=(1-p')^k\approx(1-p)^k \tag{2}$$

即

$$f^{BF}(m,k,n)=(1-p)^k=(1-\mathrm{e}^{-kn/m})^k=\exp\left(k\ln(1-\mathrm{e}^{-kn/m})\right) \tag{3}$$

如果给定布鲁姆过滤器的误判率上限 f_0,在过滤器长度和哈希函数个数 k 一定的情况下,可以直接通过式(3)计算出布鲁姆过滤器可以最多表示的元素个数 n_0.在后面的讨论中,我们称 n_0 为布鲁姆过滤器在误判率 f_0 下

表示集合规模上限:

$$n_0 = -\frac{\ln(1-\mathrm{e}^{\ln f_0/k})\cdot m}{k} \tag{4}$$

显然,哈希函数个数 k 越多,一方面,元素在向量中映射的位就越多,表达元素信息越多,误判率可能降低;另一方面,向量置1位变得稠密,误判率可能增加.如图4所示,误判率 f 随着哈希函数个数 k 的变化趋势,图中,当 k=9时,误判率最小为 0.002 135.

令 $g(k)=k\ln(1-\mathrm{e}^{-kn/m})$,可知函数 g 和 f 可以同时达到最小值,对 g 取 k 的导数,得:

$$\frac{\mathrm{d}g(k)}{\mathrm{d}k} = \ln(1-\mathrm{e}^{-kn/m}) + \frac{kn}{m}\frac{\mathrm{e}^{-kn/m}}{1-\mathrm{e}^{-kn/m}}.$$

令 $\frac{\mathrm{d}g(k)}{\mathrm{d}k}=0$,则

$$k_{\min} = (\ln 2)\left(\frac{m}{n}\right) \tag{5}$$

当 k=$k_{\min}$ 时,可以使 g 达到最小值,这时,

$$p = (1-p) = \frac{1}{2} \tag{6}$$

那么,元素出现查询误判率最小值为

$$f(k_{\min}) = (1/2)^{k_{\min}} = \left(1/2\right)^{(\ln 2)\frac{m}{n}} = (0.6185)^{\frac{m}{n}} \tag{7}$$

k 是哈希函数的个数,应为整数,即:

$$k = \left\lfloor \ln 2\left(m/n\right)\right\rfloor \tag{8}$$

式(8)给出了当集合元素个数 n 和向量空间 m 一定时,如何选择合适的哈希函数个数 k 使得布鲁姆过滤器误判率 f 最小.

上面讨论了误判率和哈希函数的关系.实际应用中,有时又需要指定可以接受的查询误判率,选择哈希函数个数,使得布鲁姆过滤器占用空间最小.根据式(3),同样可以计算出布鲁姆过滤器所需的存储空间:

$$m = -\frac{kn}{\ln\left(1-(f^{BF})^{\frac{1}{k}}\right)} \tag{9}$$

标准布鲁姆过滤器向量空间和哈希函数个数的关系如图5所示,当 k=9 时,存储空间最小值为 2 560 bit.

为了取得布鲁姆过滤器向量空间最小值,对 m 取 k 的导数:

$$\frac{\mathrm{d}m(k)}{\mathrm{d}k} = -\frac{\ln\left(1-(f^{BF})^{\frac{1}{k}}\right)\cdot n - n\cdot\dfrac{1}{1-(f^{BF})^{\frac{1}{k}}}\cdot(f^{BF})^{\frac{1}{k}}\cdot\ln\left((f^{BF})^{\frac{1}{k}}\right)}{\left(\ln\left(1-(f^{BF})^{\frac{1}{k}}\right)\right)^2} \tag{10}$$

当哈希函数个数如式(8)所示时,$\frac{\mathrm{d}m(k)}{\mathrm{d}k}=0$.因此,$k_{\min}=(\ln 2)\left(\frac{m}{n}\right)$ 是当集合规模和查询误判率一定的情况下,最优化布鲁姆过滤器所需存储空间达最小的 k 值.在实际应用中,可以根据式(8)分析所需的最小误判率和最小存储空间.

布鲁姆过滤器最大的特征是空间简洁和查询方便.使用标准布鲁姆过滤器,增加一个元素到集合,需要进行 k 次哈希运算,其一次元素插入操作的时间复杂度为 $O(k)$.当判断元素是否在集合中时,同样需要进行 k 次哈希计算,完成一次元素查询的时间复杂度为 $O(k)$.对于 n 个元素的集合,只需要 m 位向量 V,其空间复杂度为 $O(m)$.使用布鲁姆过滤器完成集合存储,只需要为每个元素平均保存 m/n 位,十分简洁.传统的树型查询算法和哈希查询算法存储空间与元素自身大小和集合规模直接相关,而布鲁姆过滤器查询算法所需空间与元素自身大小无关,

仅与元素映射到向量的位数相关.因此布鲁姆过滤器十分适合于存储空间受限又能允许稍许误判的场合.第 2 节中所述的各种扩展算法都延续了这个本质的特点.

