几何布鲁姆过滤器的设计与分析

张 震,汪斌强,陈庶樵,郭 通
(国家数字交换系统工程技术研究中心,河南郑州 450002)

摘 要: 针对经典计数型布鲁姆过滤器(NCBF)存储和查询性能较低的缺陷,提出了几何布鲁姆过滤器结构 GBF.该结构通过引入"哈希指纹"、布鲁姆过滤器两次分割、基于桶负载存放的方法,实现了集合元素的简洁存储、快速查询.基于"微分方程"和"概率论"的相关知识,对 GBF 模型进行了理论分析和求解,建立了错误概率和计算复杂度的关系表达式,论证了 GBF 的几何分布特性.仿真结果表明:与 NCBF 相比,GBF 具有较低错误概率和计算复杂度的同时,也能保持较高的空间利用率.

关键词: 布鲁姆过滤器; 几何布鲁姆过滤器; 概要数据结构
中图分类号: TP393 **文献标识码:** A **文章编号:** 0372-2112 (2012)09-1852-06
电子学报 URL: http://www.ejournal.org.cn **DOI:** 10.3969/j.issn.0372-2112.2012.09.023

Geometric Bloom Filter Designing and Its Analysis

ZHANG Zhen, WANG Bin-qiang, CHEN Shu-qiao, GUO Tong
(*National Digital Switching System Engineering & Technological R&D Center, Zhengzhou, Henan* 450002, *China*)

Abstract: Considering the poor storage and query performances of naïve counting Bloom filter (NCBF), a data structure called geometric Bloom filter (GBF) is presented. In order to achieve space-efficient storage and fast query, the structure introduces the idea of hash fingerprints, partitions Bloom filter twice and stores elements with buckets. Based on theory of differential equation and probability, analytical expressions of GBF are deduced. The relational expressions between error probability and space complexity are also established. Furthermore, the inner characteristic of GBF taking on geometric distribution is proved. Simulated results indicate that GBF can achieve lower error probability and computational complexity without sacrificing accuracy compared with NCBF.

Key words: Bloom filter; geometric Bloom filter; synopsis data structure

1 引言

随着互联网规模的不断扩大、用户数量的快速膨胀、业务环境的日趋复杂,共享数据规模呈现几何增长,给网络资源的存储、访问、以及交互带来了巨大的挑战.如何快速表示和查询大规模数据、有效提取和存储内容概要信息是网络和分布式系统资源共享的关键,也一直是国内外学术界的研究热点.Bloom filter 是一种空间高度复用的概要数据结构,能够满足高速网络发展中的资源交互需求.与传统的存储和查询算法(如:树型查询算法)相比较,Bloom filter 所需空间与元素的自身大小无关,仅与元素映射到的向量位数有关,大大节约了存储空间.同时,该结构还具有计算复杂度低、并行程度高等优势,使之特别适合于硬件实现.因此,Bloom filter 具有很好的实用价值.

自 1970 年 Bloom filter 问世以来[1],被广泛应用于各种计算机系统中,如数据库操作、字典查询等[2].近年来,随着覆盖网和 P2P 网络的广泛应用,掀起了布鲁姆过滤器算法在网络领域的研究高潮,包括 P2P 网络节点协作交互、网络流量测量以及资源路由等[3~6].目前,对 Bloom filter 自身结构的研究也有了重大进展.文献[7]提出了"计数型"布鲁姆过滤器(Naïve Counting Bloom Filter, NCBF).NCBF 是布鲁姆过滤器的一种重要的扩展结构,能够支持集合元素的动态插入、动态删除以及近似计数查询.针对分布式 Web Cache 信息共享的网络应用,文献[8]提出了"压缩型"布鲁姆过滤器(Compressed Bloom Filter, CBF)数据结构.CBF 基于"信息压缩"理论,通过引入算术编码技术,能够有效地利用空间存储资源.文献[9]提出了"光谱型"布鲁姆过滤器(Spectral Bloom Filter, SBF),用元素对应的 k 个 Counter 计数器中

收稿日期:2011-09-14;修回日期:2012-05-07
基金项目:国家重点基础研究发展规划(973 计划)项目(No.2012CB312901, No.2012CB312905);国家高技术研究发展计划(863 计划)课题(No.2011AA01A103)

最小的计数器作为元素出现的频度估计,从而能够过滤掉频度小于一定阈值的元素.为了考察布鲁姆过滤器的查询失效代价,文献[10]提出了一种"分档型"布鲁姆过滤器(Basket Bloom Filter,BBF).BBF 结构采用"区分服务"的思想,将集合元素分成 L 档,并通过对高代价的子集合分配较多哈希函数,降低查询失效率;对低代价子集合分配少量的哈希函数,适当增加查询失效率,使集合查询失效总代价最小.

