计算机科学
COMPUTER SCIENCE

过滤器数据结构研究综述

王瀚橙，戴海鹏，陈树森，陈志鹏，陈贵海

引用本文

王瀚橙，戴海鹏，陈树森，陈志鹏，陈贵海. 过滤器数据结构研究综述[J]. 计算机科学，2024，51(1)：35-40.

WANG Hancheng, DAI Haipeng, CHEN Shusen, CHEN Zhipeng, CHEN Guihai. Filter Data Structures:A Survey [J]. Computer Science, 2024, 51(1): 35-40.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.231000193

过滤器数据结构研究综述

王瀚橙　戴海鹏　陈树森　陈志鹏　陈贵海
计算机软件新技术国家重点实验室(南京大学)　南京 210023
(hanchengwang@smail.nju.edu.cn)

摘　要　过滤器数据结构可以近似地判断某个元素是否属于给定集合。典型的过滤器数据结构,如布隆过滤器、布谷鸟过滤器、商过滤器,以牺牲查询准确性为代价换取更低的内存空间消耗和查询时间开销。因此,得益于空间时间高效性,过滤器数据结构现已被广泛应用于计算机网络、物联网、数据库系统、文件系统、生物信息学、机器学习等领域的近似成员资格查询操作中。自20世纪70年代以来,过滤器数据结构受到了广泛的研究,在诸多领域取得了重要的进展,其研究思路也在不断变化。文中整理了近五十年来关于过滤器数据结构的经典研究成果,从过滤器数据结构的原理出发对已有工作进行分类总结,并比较不同工作之间的引证关系和改进思路,最后讨论了过滤器数据结构的未来研究方向。

关键词:过滤器;近似成员资格查询;概率数据结构;布隆过滤器;布谷鸟过滤器;商过滤器
中图分类号　TP391

Filter Data Structures:A Survey

WANG Hancheng,DAI Haipeng,CHEN Shusen,CHEN Zhipeng and CHEN Guihai
State Key Laboratory for Novel Software Technology,Nanjing University,Nanjing 210023,China

Abstract　Filter data structures can approximately determine whether an element exists in a given set. Typical filter data structures,such as Bloom filters,cuckoo filters,and quotient filters,sacrifice query accuracy for lower memory space consumption and lower query time overhead. Due to their spatial and temporal efficiency,filter data structures are now widely used in approximate membership query operations in computer networks,the Internet of Things,database systems,file systems,bioinformatics,machine learning,and other fields. Since the 1970s,filters have been extensively studied. Their research ideas are constantly changing. This paper compiles the classic studies on filter data structures in the past fifty years,summarizes existing studies based on the mechanism of filter data structures and analyze the relationship between different studies. Finally,future research directions in filter data structures are discussed.

Keywords　Filter,Approximate membership query,Probabilistic data structure,Bloom filter,Cuckoo filter,Quotient filter

1　引言

判断某一元素是否属于给定集合是计算机科学中的常见问题,在计算机网络、物联网、数据库系统、生物信息学、机器学习等诸多领域具有广泛应用[1-26]。以布隆过滤器和布谷鸟过滤器为代表的过滤器数据结构可以近似地判断某个元素是否属于给定集合,现已被广泛应用于网络路由[1]、信息检索[2]、文件合并[3]、垃圾邮件检测[4]、分布式系统[5]等诸多应用。

例如,数据库系统执行数据检索操作会触发大量磁盘读取并产生巨大开销。现代数据库系统将磁盘中存储的数据分成若干数据块,并将每个数据块中存储的数据保存到该数据块对应的过滤器数据结构中。当数据库执行数据检索操作时,通过查询数据块对应的过滤器数据结构可近似地判断待检索数据是否存在于该数据块中。若过滤器数据结构返回待检索数据不存在于该数据块,则可避免读取该数据块所需的磁盘读取开销,继而提升数据库检索数据的性能。