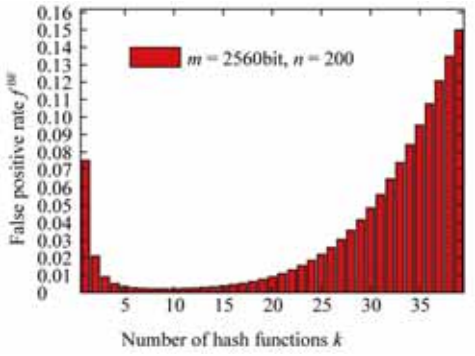

Fig.4 False positive rate vs. the number of hash functions

图 4 查询误判率与哈希函数个数关系

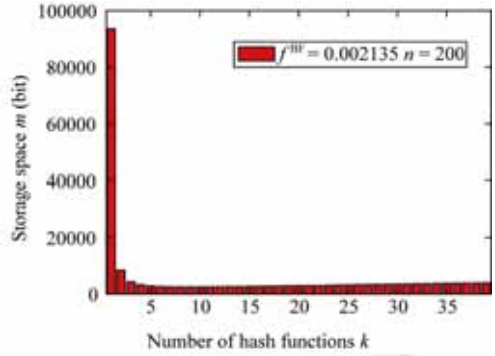

Fig.5 Storage vs. the number of hash functions

图 5 存储空间与哈希函数个数关系

标准布鲁姆过滤器是现有各种布鲁姆过滤器查询算法研究和扩展的基础.为了下文说明方便,用三元组 $BF=\{n,m,k\}$ 形式化表示一个标准布鲁姆过滤器,其中 n 为集合 S 的元素个数;m 为向量 V 的长度;k 为哈希函数个数.

2 布鲁姆过滤器的典型扩展算法

从目前公开发表的国内外文献来看,出现多种标准布鲁姆过滤器查询算法(standard Bloom filter)的扩展算法,主要包括有:

(1) 计数式布鲁姆过滤器查询算法[31,32]:由于标准布鲁姆过滤器中的 V 向量所有位被集合元素所共享,仅支持集合元素的插入操作,不支持元素的删除.为了支持集合动态变化时元素删除操作,文献[31]提出计数式布鲁姆过滤器(counting Bloom filter).如图 6 所示,用 Counter 计数器替代 V 向量的 bit 位,通过对元素对应的 k 个 Counter 计数器加、减运算,完成元素的插入和删除.

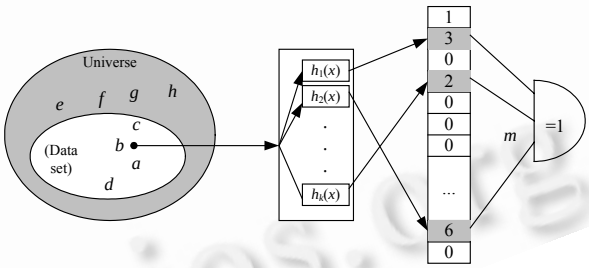

Fig.6 Counting Bloom filter principle

图 6 计数式布鲁姆过滤器原理

使用计数式布鲁姆过滤器,元素的插入和删除操作分别将对应布鲁姆过滤器的 k 个 Counter 计数器加 1 或者减 1.在判断元素是否在集合中时,只需判断这 k 个 Counter 计数器是否都大于 1.

计数式布鲁姆过滤器是布鲁姆过滤器查询算法的一大改进,提供集合元素的删除操作.使用计数器代替向量中的 bit 位,虽然可以解决元素动态删除问题,但是因为计数式布鲁姆过滤器的存储空间正比于 Counter 计数器的长度,过大的计数器会使布鲁姆过滤器所需的存储空间成倍增加;而过小的计数器很容易造成元素表示时计数器溢出.因此,在实际应用时,需要对布鲁姆过滤器中计数器的长度进行简单估计,根据集合元素增加与删除的统计规律,选择适当的计数器长度.

2) 光谱布鲁姆过滤器查询算法[33]:在计数式布鲁姆过滤器基础上,文献[33]提出光谱布鲁姆过滤器.假设集合的元素具有多样性(multiplicity),即集合中的元素可能存在多个副本.用元素对应的 k 个 Counter 计数器序列中最小的计数器作为元素出现的频度估计,从而过滤掉频度小于一定阈值的元素.

3) 压缩布鲁姆过滤器查询算法[2]:文献[2]发现,当标准布鲁姆过滤器设计为最小误判率时,布鲁姆过滤器向量这种由 0 和 1 组成的随机位串中任意位置为 0,1 的概率均为 1/2,若将此位串作为消息在网络中直接传递,不能取得任何压缩效果.因此,文献[2]提出压缩型布鲁姆过滤器,将新的压缩向量作为消息在分布式系统传递.压缩布鲁姆过滤器采用算术编码技术将 m 位长的 V 向量利用极限熵值:

$$H(p)=-p\log_2 p-(1-p)\log_2(1-p) \tag{11}$$

进行压缩,得到 z 位长的压缩向量:

$$z=H(p)\cdot m \tag{12}$$

由式(1),k 用 m,n 和 p 表示,可以得出:

$$k=-\frac{m}{n}\ln(p) \tag{13}$$

那么,布鲁姆过滤器查询误判率式(3)可以转换为

$$\begin{aligned} f&=(1-p)^k=(1-p)^{-\frac{m}{n}\ln p}=(1-p)^{-\frac{\ln p}{H(p)}\left(\frac{z}{n}\right)}\\ &=\exp\left(\frac{-\ln p\ln(1-p)}{(-\log_2 e)\left(p\ln p+(1-p)\ln(1-p)\right)}\left(\frac{z}{n}\right)\right)\end{aligned} \tag{14}$$

令

$$\gamma=\frac{p}{\ln(1-p)}+\frac{(1-p)}{\ln p},$$

求 f 的最小值可以通过求 γ 的最大值获得,经过计算,误判率 f 的最小值为

$$f=\mathrm{e}^{-\frac{z}{n}\ln 2}=(0.5)^{\frac{z}{n}}<(0.618)^{\frac{z}{n}} \tag{15}$$

利用极限熵值进行压缩,还可以改进布鲁姆过滤器的误判率.令式(7)的误判率和式(15)误判率相等,即

$$z=(\ln 2)m\approx 0.693m.$$

可以获得 69.3%的压缩率.