SBF、CBF 以及 BBF 都是针对不同的应用,对 NCBF 进行了改进,提升了 Bloom filter 的查询性能.但是,NCBF 的计算复杂度依赖于哈希函数的个数 k,即使其存储集合元素的个数为空,更新每个元素也要访问 k 次存储器,系统开销还有降低的空间;SBF 和 CBF 算法复杂,简单可用性较差,比 NCBF 更加难于硬件实现[11];BBF 试图根据查询失效率,动态改变哈希函数的个数,但是基于遗传算法的求解比较复杂,不适宜于计算敏感型的应用场合.本文立足于错误概率、计算复杂度以及空间利用率三项性能指标,针对传统的 NCBF 结构,提出了一种新的几何布鲁姆过滤器结构(Geometric Bloom Filter,GBF).

2　Bloom filter 和 NCBF 结构描述

Bloom filter 的核心是一个 $\boldsymbol{V}$ 向量和一组哈希函数,设集合 $S=\{s_1,s_2,\cdots,s_n\}$ 共有 n 个元素,通过 k 个哈希函数 $h_1,h_2,\cdots,h_k$ 映射到长度为 c 的向量 $\boldsymbol{V}$ 中.每个哈希函数相互独立且取值范围为 $\{0,1,\cdots,c-1\}$.集合到向量 $\boldsymbol{V}$ 的映射过程如下:将向量 $\boldsymbol{V}$ 所有比特初始化为 0;当元素 s_i 插入集合 S 时,计算 $h_j(s_i)(j=1,\cdots,k)$,若 $h_j(s_i)=q$,则令 $\mathrm{BF}[q]=1$;当查询元素 x 是否属于集合 S 时,检查向量 $\boldsymbol{V}$ 的 k 个位置 $(h_1(x),h_2(x),\cdots,h_k(x))$ 是否为 1,如果其中有一个为 0,则判定 $x\notin S$;若全部值为 1,则 x 可能属于 S.Bloom filter 在查询时会出现"假阳性误判",即将不属于集合的元素误判为属于该集合.根据文献[1],误判概率满足如下等式:

$$f^{\mathrm{BF}}(n,c,k)=(1-\mathrm{e}^{-kn/c})^k=\exp(k\ln(1-\mathrm{e}^{-nk/c})) \tag{1}$$

令 $g(k)=k\ln(1-\mathrm{e}^{-nk/c})$,当 $\partial g(k)/\partial k=0$ 时,可得最优哈希函数个数和最小误判概率:$k_{\mathrm{opt}}=\lceil \ln 2\cdot(c/n)\rceil$ 和 $f^{\mathrm{BF}}_{\min}(n,c,k)=f^{\mathrm{BF}}_{\min}(n,c,k_{\mathrm{opt}})=(1/2)^{k_{\mathrm{opt}}}$.

Bloom filter 能够支持集合元素的插入和散列查询,但不能支持元素的删除.NCBF 可以解决这一问题:将向量 $\boldsymbol{V}$ 的每一维 $i(i=1,\cdots,c)$ 设置成一个计数器,初始值为 0;当要增加集合元素 x 时,令 $c(h_j(x))=c(h_j(x))+1(j=1,\cdots,k)$;当要删除集合元素 x 时,令 $c(h_j(x))=c(h_j(x))-1(j=1,\cdots,k)$.文献[12]证明了:当一组无重复的元素插入到 NCBF 中时,每计数器用 4bit 就可以保证其溢出概率满足 $P(\max(c(i))\geqslant 16)\leqslant 1.37\times 10^{-15}c$.根据式(1),布鲁姆过滤器的误判概率只和 $\{n,c,k\}$ 有关,与每个计数器的大小无关.因此,NCBF 的"假阳性"错误概率和 Bloom filter 相等,即 $f^{\mathrm{NCBF}}(n,c,k)=f^{\mathrm{BF}}(n,c,k)$.

虽然 NCBF 支持元素的动态删除,但是 NCBF 计数器向量 $\boldsymbol{V}$ 的每个计数器需要根据最大可能的元素频率值设置其计数范围,因而 NCBF 的空间效率较低.同时,在实际应用中,需要谨慎选择 NCBF 计数器的长度:过大的计数器会使 NCBF 所需的存储空间成倍增加;而过小的计数器容易造成计数器的溢出.