随着现实世界中需要处理的数据集合规模不断增大,在海量数据中快速检索数据愈发成为现有系统的瓶颈。优化过滤器数据结构可以显著提升现有系统性能,因此受到了国内外专家学者的广泛研究,并取得了诸多研究成果。现阶段针对过滤器数据结构的研究工作众多,但缺乏对过滤器数据结构系统性的分析和总结。这影响了过滤器数据结构的进一步提升,也限制了过滤器数据结构的广泛应用。因此,本文分类总结了近五十年来关于过滤器数据结构的经典研究成果,并对本领域的未来发展趋势进行了展望。

本文第1章介绍了研究背景;第2章概述了过滤器数据结构,并从原理出发将现有过滤器数据结构分成两类;第3章

到稿日期:2023-10-26　返修日期:2023-11-23
基金项目:国家自然科学基金(62272223)
This work was supported by the National Natural Science Foundation of China(62272223).
通信作者:戴海鹏(haipengdai@nju.edu.cn)

和第 4 章分别回顾上述两类过滤器数据结构的研究进展，分析不同工作之间的引证关系和改进之处；第 5 章对过滤器数据结构的未来发展趋势进行了展望；最后总结全文并展望未来。

2 过滤器数据结构概述

过滤器数据结构可以近似地判断某个元素是否属于给定集合。形式化定义如下：对于给定集合 S 以及待查询元素 x，过滤器数据结构可以近似地回答“x 是否属于 S”。其中，近似地指：当元素 x 不属于集合 S 时，过滤器数据结构查询的结果以较小的概率 p 返回“x 属于 S”；当元素 x 属于集合 S 时，过滤器数据结构查询的结果一定是“x 属于 S”。

过滤器数据结构通过牺牲查询准确性来换取更高的时间空间效率。具体来说，支持准确查询的数据结构通常需要保存元素的全部信息。与此类数据结构不同，过滤器数据结构通过保存元素的部分信息（例如指纹）近似地存储元素。基于过滤器数据结构保存元素部分信息的原理，现有过滤器数据结构可分为两类：1）以布隆过滤器为代表的基于位图的过滤器数据结构；2）以布谷鸟过滤器为代表的基于指纹的过滤器数据结构。基于上述认识，本文将从不同过滤器数据结构保存元素部分信息的原理出发，对现有工作进行分类总结，并分析不同工作之间的引证关系和改进之处。

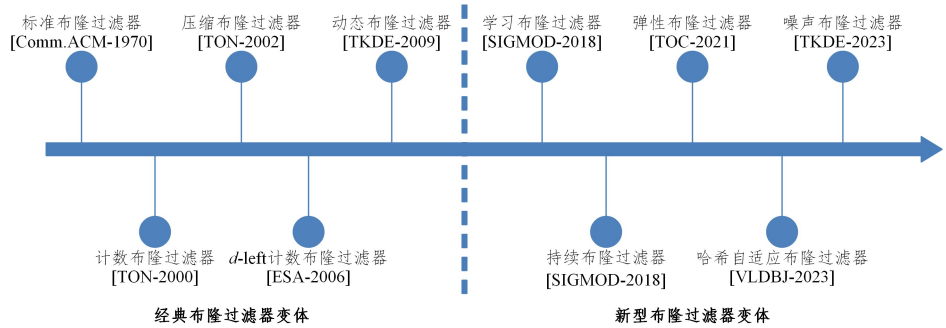

图 1 基于位图的过滤器数据结构

Fig. 1 Bitmap-based filter data structures

3 基于位图的过滤器数据结构

基于位图的过滤器数据结构主要包括标准布隆过滤器及其变体。图 1 给出了此类过滤器数据结构的发展历程及典型代表。图 1 中，基于位图的过滤器数据结构的发展历程主要可分为经典布隆过滤器变体和新型布隆过滤器变体两个阶段，前一阶段主要填补了标准布隆过滤器功能的空白，并在标准布隆过滤器基础上不断提高操作性能；后一阶段主要结合机器学习、数据挖掘等新技术、新场景进行进一步拓展。后文将以图 1 为大纲，按照时间顺序梳理基于位图的过滤器数据结构。