实际中,当获得相同误判率时,需要传输的压缩后向量长度 z 可能会比实际的压缩前向量长度 m 小很多.虽然压缩布鲁姆过滤器中压缩后的向量长度 z 很小,但其原始向量长度 m 可能很长,向量置 1 位变得十分稀疏,布鲁姆过滤器的误判率自然降低.

4) 拆分型布鲁姆过滤器查询算法[34]、动态布鲁姆过滤器查询算法[35]、可扩展布鲁姆过滤器查询算法[36]和文献[39]都是针对布鲁姆过滤器的扩展性问题而展开的研究,即当集合元素规模增长超过误判率 f_0 下表示集合规模上限 n_0 时,如何调整布鲁姆过滤器设计以适应这种增长.拆分型布鲁姆过滤器[34]和动态布鲁姆过滤器[35]及文献[39]解决增长问题的思路十分相似,通过将过滤器的位向量转换为由多个位向量组成的矩阵来解决可扩展性问题.可扩展布鲁姆过滤器[36]以增加长度成倍增长的过滤器向量来适应集合元素的增长,由多个长度不断增加的过滤器向量组成.文献[36]亦给出了可扩展布鲁姆过滤器算法的硬件设计.通过例 1 说明两类扩展算法的主要思想.

例 1:图 7 和图 8 给出动态布鲁姆过滤器 DBF 和可扩展布鲁姆过滤器 SBF 扩展实例.假设 DBF 和 SBF 都是从相同的初始标准布鲁姆过滤器开始扩展的,BF={n,m,k},设置 m=16 bit,k=2,为了画图说明方便,设置误判率上限 f_0=0.155,那么 BF 能够容纳的元素个数为

$$n_0=-\frac{\ln\left(1-\mathrm{e}^{\frac{(\ln 0.155)}{2}}\right)\cdot 16}{2}=4.003.$$

n_0 是集合的元素个数,必为整数,n_0=4.集合中依次插入元素 a,b,c,d,e,f,g,h,i,两种算法的向量状态和运行机理分别如图 7 和图 8 所示.

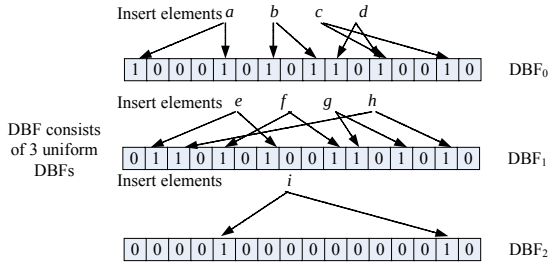

Fig.7 An example of dynamic Bloom filter process

图 7 动态布鲁姆过滤器运行实例

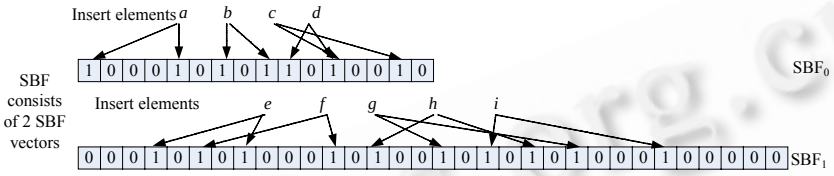

Fig.8 An example of scalable Bloom filter process

图 8 可扩展布鲁姆过滤器运行实例

如图 7 和图 8 所示,DBF 和 SBF 最开始的初始向量都为 1 个 16 bit 长的位向量.插入元素 a,b,c,d 后,两种过滤器的向量状态分别为 DBF_0 和 SBF_0.此时,新来一个元素 e,由于元素个数 n=5 超过了布鲁姆过滤器设计的 n_0=4 的限制,动态布鲁姆过滤器产生一个新的长度 m=16 bit 的空向量 DBF_1 用于表示新的元素,元素 e,f,g,h 映射到 DBF_1 后,依此类推,元素 i 的到来使得 DBF 产生新的 DBF_2 向量,这样共使用 3 个向量表示这 9 个元素,每个向量的长度都是 16 bit,也就是一个 3×16 bit 矩阵.而扩展布鲁姆过滤器则是当 e 到来时产生一个新的长度为 m=32 bit 的向量 SBF_1 用来表示新插入的元素,这个新向量可以容纳 8 个元素.元素 e,f,g,h,i 都能够表示到 SBF_1 中,使用 SBF 只需要扩展一次.在大数据量的情况下,可扩展布鲁姆过滤器比动态布鲁姆过滤器的扩展轮数明显减少,误判率和查询时间明显改进.

5) 分档布鲁姆过滤器查询算法[38]:在布鲁姆过滤器领域考察查询代价,算法设计采用区分服务的思想,将集合 S 根据元素的查询失效代价分为 L 档,通过对高代价子集合分配多数目哈希函数,降低查询失效率;对低代价子集合分配少数目哈希函数,适当增加查询失效率,使集合查询总代价最小.

将集合 S 根据元素的查询失效代价分为 L 档:

$$S=\{\{S_1\},\{S_2\},...,\{S_L\}\}.$$

每档子集合的元素个数为 $n_i=|S_i|$,子集合 S_i 中的元素发生查询失效时,所需付出的额外 I/O 操作代价为 $c_i(1\le i\le L)$,子集合 S_i 的哈希映射函数个数为 $k_i(1\le i\le L)$,对应的假阳性误判率为 $f_i(1\le i\le L)$.