3　GBF 模型构建

3.1　GBF 结构描述

如图 1 所示,GBF 由 k 维子布鲁姆过滤器 $T_1,T_2,\cdots,T_j,\cdots,T_k$ 组成,每维子布鲁姆过滤器 T_j 包含若干个桶,每个桶由相同数量的存储单元组成(每桶深度为 h),集合元素存放在桶单元中.不妨设哈希空间 $H=\{h_j,j=1,\cdots,k\}$,每个哈希函数 h_j 对应子布鲁姆过滤器 T_j,每个子布鲁姆过滤器 T_j 对应 $\alpha_j m$ 个桶,其中 $\sum\alpha_j=1$.第 j 个子布鲁姆过滤器 T_j 对应的桶向量 BV[j]即为 $(B^j_1,B^j_2,\cdots,B^j_{\alpha_j m})$.另外,GBF 引入了"哈希指纹"(Hash Fingerprint)的概念,即每个桶的存储单元值是由对应哈希函数计算得到的,其内容主要包括桶索引(Bucket Index,BI)、单元索引(Cell Index,CI)和元素标识(Element Identifier,EI).桶索引和单元索引分别用于快速定位相应的桶和存储单元,元素标识用来区分每个集合元素.因此,对于元素 $y\notin S$,当且仅当存在元素 $x\in S$,使得 $H(y)=H(x)$ 成立时,GBF 才会产生误判.

表 1 和表 2 用伪码给出了 GBF 元素的插入和查询流程,删除过程和查询过程类似.其中最为重要的步骤是:GBF 顺序计算集合元素的哈希值——"BI + CI + EI",直到找到负载未满的桶,并将元素值置于该存储单

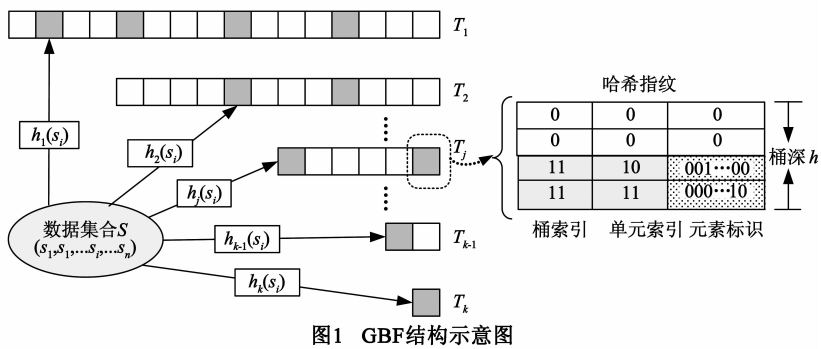

图1　GBF结构示意图

元中.如果某元素对应的 k 个桶都是满负载的,就会出现“溢出”现象.

表 1 GBF 元素插入流程

Algorithm 1 Insert Procedure of GBF

```
Input: element x
let i = 1
repeat
    compute hash value h_i(x) = BI + CI + EI
    if sizeof(Bucket[i][BI]) < h then
        create cell in Bucket[i][BI]
        insert "BI + CI + EI" into the cell
        return i
    else i = i + 1
until i = k
overflow error "Element is discarded"
```

表 2 GBF 元素查询流程

Algorithm 2 Search Procedure of GBF

```
Input: element x
let i = 1
repeat
    compute hash value h_i(x) = BI + CI + EI
    search the corresponding cell EI' in
        Bucket[i][BI]
    if EI = EI' then
        return i
    else i = i + 1
until i = k
error "Unsuccessful search"
```

3.2 性能指标和相关定义

针对 GBF 的性能评价,特定义了以下三项指标:

定义 1 错误概率 e:对于任意的 $x\in S, y\notin S$,当且仅当 $H(y)=H(x)$时,会产生“误判概率”f^{GBF};当集合元素 x 插入时,若 sizeof($h_1(x)$),…,sizeof($h_k(x)$)全部等于 h(即对应的 k 个桶都已满载),则元素 x 将被丢弃,此现象发生的概率为“溢出概率”γ^{GBF}.错误概率即为 $e=f^{\mathrm{GBF}}+\gamma^{\mathrm{GBF}}$.

定义 2 计算复杂度 ξ:由于新到达的数据可能会触发更新或者查询存储器的操作.GBF 的计算复杂度 ξ 定义为每个元素插入时,平均访问存储单元的次数.