3.1 布隆过滤器

基于位图的过滤器数据结构研究最早开始于 20 世纪 70 年代的标准布隆过滤器[27]。标准布隆过滤器由一个长度为 m 比特的布尔数组 B 和 k 个独立的哈希函数组成，每个哈希函数随机地将元素 x 映射到布尔数组的某一位置。基于上述结构，标准布隆过滤器的插入操作将 k 个哈希函数映射到的 k 个位置设置为 1。当查询某个元素时，标准布隆过滤器通过检查 k 个位置是否全部为 1 来判断元素是否存在。图 2 给出了一个使用 3 个哈希函数将元素 x 和 y 插入到标准布隆过滤器的示例。

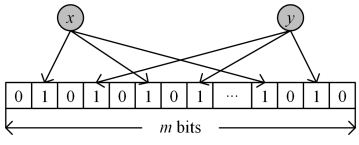

图 2 标准布隆过滤器数据结构示例

Fig. 2 Example of a standard Bloom filter

由于标准布隆过滤器使用布尔数组存储元素，当有多个元素同时映射到相同位置时，布隆过滤器仅会被设置一次。加之标准布隆过滤器不存储每个位置由哪些元素设置，因此当多个元素由于哈希冲突同时映射到相同位置时便会产生假阳性。更重要的是，由于可能有多个元素同时共享一个比特位，删除某一个元素时会影响到同样映射到该位置的其他元素，因此标准布隆过滤器无法支持删除操作。

作为基于位图的过滤器数据结构的首创工作，标准布隆过滤器实现了近似成员资格查询的最基本功能。但是由于现实应用场景的复杂性，标准布隆过滤器缺乏删除、压缩、扩容之类的功能。因此，产生了一系列有关填补布隆过滤器功能空白的工作。

3.2 经典布隆过滤器变体

3.2.1 增加删除功能的布隆过滤器变体

由于标准布隆过滤器不支持删除操作，系列有关标准布隆过滤器可删除变体的研究陆续被提出。其中，最具代表性的工作是 2000 年提出的计数布隆过滤器[28]。

计数布隆过滤器使用计数器数组代替布尔数组，不同于标准布隆过滤器直接将布尔数组比特位设置为 1，计数布隆过滤器执行插入操作时将计数器数组对应位置的值加 1。计数布隆过滤器的查找操作和标准布隆过滤器一致，需要检查每个计数器是否为零。计数布隆过滤器的删除操作是插入操作的逆操作，即当执行删除操作时，直接将对应计数器的值减 1。

为了防止计数器溢出，计数布隆过滤器数组中的每个计数器都需要具有足够大的空间以防止插入溢出。因此，在实践中，计数布隆过滤器的空间消耗通常是标准布隆过滤器空间消耗的 4 倍。这引发了一系列针对计数布隆过滤器的

改进。具体来说,由于计数布隆过滤器存在巨大空间开销,因此产生了许多可在保持低空间开销前提下支持删除功能的标准布隆过滤器变体。其中最具代表性的工作是 d-left 计数布隆过滤器[29]。d-left 计数布隆过滤器将 d-left 哈希应用于计数布隆过滤器中,在不影响删除功能的前提下,空间占用平均节约 50%以上。