文献[38]中定义集合查询失效总代价为各子集合查询失效代价之和:

$$F=n_1f_1c_1+n_2f_2c_2+...+n_Lf_Lc_L \tag{16}$$

考察了每档子集合最小误判率和算法设计的关系,发现每档子集合可以同时达到最小误判率.即每档元素需要映射的哈希函数个数为

$$k_i=\ln 2\cdot\left(r_im\Big/\sum_{j=1}^{L}n_jr_j\right) \tag{17}$$

此时,可达的集合最低查询失效总代价表达式为

$$F_{baskets}(L)=\sum_{i=1}^{L}n_ic_i f_i=\sum_{i=1}^{L}n_ic_i\left(\frac{1}{2}\right)^{\ln 2\cdot\left(r_im\Big/\sum_{j=1}^{L}n_jr_j\right)} \tag{18}$$

其中, $r_i=\ln(f_i)$ $(1\le i\le L)$.这样,将每档子集的最小误判率代入集合查询失效总代价表达式,得出集合最低查询失效总代价目标函数,该函数由参数 r-pair$(r_i,r_j)(1\le i,j\le L)$决定.通过求解式(18)函数最小值,获得每档映射哈希函数个数 k_i,完成元素的插入和查找.

6) 多维布鲁姆过滤器(multi-dimensional Bloom filter)查询算法[35,37]:用多个标准布鲁姆过滤器表示多维集合的单属性域,多过滤器共同完成元素的表示及是否属于集合的查询判断.

表 1 列出了目前布鲁姆查询算法的主要扩展.从表 1 可以看出,目前存在的主要扩展算法着眼点各不相同:计数式布鲁姆过滤器首次支持集合的删除,极大地拓宽了算法的应用场合,文献[31]提出了布鲁姆过滤器在 Web cache 中的应用,成为目前该领域引用率最高的文献;光谱布鲁姆过滤器能够完成元素的频度估计,可拓展算法应用于数据库冰山查找等方面;压缩布鲁姆过滤器可以改变误判率,减少布鲁姆过滤器作为消息传递的消息长度,进而减少传输带宽,但是却增加了消息的压缩和解压缩的代价;布鲁姆过滤器可扩展性研究,使布鲁姆过滤器能够用于集合规模不断增加的应用环境中,尤其适应于数据集合规模事先无法预测、不断增长的应用场合;分档布鲁姆过滤器适用于应用环境存在区分服务和需区分对待元素的场合,可大幅度地减少集合总体查询代价,由此降低资源总消耗;多维布鲁姆过滤器查询算法[35,37]将元素集合由传统的单维集合扩展为多维元素集合,对于扩展网络应用具有很强的实际意义,如网络业务流五元组(Source Address,Dest Address,Source Port,Dest Port,Protocol)的处理.目前这些扩展算法,在解决单一应用问题时有较好的性能,但是对于复杂的应用,需要深入研究应用的特点,融合各种扩展算法的优点,设计新的算法.

Table 1 Extensions of Bloom filter

表 1 布鲁姆过滤器查询算法的主要扩展

Bloom filter name	Deletion support for set element	Compression support	Dynamic data set support	Multi-Dimension support	Differentiating query support
Counting Bloom filter	√				
Spectral Bloom filter	√				
Compressed Bloom filter		√			
Split Bloom filter			√		
Dynamic Bloom filter			√		
Scalable Bloom filter			√		
Multi-Dimensional Bloom filter				√	
Basket Bloom filter					√

除了上述主要的布鲁姆过滤器查询扩展算法外,目前存在的布鲁姆过滤器算法研究还有:为了设计更有效的计数式布鲁姆过滤器,文献[40]研究计数器溢出条件,运用层次化设计思想和哈夫曼编码,设计新的布鲁姆过滤器以进一步减少空间消耗,使之适用于网络处理器;文献[41]研究查询元素和集合元素相似度的查询“Is x close to an element of S?”,提出距离敏感的布鲁姆过滤器(distance-sensitive Bloom filter);文献[42]将 Power of Two 和布鲁姆过滤器相结合,提出用两组哈希函数进行元素映射和查询的双选择布鲁姆过滤器,进一步减少过滤器查询算法误判率;在此基础上,文献[43]定义填充因子(fill factor)为所有元素被哈希映射后位串中单 bit 的置位情况,利用分割的多组哈希函数(partitioned hashing)为元素选择填充因子最小的哈希函数组来为元素置位,从而减少查询误判率;文献[44]研究使用布鲁姆过滤器检测流数据中重复数据的方法;文献[45]研究布鲁姆过滤器中哈希函数设计,指出两个哈希函数组成的 $g_i(x)=h_1(x)+ih_2(x)(0\le x\le k-1)$可以胜任布鲁姆过滤器中哈希函数设计,减少布鲁姆过滤器应用时的计算量;文献[10,32]研究布鲁姆过滤器支持状态空间查询问题,提出了近似表示协作状态机的方法,给出在网络视频拥塞控制的使用实例;文献[46,47]讨论了布鲁姆过滤器中映射空间的数据还原问题,在独立空间布鲁姆过滤器基础上,利用哈希映射串的重叠和数量的一致性来解决同源哈希映射串拼接成源串(元素)的问题,为元素的还原创造了条件,并提出了轻量级拒绝服务攻击检测算法;文献[48]探讨布鲁姆过滤器代数运算和集合查询的关系,定义布鲁姆过滤器的“并”、“交”、“异或”、“补”、“差”代数运算,进行实

验实现和分析,发现布鲁姆过滤器的“并”、“交”运算能够支持集合并集交集的元素查询,这一结论可以简化利用布鲁姆过滤器进行的系统设计.

3 布鲁姆过滤器的应用探讨

通常来说,布鲁姆过滤器在实际应用中扮演两种角色,第 1 种角色是作为一种压缩的数据集合(集合的缩减表示),可以替代原始的数据集合,完成元素是否在集合的查询判断;第 2 种角色,布鲁姆过滤器用作摘要消息在数据库等其他分布式节点中进行传递.