定义 3 空间利用率 η:每元素占用的空间大小.GBF 为了提高空间利用率,引入了“哈希指纹”的概念,用有效的几个比特来标识结合元素.

为了方便对 GBF 模型进行刻画和描述,特进行了以下定义:

定义 4 简单 GBF:GBF 是由 k 维子布鲁姆过滤器组成,每一维子布鲁姆过滤器定义为简单几何布鲁姆过滤器(Naïve Geometric Bloom Filter,Naïve-GBF).

定义 5 访问时间:定义访问时间 $t=i/n$ ($0\leqslant t\leqslant 1$)表示第 i 个元素访问的时刻,$t=1$ 表示所有的 n 个元素都已经插入完毕.对 GBF 而言,“访问存储单元的次数”区别于“元素插入的次数”,这是因为 GBF 一次元素的插入有可能会引起多次对存储单元的访问.

4 理论分析与求解

4.1 Naïve-GBF 模型分析

由于 GBF 由若干 Naïve-GBF 组成,下面首先对 Naïve-GBF 进行数学建模.如图 2 所示,Naïve-GBF 由若干个等深的桶组成,集合元素通过哈希函数 h_j 映射到各个桶中.令 $m_j=\alpha_j m$,并假设 h_j 在[1,m_j]上服从均匀分布.如果某集合元素对应的桶已满负载,则该元素将被丢弃;否则,插入该桶.如下图,由于元素 x 对应的桶已满,元素 x 将被丢弃;而元素 y 则被插入相应的位置.

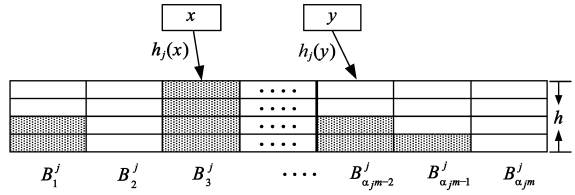

图2 Naïve-GBF结构示意图

初始时,Naïve-GBF 是空载的,之后,Naïve-GBF 顺序插入 n_j 个元素.设 $F_x(i/n_j)$表示第 i 个元素插入时,存储 x 个元素的桶在 Naïve-GBF 所有桶中所占的比例,则 $\sum_{x=0}^{h}F_x(i/n_j)=1$.并令 $\Delta F_x((i+1)/n_j)=F_x((i+1)/n_j)-F_x(i/n_j)$表示在[$i/n_j$,$(i+1)/n_j$]时间窗内,含有 x 个元素的桶比例的变化量.当时刻 $t=i/n_j$ 时,不难分析桶负载的变化规律:

(1)若 $x=0$,仅当下一时刻$(i+1)/n_j$,某元素均匀插入到“零负载”的桶时,$F_0(i/n_j)$才会减小;其他情况均不会变化;

(2)若 $x=h$,仅当下一时刻$(i+1)/n_j$,某元素均匀插入到桶负载等于 $h-1$ 时,$F_{h-1}(i/n_j)$才会增加;其他情况均不会变化;

(3)若 $x\in(1,h)$,仅当下一时刻$(i+1)/n_j$,某元素均匀插入到桶负载等于 x 或者等于 $x-1$ 时,$F_x(i/n_j)$才会相应减少或增加.

由此,可得到如下表达式:

$$E(\Delta F_x((i+1)/n_j)\mid F_x(i/n_j))=\begin{cases}-1/m_j\cdot F_0(i/n_j), & x=0\\ 1/m_j\cdot(F_{x-1}(i/n_j)-F_x(i/n_j)), & x\in(1,h)\\ 1/m_j\cdot F_{h-1}(i/n_j), & x=h\end{cases}\tag{2}$$

上式满足初始条件:$F_0(0)=1$ 且 $F_i(0)=0(i\neq 0)$.两边同时除以 $1/n_j$,进一步得到:

$$\frac{E(\Delta F_x((i+1)/n_j)\mid F_x(i/n_j))}{1/n_j}$$

$$=\begin{cases}-n_j/m_j\cdot F_0(i/n_j), x=0\\ n_j/m_j\cdot(F_{x-1}(i/n_j)-F_x(i/n_j)), x\in(1,h)\\ n_j/m_j\cdot F_{h-1}(i/n_j), x=h\end{cases}\tag{3}$$