3.2.2 增加压缩功能的布隆过滤器变体

标准布隆过滤器以及计数布隆过滤器在计算机网络、物联网等空间要求严格的应用中存在不足。具体来说,在计算机网络应用中,网络设备上的内存空间资源以及网络设备之间的带宽资源有限,现实应用中可供标准布隆过滤器使用的空间极少。因此过滤器数据结构的空间优化受到了广泛的关注,其中最具代表性的工作是压缩布隆过滤器[30]。具体来说,标准布隆过滤器作为一种基于哈希的数据结构难以被压缩。当标准布隆过滤器达到最优误报率时,布尔数组每一位被置为 1 的概率和被置为 0 的概率相同。根据香农信息熵理论,此时无法进一步压缩数据结构空间。因此,简单地对标准布隆过滤器应用常见的压缩算法,难以获得明显的空间优化。这需要通过对标准布隆过滤器理论进行新的分析,才有望进一步降低内存消耗。具体来说,标准布隆过滤器的性能主要受到 3 个参数的影响:布尔数组长度 m、假阳率 p 以及哈希函数个数 k。压缩布隆过滤器将压缩率作为第四个优化目标,通过理论分析后发现,在最优标准布隆过滤器的基础上进一步增大布尔数组的长度,不但可以提升压缩率,还能进一步降低其假阳率[30]。基于这一设计,压缩布隆过滤器获得了比标准布隆过滤器更优的空间性能,即相同空间下误报率更低,相同误报率下使用空间更少,极大地降低了过滤器数据结构的传输成本。

压缩布隆过滤器现已被广泛应用于带宽有限的应用场景。作为标准布隆过滤器的一个重要变体,压缩布隆过滤器启发了诸多有关过滤器数据结构空间优化的应用与研究。

3.2.3 增加扩容功能的布隆过滤器变体

标准布隆过滤器不支持扩容操作。具体来说,标准布隆过滤器需要在初始化时确定布尔数组长度,之后布尔数组长度在整个标准布隆过滤器使用期间无法修改。这要求标准布隆过滤器在构造时预判未来可能插入的元素总数。但是,实际应用中过滤器数据结构中存储的元素数量是动态变化的。基于上述背景,支持扩容的标准布隆过滤器被纷纷提出,其中最具有代表性的是动态布隆过滤器[31]。为了支持动态数据集,动态布隆过滤器根据数据量自适应调整大小。动态布隆过滤器通过不断链接新的计数布隆过滤器来实现增量扩容。具体来说,当动态布隆过滤器中的元素数量达到其容量后,动态布隆过滤器在原有结构的基础上链接一个新的计数布隆过滤器,以增大其容量。这种思路简单易行,但存在两点问题:1)添加更多的过滤器后,每次执行查询操作时需要检查更多的过滤器,导致查询时间随着扩容不断增长;2)虽然动态布隆过滤器是基于可删除的计数布隆过滤器,但是当其执行删除操作时无法判断应该删除哪个过滤器中的元素,因此其不支持可信删除。针对上述问题,弹性布隆过滤器[32]实现了支持删除操作的可扩容布隆过滤器。

上述 3 个小节主要介绍了 2010 年前基于位图的过滤器数据结构的主要研究工作。本文将这一阶段称为经典布隆过滤器变体阶段。这一阶段的研究主要针对过滤器数据结构常用功能进行优化,填补了原始标准布隆过滤器存在的诸多空白。2010 年后依然有许多工作针对这些基本功能进行优化,从理论和实践两个维度不断追求更优的性能。与此同时,2010 年后针对基于位图的过滤器数据结构的研究思路逐渐改变,从先前功能导向的优化逐渐转变为结合新场景、新技术、新任务进行优化,产生了诸多性能极优的新型过滤器数据结构。下文将按照时间顺序对其中的代表性研究进行介绍。

3.3 新型布隆过滤器变体

3.3.1 基于机器学习的布隆过滤器变体

使用机器学习模型创建学习布隆过滤器有望提高标准布隆过滤器的空间效率[33]。学习布隆过滤器假设插入的元素遵循某种特定的模式,使用机器学习模型学习该模式后,可通过机器学习模型判断元素是否已插入到集合中。具体来说,学习布隆过滤器往往由分类模型和辅助数据结构组成。分类模型首先对大多数元素进行粗略分类,之后辅助结构作为备份用来弥补模型不能正确分类的元素。由于学习布隆过滤器中的分类模型可对插入数据进行高度概括,并且学习布隆过滤器的模型部分仅存储数据模式,因此可以极大地降低空间消耗。基于这一设计,在实践中,学习布隆过滤器实现了在空间消耗维度远低于标准布隆过滤器的效果。