3.1 布鲁姆过滤器的典型应用模式

布鲁姆过滤器在分布式系统中最典型的应用是作为文件目录信息摘要在分布式交互系统中传递.如图 9 所示,以 Web 缓存系统设计为例,针对 ICP 广播请求大幅度占用网络带宽的缺点,Summary Cache[31]和 Cache Digest[49]通过保存其他缓存代理文件目录摘要来减少请求数.为了减少文件目录的存储空间和交互传递的带宽开销,Summary Cache 的文件目录用布鲁姆过滤器表示,将文件目录存储到代理的高速缓存 DRAM 中,可以加快文件查找过程,而 DRAM 价格昂贵且容量有限,不适合存放大数据结构,使用布鲁姆过滤器表示的文件目录摘要简洁,查找方便.而且缓存代理交互使用的是布鲁姆过滤器表示的文件目录,可以减少交互时的带宽消耗.这样,对待文件查询请求的过程是:首先检查本身的缓存和其他兄弟缓存的用布鲁姆过滤器表示的目录摘要,如果其他兄弟缓存目录摘要里有文档信息,则向相应的兄弟缓存代理服务器请求以获取文件,如果没有,就直接向上级服务器发送请求,如图 9 所示.

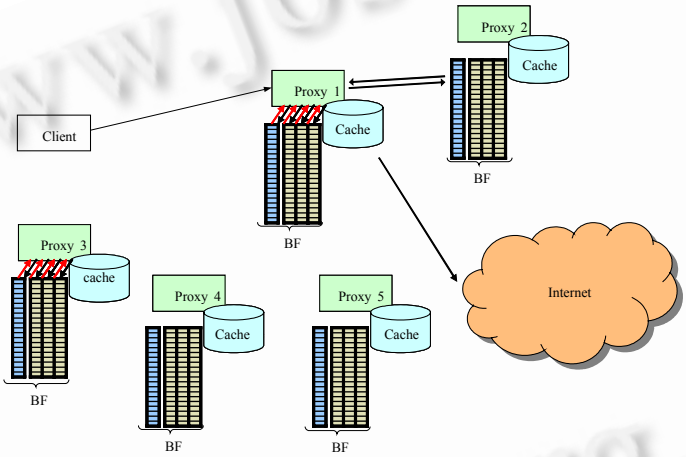

Fig.9 Application of Bloom filter in distributed systems

图 9 布鲁姆过滤器在分布式系统中应用

3.2 计算机网络中的主要应用

布鲁姆过滤器自 1970 年被提出以来,被广泛应用到各种计算机系统之中,以表达庞大数据集合、提高查找效率.最初的应用范围很窄,主要集中在数据库操作、字典查询和文件操作[5−9],直到最近 10 年[3,10],随着网络研究的发展以及新的覆盖网和 P2P 网络应用技术的涌现,出现了布鲁姆过滤器查询算法研究的高潮,应用于网络中的领域和方式越来越多,主要体现在:

- 覆盖网和 P2P 网节点协作交互(P2P/overlay networks)[11−13]:网络节点利用布鲁姆过滤器存储节点的内容摘要.节点间通过互传布鲁姆过滤器表示的摘要信息获得其他节点的存储内容信息,借助自身节点或者上级节点存储的其他节点的资源内容摘要,无须广播资源请求,完成资源的搜索和定位.节约节点间带宽资源,优化 P2P 网络的资源交互和查找.
- 资源路由(resource routing)[14]:利用布鲁姆过滤器记录邻接节点组或者节点生成树上的资源信息,通过

一个多级的布鲁姆过滤器结构存储资源的路由分级信息,快速定位最短路径路由并确定到达目的节点的跳数.

- 封包路由(pacekt routing)[15]:又名数据帧路由标签,通过在 IP 报文字段中添加用布鲁姆过滤器表示的所通过的路由器,来判断数据帧是否重复经过路由器,从而确定是否出现环路,弥补了传统网络设计中仅通过 *TTL* 控制路由环路的缺陷.
- 网络测量方法[16−23]:利用布鲁姆过滤器为网络业务流计数,当流量超过阈值时产生告警并进行拥塞控制[16,17];在进行 IP 数据包回溯时,通过在路由器中用布鲁姆过滤器将已经转发过的数据报文记录下来[18,19],只要查找该数据包是否属于转发记录中的数据包即可完成数据包的回溯检查;布鲁姆过滤器还广泛应用于网络测量中,包括:网络数据包分类[20,24]、网络抽样[21]、抽样的还原[22]以及流分布估计[23]等.
- 分布式网络测量系统设计[50]:在分布式协作的网络测量中,同一高速链路的监测由多个监测探针共同完成,利用布鲁姆过滤器完成不同业务流监测的任务分配管理,各个探针负责监测不同的网络业务流.

此外,布鲁姆过滤器在网络入侵检测(intrusion detection)[25−27]、Ad hoc 网络中信息共享[28]和传感器网络数据过滤和路由[29,30]等网络应用领域得到越来越广泛的应用.

布鲁姆过滤器在网络中的应用有愈演愈烈的趋势,布鲁姆过滤器节约空间的特点是其最近在网络中广泛应用的根本原因.因为网络的基本功能是资源的交互共享,在网络中经常会出现需要表示和传输资源集合或者信息链表的情况,如果直接表示或传输此类交换信息,可能会占用大量的存储资源或者带宽资源.布鲁姆过滤器成为网络协议和应用系统设计的首选,用以传递信息,节约带宽.这种优秀、简洁的查询算法仅仅以引入一定的查询误判率为代价,只要这种少量的误判率在系统可接受的范围内,则布鲁姆过滤器的引入将极大地增强系统的性能.

而且,我们发现,即使在半导体技术和网络技术高速发展的今天,虽然存储设备容量和网络带宽都得到了长足的进步,但是布鲁姆过滤器查询算法仍然在网络传输带宽优化[31]、高速缓存设计[51]和超大规模海量集合存储需求[19]上具有较大的使用价值.