当 $n_j\to+\infty$ 时,$1/n_j\to 0$,并用 t 替换 i/n_j,可得到如下微分表达式[13,14]:

$$\frac{\mathrm{d}F_x(t)}{\mathrm{d}t}=\begin{cases}-n_j/m_j\cdot F_0(t), x=0\\ n_j/m_j\cdot(F_{x-1}(t)-F_x(t)), x\in(1,h)\\ n_j/m_j\cdot F_{h-1}(t), x=h\end{cases}\tag{4}$$

基于式(4),容易得到 $F_0(t)=\mathrm{e}^{-m_j/n_j\cdot t}$.通过迭代法,可以进一步得到 $F_x(t)$的解析表达式:

$$F_x(t)=\begin{cases}\dfrac{1}{x!}(n_jt/m_j)^x\mathrm{e}^{-n_jt/m_j}, x\in[0,h)\\ 1-\displaystyle\sum_{k=0}^{h-1}\frac{1}{k!}(n_jt/m_j)^k\mathrm{e}^{-n_jt/m_j}, x=h\end{cases}\tag{5}$$

从式(5)可以看出,$F_x(t)$服从参数 $\lambda=n_jt/m_j$ 的泊松分布.主要原因是:对与给定的时间 t,当 n_j 趋于非常大时,参数为$(n_jt,1/m)$的二项分布泊松分布近似为参数 $\lambda=n_jt/m_j$ 的泊松分布[15].根据式(5),可推导出 t 时刻溢出元素的平均个数 $n_jt-m_j\sum_{x=0}^{h}x\cdot F_x(t)$,除以 n_jt 后得到 Naïve-GBF 的溢出概率表达式如下:

$$\gamma^{\mathrm{Naive}}(t)=1-\frac{m_j}{n_jt}\sum_{x=0}^{h-1}x\cdot\frac{1}{x!}(n_jt/m_j)^x\mathrm{e}^{-n_jt/m_j}-\frac{m_jh}{n_jt}\left(1-\sum_{k=0}^{h-1}\frac{1}{k!}(n_jt/m_j)^k\mathrm{e}^{-n_jt/m_j}\right)\tag{6}$$

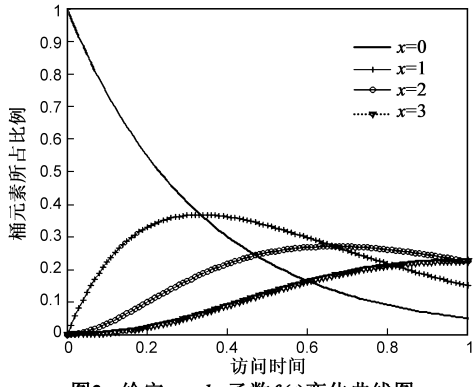

图3　给定n,m,h, 函数$f_x(t)$变化曲线图

图 3 在给定 $n=m$,$h=3$ 的情况下,分别对函数 $f_i(t)(i=0,1,2,3)$进行了仿真;图 4 在 $n/m=1$ 的情况下,分别针对不同的 h 对溢出概率进行了仿真.图 3 表明:由于元素的连续插入,存储元素数量为 0 的桶所占比例逐渐下降;存储元素数量较少的桶所占比例增幅相对较快,并随着时间的推移逐渐减少.从图 4 可以看出:n/m 一定的情况下,随着 h 的增加,溢出概率逐渐减小;给定 h,随着 n/m 的增大,溢出概率也逐渐变大.

4.2　GBF 模型求解

令 $l_j(t)$表示 t 时刻访问 T_{j-1}后溢出元素的个数,基于 GBF 的插入流程,$l_j(t)$也等于访问 T_j 的元素个数;同时,由于所有的 n 个元素都要访问 T_1,所以 $l_1(t)=nt$.基于以上事实,可得:

$$l_j(t)=\begin{cases}nt, & j=1\\ nt\cdot\displaystyle\prod_{i=1}^{j-1}\gamma_0^i(t), & j\in[2,k]\end{cases}\tag{7}$$

根据定义 2,为了达到计算复杂度 ξ,GBF 一共需要访问 $n\xi$ 次存储单元.不妨设 $t=t_\xi$ 时,GBF 的计算复杂度达到 ξ.假设 k 个哈希函数服从均匀分布,所以当 $t=t_\xi$ 时,访问 T_j 的次数 $l_j(t_\xi)$应与访问 GBF 的总数 $n\xi$ 成比例,即 $l_j(t_\xi)/n\xi=m_j/m=\alpha_j$.根据式(7),将 $l_j(t_\xi)$替换式(6)中的 n_jt,并将 $m_j=m\cdot l_j(t_\xi)/n\xi$ 代入(6)式,可得:

$$\gamma_j^{\mathrm{GBF}}(t_\xi)=1-\frac{mh}{n\xi}+\frac{m}{n\xi}\sum_{i=0}^{h-1}(h-i)\cdot\frac{1}{i!}(n\xi/m)^i\mathrm{e}^{-n\xi/m}=p(\xi)\tag{8}$$

基于以上解析表达式,下面给出关于 GBF 的几何分布定理.