但是现阶段学习布隆过滤器依然有诸多不足,其中最为突出的是分类模型往往需要长时间训练,这造成了学习布隆过滤器难以支持插入、删除操作。因此,学习布隆过滤器不擅长动态变化的数据集合的处理。文献[22]为学习布隆过滤器增加了插入和删除功能,有望解决这一问题。

3.3.2 面向数据挖掘的布隆过滤器变体

近年来,随着物联网、移动互联网在工业生产、城市交通、数字经济、网络直播等领域的快速发展,主流数据类型逐渐从传统的文件、表格等静态形式向非结构化、多模态等动态形式转变。随着计算设备的微型化和普及化,海量设备无时无刻不在产生着动态流式数据,这为信息的采集、存储、计算、分析带来了巨大的机遇和挑战。这些数据源源不断地生成,其中包含着许多有价值的信息等待挖掘,分析这些动态流式数据可以帮助人们更好地利用大数据信息。

基于上述背景,使用布隆过滤器进行数据挖掘的研究逐渐增多,其中最具有代表性的工作是持续布隆过滤器[34]。具体来说,标准布隆过滤器插入元素时并未存储元素的时间信息。在动态数据流场景下,所有存储在过滤器中的元素是没有时间先后顺序的。因此,标准布隆过滤器无法判断某个元素在过去的某个时间段内是否存在于集合中。持续布隆过滤器通过将布隆过滤器组织成二叉树结构,将每个时间片不断细分,从而存储每个时间段的数据信息。之后所有的插入、查询操作都会转换为针对二叉树上的多个布隆过滤器的插入、查找操作。当需要查询一个时间段的元素时,只需要查找对应的一系列布隆过滤器中是否存在该元素即可。

3.3.3 基于代价模型的布隆过滤器变体

在真实世界应用中,不同元素发生误报时产生的代价

存在差异，因此不同元素存在不同权重。标准布隆过滤器忽略了不同元素误报时产生的代价不同，因此将标准布隆过滤器与代价模型结合可获得加权最优的性能，有助于优化系统整体性能。与代价模型结合进行优化的典型代表是哈希自适应布隆过滤器[20,35]。该工作为误报代价较高的元素分配更多哈希函数。这一设计可以降低误报代价较高的元素发生误报的概率，从而实现加权最优的误报率。

上文主要总结了2010年后基于位图的过滤器数据结构的主要优化方向及其代表工作。本文将这一阶段称为新型布隆过滤器变体阶段。该阶段结合新技术、新场景提出了诸多性能极优的新型布隆过滤器变体。

综上所述，以布隆过滤器及其变体为代表的基于位图的过滤器数据结构[27-44]可分为经典布隆过滤器变体和新型布隆过滤器变体两个阶段。此类过滤器数据结构存储元素信息的原理均基于位图技术。下文将介绍另一类基于指纹原理存储元素信息的过滤器数据结构。本文将该类过滤器数据结构命名为基于指纹的过滤器数据结构。

4 基于指纹的过滤器数据结构

基于指纹的过滤器数据结构主要包括布谷鸟过滤器[45]及其变体，以及商过滤器[46]及其变体。布谷鸟过滤器的变体主要包括动态布谷鸟过滤器[47]、莫顿过滤器[48-49]、真空过滤器[50]、异或过滤器[51]、对数布谷鸟过滤器[52]、竹过滤器[18]、跳跃过滤器[53]、加减布谷鸟过滤器[54]、标签布谷鸟过滤器[55]等等。商过滤器的变体主要包括计数商过滤器[56]、向量商过滤器[57]、无限过滤器[58]等。接下来本文从数据结构的两大起源出发，详细介绍此类过滤器数据结构的发展历程。