人们在多次成功应用布鲁姆过滤器之后,得出布鲁姆过滤器的使用原则[3]:当用受限的空间完成集合或链表的查询时,就应该考虑使用布鲁姆过滤器.

4 总结与研究展望

虽然上述算法研究在一定程度上扩大了算法的内涵,但是,由于布鲁姆过滤器算法发展前期受到的关注度不够,算法还存在大量的扩展和改进空间;而且,当前网络快速演进发展的趋势,也给布鲁姆过滤器算法带来了新的发展契机和理论以及应用改进需求,主要体现在:

(a) 集合多样性带来了布鲁姆过滤器所支持的集合信息结构扩展需求

多样性的网络应用带来了多样性的操作集合,多样性操作集合迫切需要扩展布鲁姆过滤器所支持的集合形式,以扩展布鲁姆过滤器的应用领域和应用方式.布鲁姆过滤器作为一种表示集合的数据结构,集合是布鲁姆过滤器算法实施的对象和前提.但是,现有的布鲁姆过滤器对集合的表示大多限制为数字或者类似数字组成的集合,即将非数值集合的元素转换成一定范围的数值数据.这种对数据集合形式的限制,成为布鲁姆过滤器在持续发展的网络中广泛应用的绊脚石.因此,扩展布鲁姆过滤器对不同集合的表示能力,突破对数值集合的限制,推广布鲁姆过滤器算法用以表达非数值集合,成为进一步扩大布鲁姆过滤器应用范围极富挑战的工作.

(b) 流数据处理中对布鲁姆过滤器时间纬度扩展的需求

布鲁姆过滤器在处理传统的静态数据集合上已经得到了广泛的关注并取得了较多的研究成果,但是,布鲁姆过滤器对动态实时集合的处理还相当受限.随着网络快速朝着高速、无线、移动方向的演进,大量的数据如传感器探测数据、网络监测数据、金融交易、股票数据等都以流数据这种动态数据形式存在,对流数据处理和操作已经成为数据管理的热点.流数据与一般数据的区别是数据持续、快速到达、规模宏大,具有时变性和不可预测性.布鲁姆过滤器在数据处理上最大的优势就是可以用远小于数据规模的存储空间支持快速、高效的查询,能够满足动态流数据处理中概要数据结构设计的需求.但是,目前布鲁姆过滤器还没有可以用于表示流数据

的顺序结构和时间结构,无法直接用于时间相关动态流数据的处理.因此,研究布鲁姆过滤器算法的时间维度扩展,支持动态流数据处理,降低流数据处理的存储空间和查询开销,成为布鲁姆过滤器新的研究点,为流数据处理提供新的解决方案.

(c) 多集合下的布鲁姆过滤器位串序列运算研究和扩展需求

复杂应用的发展使得传统的针对单集合处理的布鲁姆过滤器不能解决应用中多集合存在逻辑关系的场合.数据库作为表示和存储集合的数据结构,在数据库中研究集合的关系代数,探讨数据库代数运算下的多集合查询关系,该领域已有不少研究成果.布鲁姆过滤器作为一种表示集合的 0,1 位串数据结构,它的位串序列运算是否与集合的运算存在某种潜在的关系或规律?布鲁姆过滤器相关的运算方法或者其他泛函运算的研究是否一样可以优化多集合的查询操作?如果能够找出布鲁姆过滤器集合间的某种运算关系,将极大地优化多集合查询操作.因此,研究多集合扩展下的布鲁姆过滤器查询算法,从序列运算出发,减少多集合表示和查询的哈希运算次数,缩短计算时间,简化应用操作,成为布鲁姆过滤器领域尚未解决的有趣问题.

(d) 布鲁姆过滤器在高速网络、无线网络推进过程中新的应用和优化

随着网络向着大规模、高速、无线网络方向发展,传统的网络数据处理技术不再适应网络应用发展需求,迫切需要对现有的数据处理技术进行变革,给布鲁姆过滤器算法的发展带来了巨大的应用空间.以分布式网络监测为例,在高速网络中,产生了大量的网络测量数据和网络通信开销,存储、传输、处理如此海量的网络监测数据对传统的监控系统结构并非易事,线速网络数据处理需要快速、简洁的结构来存储、传输、处理海量数据;又以 3GPP 为例,随着无线网络向大规模分布式的全面演进,分布式基站和 AP 节点以及大规模终端集合的出现,使构建大规模无线网络同样迫切需要处理大规模数据集合.因此,设计简洁的结构存储网络信息,减少信息交互和查询占用的网络带宽,提升系统的数据管理能力,成为高速网络发展和大规模异构无线网络发展中的又一大难题.

总之,布鲁姆过滤器近年来的研究和应用发展速度迅猛,布鲁姆过滤器空间简洁的特性使其在分布式系统查询交互上具有天然的优势.本文围绕布鲁姆过滤器的原理、扩展和应用进行探讨,对算法现有的主要研究成果进行汇总和分析,展望可能的研究方向,希望以此作为引子,吸引更多的人研究布鲁姆过滤器这一年轻的算法.

References:

[1] Burton HB. Space/Time trade-offs in hash coding with allowable errors. Communications of the ACM, 1970,13(7):422−426.

[2] Mitzenmacher M. Compressed Bloom filters. IEEE-ACM Trans. on Networking, 2002,10(5):604−612.

[3] Broder A, Mitzenmacher M. Network applications of Bloom filters: A survey. Internet Mathematics, 2002,1(4):636−646.

[4] Kirsch A, Mitzenmacher M. Simple summaries for hashing with choices. IEEE/ACM Trans. on Networking, 2008,16(1):218−231.