定理 1　如果给定 GBF 的计算复杂度 ξ,则组成 GBF 的 k 个 Naïve-GBF 的大小 $\alpha_j(j\in[1,k])$满足等式 $\alpha_j=\left(\frac{1-p(\xi)}{1-p(\xi)^k}\right)\cdot p(\xi)^{j-1}$,即 GBF 结构服从几何分布.

证明:由式(7)和式(8)可知,$l_j(t_\xi)=nt_\xi\cdot p(\xi)^{j-1}$.因为 $\alpha_j=l_j(t_\xi)/n\xi$,则 $\alpha_j=(t_\xi\cdot p(\xi)^{j-1})/\xi$.又 $\sum_{j=1}^{k}\alpha_j=1$,可推出:$t_\xi=\xi\cdot\left(\frac{1-p(\xi)}{1-p(\xi)^k}\right)$.将 t_ξ 代入 $\alpha_j=\frac{t_\xi\cdot p(\xi)^{j-1}}{\xi}$,可得 $\alpha_j=\left(\frac{1-p(\xi)}{1-p(\xi)^k}\right)\cdot p(\xi)^{j-1}$,证毕.

当 $t=t_\xi$ 时,GBF 溢出元素个数=时刻 t_ξ 未插入的元素+插入过程中丢弃的元素,即 $n-nt_\xi+nt_\xi\cdot(p(\xi)^k)$.除以 n,容易得到 GBF 的溢出概率表达式:

$$\gamma^{\mathrm{GBF}}(t_\xi)=1-\xi(1-p(\xi))=\gamma^{\mathrm{GBF}}(\xi)\tag{9}$$

5　实验结果与分析

5.1　错误概率比较

根据定义 1,GBF 的错误概率为 $e=f^{\mathrm{GBF}}+\gamma^{\mathrm{GBF}}$.溢出概率由式(9)计算,误判概率由下面的定理给出.

定理 2　设 GBF"哈希指纹"的长度为 $l=l_b+l_c+l_e$,其中 l_b、l_c、l_e 分别代表桶索引长度、单元索引长度、元素标识符长度,则 GBF 误判概率表达式为:$f^{\mathrm{GBF}}=\frac{1}{(\alpha_1\cdot 2^{l_e})}$.

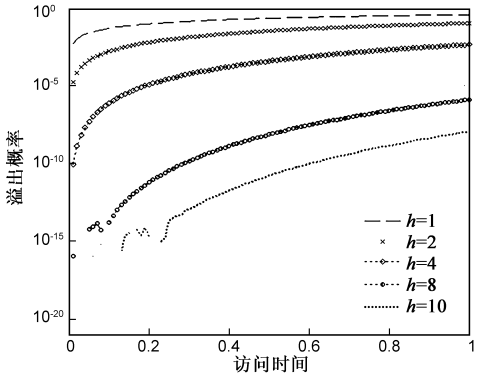

图4 给定n/m=1,溢出概率变化曲线图

证明:考虑 GBF 所能承载最多元素 $n = mh$ 的情况,当元素 $y \notin S$ 出现误判时,当且仅当存在集合元素 $x \in S$,使得 $H(y) = H(x)$时,GBF 才会产生误判.因此,可得误判概率为 $f^{\mathrm{GBF}} = \frac{mh}{2^l} = \frac{mh}{2^{l_b} \cdot 2^{l_c} \cdot 2^{l_e}}$.因为 GBF 中第一个 Naïve-GBF 桶的数量最多,所以令 $\alpha_1 m = 2^{l_b}$.又 $h = 2^{l_c}$,则 $f^{\mathrm{GBF}} = \frac{mh}{\alpha_1 m \cdot h \cdot 2^{l_e}} = \frac{1}{\alpha_1 \cdot 2^{l_e}}$,误判概率与 m, n, k 无关,证毕.