[5] Mullin JK. Optimal semijoins for distributed database systems. IEEE Trans. on Software Engineering, 1990,16(5):558−560.

[6] McIlroy M. Development of a spelling list. IEEE Trans. on Communications, 1982,30(1):91−99.

[7] Udi M, Sun W. An algorithm for approximate membership checking with application to password security. Information Processing Letters, 1994,50(4):191−197.

[8] Gremillion LL. Designing a Bloom filter for differential file access. Communications of the ACM, 1982,25(9):600−604.

[9] James KM. A second look at Bloom filters. Communications of the ACM, 1983,26(8):570−571.

[10] Flavio B, Michael M, Rina P, Sushil S, George V. Beyond Bloom filters: From approximate membership checks to approximate state machines. Computer Communication Review, 2006,36:315−326.

[11] Jonathan L, Jacob MT, Laura S, Margo S. Self-Organization in peer-to-peer systems. In: Proc. of the 10th Workshop on ACM SIGOPS European Workshop. Saint-Emilion: ACM, 2002. 125−132.

[12] Cuenca-Acuna FM, Peery C, Martin RP, Nguyen TD. PlanetP: Using gossiping to build content addressable peer-to-peer information sharing communities. In: Proc. of the 12th IEEE Int'l Symp. on High Performance Distributed Computing (HPDC 2003). Washington: IEEE Computer Society, 2003. 236−246.

[13] Kun X, Dafang Z, Gaogang X, Jigang W. A trace label based consistency maintenance algorithm in unstructured P2P systems. Journal of Software, 2007,18(1):105−116 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/18/105.htm

[14] Rhea SC, Kubiatowicz J. Probabilistic location and routing. In: Proc. of the INFOCOM 2002, Vol.3. Washington: IEEE Computer Society, 2002. 1248−1257.

[15] Whitaker A, Wetherall D. Forwarding without loops in Icarus. In: Proc. of the IEEE Openarch 2002. Washington: IEEE Computer Society, 2002. 63−75.

[16] Cristian E, George V. New directions in traffic measurement and accounting. SIGCOMM Computer Communication Review, 2002,32(4):323−336.

[17] Sarang D, Praveen K, David ET. Longest prefix matching using Bloom filters. In: Proc. of the Sigcomm 2003. Karlsruhe: ACM, 2003. 201−212.

[18] Alex CS. Hash-Based IP traceback. SIGCOMM Computer Communication Review, 2001,1(4):3−14.

[19] Alex CS, Craig P, Luis AS, Christine EJ, Fabrice T, Beverly S, Stephen TK, Strayer WT. Single-Packet IP traceback. IEEE/ACM Trans. on Networking, 2002,10(6):721−734.

[20] Sarang D, Haoyu S, Jonathan T, John L. Fast packet classification using Bloom filters. In: Proc. of the 2006 ACM/IEEE Symp. Architecture for Networking and Communications Systems. San Jose: ACM, 2006. 61−70.

[21] Kumar A, Jimxu J, Wang J. Space-Code Bloom filter for efficient per-flow traffic measurement. IEEE Journal on Selected Areas in Communications, 2006,24(12):2327−2339.

[22] Nicolas H, Darryl V. Inverting sampled traffic. IEEE/ACM Trans. on Networking, 2006,14(1):68−80.

[23] Abhishek K, Minho S, Jun X, Jia W. Data streaming algorithms for efficient and accurate estimation of flow size distribution. ACM SIGMETRICS Performance Evaluation Review, 2004,32(1):177−188.

[24] Heeyeol Y, Mahapatra R. A memory-efficient hashing by multi-predicate Bloom filters for packet classification. In: Proc. of the INFOCOM 2008. Washington: IEEE Computer Society, 2008. 1795−1803.

[25] Suresh D, Guo Z, Buyukkurt B, Najjar W. Automatic compilation framework for Bloom filter based intrusion detection. In Proc. of the ARC 2006. 2006. 413−418. http://www.cs.ucr.edu/~roccc/papers/ARC-2006-Bloom.pdf

[26] Locasto ME, Parekh JJ, Keromytis AD, Stolfo SJ. Towards collaborative security and P2P intrusion detection. In: Proc. of the 2005 IEEE Workshop on Information Assureance and Security. Washington: IEEE Computer Society, 2005. 333−339.

[27] Chen W, He YX, Peng WL. A light weight detection method against DDoS Attack. Chinese Journal of Computers, 2006,29(8): 1392−1400 (in Chinese with English abstract).

[28] Danyu Z, Mutka M. Sharing presence information and message notification in an ad hoc network. In: Proc. of the 1st IEEE Int'l Conf. on Pervasive Computing and Communications, 2003. Washington: IEEE Computer Society, 2003. 351−358.

[29] Ye F, Luo HY, Lu SW, Zhang LX. Statistical en-route filtering of injected false data in sensor networks. In: Proc. of the INFOCOM 2004, Vol.4. Washington: IEEE Computer Society, 2004. 2446−2457.

[30] Hebden P, Pearce AR. Data-Centric routing using Bloom filters in wireless sensor networks. In: Proc. of the 4th Int'l Conf. on Intelligent Sensing and Information. Washington: IEEE Computer Society, 2006. 72−77.

[31] Fan L, Cao P, Almeida J, Broder AZ. Summary cache: A scalable wide-area Web cache sharing protocol. IEEE-ACM Trans. on Networking, 2000,8(3):281−293.

[32] Flavio B, Michael M, Rina P, Sushil S, George V. An improved construction for counting Bloom filters. In: Proc. of the 14th Conf. on Annual European Symp., Vol.14. Zurich: Springer-Verlag, 2006. 684−695.

[33] Saar C, Yossi M. Spectral Bloom filters. In: Proc. of the 2003 ACM SIGMOD Int'l Conf. on Management of Data. San Diego: ACM, 2003. 241−252.

[34] Xiao MZ, Dai YF, Li XM. Split Bloom filter. Acta Electronica Sinica, 2004,32(2):241−245 (in Chinese with English abstract).

[35] Guo D, Wu J, Chen H, Luo X. Theory and network applications of dynamic Bloom filters. In: Proc. of the INFOCOM 2006. Washington: IEEE Computer Society, 2006. 1−12.

[36] Kun X, Yinghua M, Dafang Z, Jigang W, Gaogang X. A scalable Bloom filter for membership queries. In: Proc. of the GLOBECOM 2007. Washington: IEEE Computer Society, 2007. 543−547.

[37] Xie K, Qin Z, Wen JG, Zhang DF, Xie GG. Combine multi-dimension Bloom filter for membership queries. Journal on Communications, 2008,29(1):56−64 (in Chinese with English abstract).

[38] Xie K, Ming YH , Zhang DF, Xie GG, Wen JG. Basket Bloom filters for membership queries. Chinese Journal of Computers, 2007,30(4):597−607 (in Chinese with English abstract).

[39] Wang JC, Xiao MZ, Dai YF. MBF: A real matrix Bloom filter representation method on dynamic set. In: Proc. of the 2007 IFIP Int'l Conf. on Network and Parallel Computing Workshops. Washington: IEEE Computer Society, 2007. 733−736.

[40] Ficara D, Giordano S, Procissi G, Vitucci F. MultiLayer compressed counting Bloom filters. In: Proc. of the INFOCOM 2008. Washington: IEEE Computer Society, 2008. 311−315.

[41] Kirsch A, Mitzenmacher M. Distance-Sensitive Bloom filters. In: Proc. of the Algorithm Engineering and Experiments (ALENEX). 2006. http://www.siam.org/proceedings/alenex/2006/alx06_004akirsch.pdf

[42] Lumetta S, Mitzenmacher M. Using the power of two choices to improve Bloom filters. http://www.eecs.harvard.edu/~michaelm.

[43] Fang H, Murali K, Lakshman TV. Building high accuracy Bloom filters using partitioned hashing. ACM SIGMETRICS Performance Evaluation Review, 2007,35(1):277−288.

[44] Fan D, Davood R. Approximately detecting duplicates for streaming data using stable Bloom filters. In: Proc. of the 2006 ACM SIGMOD Int'l Conf. on Management of Data. Chicago: ACM, 2006. 25−36.

[45] Adam K, Michael M. Less hashing, same performance: Building a better Bloom filter. Random Structures & Algorithms, 2008,33(2):187−218.

[46] Gong J, Peng YB, Yang W, Liu WJ. Reconstructing the parameter for massive abnormal TCP connections with Bloom filter. Journal of Software, 2006,17(3):434−444 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/17/434.htm

[47] Peng YB, Gong J, Liu WJ, Yang W. Element recovery from counting Bloom filters' hash space. Acta ElectronicA Sinica, 2006,34(5):822−827 (in Chinese with English abstract).

[48] Xie K, Zhang DF, Wen JG, Xie GG, You ZQ. Algebraic operations on Bloom filters. Acta ElectronicA Sinica, 2008,36(5):869−874 (in Chinese with English abstract).

[49] Rousskov A, Wessels D. Cache digests. Computer Networks and ISDN Systems, 1998,30(22-23):2155−2168.

[50] Sharma MR, Byers JW. Scalable coordination techniques for distributed network monitoring. In: Proc. of the Passive and Active Network Measurement. 2005. 349−352. http://www.pamconf.org/2005/PDF/34310352.pdf

[51] Sarang D, Praveen K, Sproull TS, Lockwood JW. Deep packet inspection using parallel Bloom filters. Micro IEEE, 2004,24(1): 52−61.

附中文参考文献:

[13] 谢鲲,张大方,谢高岗,文吉刚.基于轨迹标签的无结构 P2P 副本一致性维护算法.软件学报,2007,18(1):105−116. http://www.jos.org.cn/1000-9825/18/105.htm

[27] 陈伟,何炎祥,彭文灵.一种轻量级的拒绝服务攻击检测方法.计算机学报,2006,29(8):1392−1400.

[34] 肖明忠,代亚非,李晓明.拆分型 Bloom filter.电子学报,2004,32(2):241−245.

[37] 谢鲲,秦拯,文吉刚,张大方,谢高岗.联合多维布鲁姆过滤器查询算法.通信学报,2008,29(1):56−64.

[38] 谢鲲,闵应骅,张大方,谢高岗,文吉刚.基于分档布鲁姆过滤器的查询算法.计算机学报,2007,30(4):597−607.

[46] 龚俭,彭艳兵,杨望,刘卫江.基于 BloomFilter 的大规模异常 TCP 连接参数再现方法.软件学报,2006,17(3):434−444. http://www.jos.org.cn/1000-9825/17/434.htm

[47] 彭艳兵,龚俭,刘卫江,杨望.Bloom filter 哈希空间的元素还原.电子学报,2006,34(5):822−827.

[48] 谢鲲,张大方,文吉刚,谢高岗,尤志强.布鲁姆过滤器布尔代数探讨.电子学报,2008,36(5):869−874.

谢鲲(1978－),女,湖南黔阳人,博士,副教授,主要研究领域为可信系统与网络,无线网络,算法设计.

张大方(1959－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为可信系统与网络,网络测试,软件容错,软件测试.

文吉刚(1978－),男,博士生,主要研究领域为分布式计算,P2P 流媒体,无线网络.

谢高岗(1974－),男,博士,副研究员,CCF 高级会员,主要研究领域为网络测量,分析与模型化,P2P 计算.