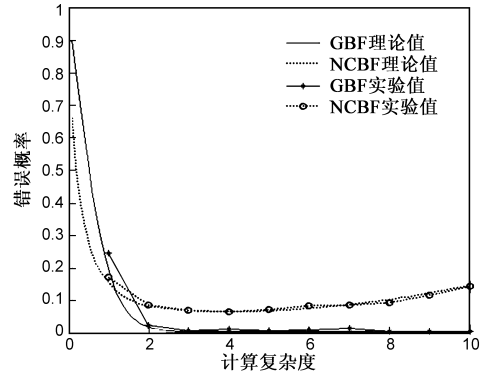

图5 错误概率与计算复杂度曲线图

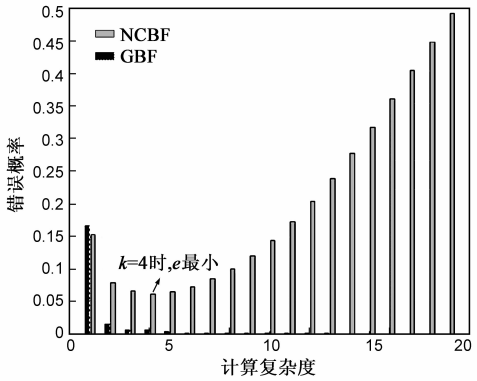

图6 错误概率与计算复杂度柱状关系图

实验中设定 $n = 4m$, $m = 4096$, $h = 4$, $l_e = 10$,则 $l_c = \log_2^h = 2$, $l_b = \log_2^{\alpha_1 m}$.为了保证空间大小一致,令 NCBF 的空间大小与 GBF 相等,等于 $mh(l_b + l_e + l_c)$.根据第2节的分析,若设定 NCBF 每计数器占用 4 比特,易得 NCBF 计数器个数 $c = mh(l_b + l_e + l_c)/4$.在空间大小一致的情况下,图 5 分别通过理论解析式和实验模拟的方式,仿真比较了 NCBF 和 GBF 的错误概率.从图中可以看出为:当 $\xi \leqslant 1$ 时,GBF 的错误概率大于 NCBF;随着 ξ 的增大,GBF 能获得非常低的错误概率,而 NCBF 的误判概率还会增加.这主要是因为当 ξ 较小时,在空间大小相等的情况下,GBF 桶的数量较少,哈希冲突比较严重,导致错误概率增大.

5.2 计算复杂度比较

在空间大小相同的前提下,图 6 对 GBF 和 NCBF 进行了计算复杂度的比较.实验中的相关参数与 5.1 节一致.NCBF 的计算复杂度为哈希函数的个数 k,等同于 GBF 的 ξ.图 6 表明:当 $k_{\mathrm{opt}} = \lceil \ln 2 \cdot (c/n) \rceil = 4$ 时,NCBF 误判概率能达到最小;当 $\xi = 1$ 时,GBF 错误概率稍微比 NCBF 大,但随着 ξ 的增大,GBF 的错误概率要比 NCBF 小很多.这主要是由于 GBF 结构具有很好的哈希冲突处理机制:只有被插入元素对应的桶满载时,才会继续搜索下一个 Naïve-GBF,使得尽量多的集合元素存放在前面的 Naïve-GBF 中.

5.3 空间复杂度比较

根据定义 3,空间复杂度用每元素所占用比特数来衡量,即 $\eta_{\mathrm{GBF}} = \lceil mhl/n \rceil$, $\eta_{\mathrm{NCBF}} = \lceil 4c/n \rceil$.表 3 给出了在错误概率近似相等的情况下,NCBF 和 GBF 空间复杂度的对比.由式(9)可知,GBF 的错误概率只与 $< m, n, h, \xi >$ 有关,因此,每组测试都是在 $< m, n, h, \xi >$ 确定的情况下进行的.同时,为了保证计算复杂度的公平性,每次测试令 NCBF 哈希函数的个数等于 ξ.实验中设定元素基数 $N = 4096$,并分别在 h 和 m/n 变化的情况下进行了比较.通过表 3,可以看出:在错误概率一定的情况下,GBF 空间利用率的空间利用率比 NCBF 大约提高 20%.

表 3 空间复杂度比较

错误概率	变量	NCBF 空间利用率	GBF 空间利用率
$< N, 4N, h, 4 >$	$h = 2$	14	11
	$h = 4$	26	23
	$h = 8$	62	50
$< m, n, 4, 4 >$	$m = N, n = 2N$	63	47
	$m = N, n = 4N$	26	23
	$m = N, n = 8N$	16	11

6 结论

本文立足于错误概率、空间复杂度和计算复杂度,提出了几何布鲁姆过滤器 GBF.该结构首先引入了“哈希指纹”的概念,使得误判概率不受不受集合元素个数 n、向量位数 m 以及哈希函数个数 k 的影响;其次,把 NCBF 分割成若干呈几何分布的 Naïve-GBF,每个 Naïve-

GBF 再次分割成等深度的桶,并将哈希冲突的元素存放在桶中,使得冲突概率大大降低;最后,利用"微分方程"的相关理论,给出了错误概率和计算复杂度的解析表达式,给出了 GBF 的结构性定理.实验仿真表明:GBF 的准确性、空间利用率以及计算复杂度等方面都有较大改善.针对 GBF 还可以进一步关注和探讨:如何根据数据元素的分布特性自适应调整桶的深度和存储单元的长度.

参考文献

[1] B Bloom. Space/ time tradeoffs in hash coding with allowable errors [J]. Communications of the ACM, 1970, 13(7): 422 – 426.

[2] Mullin JK. Optimal semi-joins for distributed database systems [J]. IEEE Transaction on Software Engineering, 1990, 16(5): 558 – 560.

[3] Byers J W, et al. Informed content delivery across adaptive overlay networks [J]. IEEE/ACM Transactions on Networking, 2002, 12(5): 767 – 780.

[4] 王洪波,程时端,林宇.高速网络超链接主机检测中的流抽样算法研究[J].电子学报,2008,36(4):809 – 818.
WANG Hongbo, CHENG Shiduan, LIN Yu. On flow sampling for identifying super-connection hosts [J]. Acta Electronica Sinica, 2008, 36(4): 809 – 818. (in Chinese).

[5] Rhea SC, Kubiatowicz J. Probabilistic location and routing [A]. IEEE INFOCOM [C]. California: Berkeley, 2002. 1248 – 1257.

[6] Whitaker A, Wetherall D. Forwarding without loops in Icarus [A]. IEEE Open Architectures and Network Programming Proceedings [C]. Washington: University of Washington, 2002. 63 – 75.

[7] L Fan, P Cao, J Almeida, A Z Broder. Summary cache: a scalable wide-area web cache sharing protocol [J]. IEEE/ACM Transactions on Networking, 2000, 8(3): 281 – 293.

[8] M Mitzenmacher. Compressed Bloom filters [J]. IEEE/ACM Transactions on Networking, 2002, 10(5): 604 – 612.

[9] Saar C, Yossi M. Spectral Bloom filters [A]. ACM SIGMOD [C]. San Diego: ACM Press, 2003. 241 – 252.

[10] Xie K, Ming YH, Zhang DF, Xie GG, Wen JG. Basket Bloom filters for membership queries [J]. Chinese Journal of Computers, 2007, 30(4): 597 – 607.

[11] Flavio B, Michael M, Rina P, Sushil S, George V. An improved construction for counting Bloom filters [A]. Annual European Symposium [C]. Zurich: Springer-Verlag, 2006. 684 – 695.

[12] A Broder, M Mitzenmacher. Network applications of Bloom filters: a survey [J]. Internet Mathematics, 2004, 1(4): 485 – 509.

[13] T G Kurtz. Solutions of ordinary differential equations as limits of pure jump Markov processes [J]. Journal of Applied Probability, 1970, 7(1): 49 – 58.

[14] M Mitzenmacher. The Power of Two Choices in Randomized Load Balancing [D]. Berkeley: University of California, 1996.

[15] M Mitzenmacher, E Upfal. Probability and Computing: Randomized Algorithm and Probabilistic Analysis [M]. Cambridge, U K: Cambridge Univ Press, 2005. 823 – 829.

作者简介

张　震　男,1985 年生于山东省济宁市.现为国家数字交换系统工程技术研究中心博士研究生.主要研究方向为网络测量和网络管理.
E-mail: zhangzhenhigh@gmail.com

汪斌强　男,1963 生于安徽省桐城.现为国家数字交换系统工程技术研究中心教授,博士生导师.主要研究方向为宽带信息网络和路由交换技术.
E-mail: wbq@mail.ndsc.com.cn

陈庶樵　男,1973 年生于黑龙江省肇县.现为国家数字交换系统工程技术研究中心教授,硕士生导师.主要方向为网络体系结构和路由交换技术.
E-mail: Chenshuqiao1973@163.com

郭　通　男,1984 年生于江西省南昌市.现为国家数字交换系统工程技术研究中心博士研究生.主要研究方向为网络测量.
E-mail: guotong1984@yahoo.com.cn