4.1 布谷鸟过滤器及其变体

布谷鸟过滤器使用指纹存储插入的元素。具体来说，布谷鸟过滤器的核心结构是布谷鸟哈希表。当执行插入操作时，布谷鸟过滤器将元素的 f 比特哈希值(即指纹)存储到布谷鸟哈希表中。当执行查询操作时，布谷鸟过滤器通过查询元素的指纹是否存在于哈希表中来判断元素是否存在。不同于基于位图的布隆过滤器，布谷鸟过滤器执行删除操作时直接删除元素对应的 f 比特指纹即可，因此布谷鸟过滤器支持删除功能。近期大量工作从功能和性能角度出发，促进了布谷鸟过滤器的实用性。

在功能方面，由于布谷鸟过滤器不支持扩容操作，动态布谷鸟过滤器[47]、莫顿过滤器[49]、对数布谷鸟过滤器[52]、竹过滤器[18]、跳跃过滤器[53]为布谷鸟过滤器添加了扩容功能，并不断减小扩容对布谷鸟过滤器的影响。具体来说，动态布谷鸟过滤器将若干个相同的布谷鸟过滤器组织成链表。之后，动态布谷鸟过滤器通过增加链表中布谷鸟过滤器的数量来增加动态布谷鸟过滤器的容量。但是，需要注意，动态布谷鸟过滤器的查询操作需要依次检查链表中的每个布谷鸟过滤器。因此，多次扩容后查询性能会不断下降。为解决这一问题，竹过滤器[18]提出了部分键线性哈希技术，用于实现细粒度扩容。基于这一技术，即使多次扩容后，竹过滤器依然可以维持较高的查询吞吐。

在性能方面，莫顿过滤器[48-49]、异或过滤器[51]提升了布谷鸟过滤器执行插入、查找和删除操作的吞吐，并进一步降低了误报率。此外，加减布谷鸟过滤器[54]和标签布谷鸟过滤器[55]可以摆脱布谷鸟过滤器的哈希表长度必须是2的整数次幂的限制，极大地增强了布谷鸟过滤器的实用性。具体来说，标准布谷鸟过滤器使用异或运算计算两个候选桶的索引，这造成布谷鸟过滤器的哈希表长度必须是2的整数次幂。加减布谷鸟过滤器将异或运算修改为加减运算，这一设计摆脱了哈希表长度必须是2的整数次幂的限制，提升了布谷鸟过滤器的空间效率。标签布谷鸟过滤器将实际长度小于2的整数次幂的哈希表在逻辑层面补充为2的整数次幂大小。之后，标签布谷鸟过滤器为每个指纹添加一个比特位，用于标识该元素是否应保存在逻辑层面补充的部分。这一设计可使哈希表长度支持任意大小，摆脱了哈希表长度必须是2的整数次幂的限制，实现了更优的空间效率。

4.2 商过滤器及其变体

商过滤器也通过保存元素的指纹存储元素。具体来说，商过滤器将元素的哈希值划分成商和指纹两部分，商用于定位指纹存储的位置，指纹用于存储元素。由于商过滤器使用指纹存储元素，因此商过滤器支持删除操作。此外，由于商过滤器可以将商和指纹拼接来获得原始哈希值，因此商过滤器支持扩容操作。计数商过滤器[56]在商过滤器的基础上降低了空间开销。此外，该过滤器还通过变长计数器编码方案实现了计数功能，并通过细粒度锁实现了高效并发。向量商过滤器[57]主要面向高负载因子场景，优化哈希冲突导致的插入操作性能恶化的问题，并且通过细粒度自旋锁实现了细粒度并发。

5 总结与展望

根据过滤器数据结构存储元素的原理，现有过滤器数据结构可分为基于位图的过滤器数据结构和基于指纹的过滤器数据结构两类。其中基于位图的过滤器数据结构主要包括布隆过滤器及其变体，基于指纹的过滤器数据结构主要包括布谷鸟过滤器、商过滤器及其变体。

回顾过滤器数据结构的发展历程，不难发现不同种类的过滤器数据结构均经历了相似的发展过程：完善功能、提升性能、面向应用。基于上述认识以及现阶段的研究热点，本文针对过滤器数据结构的未来发展进行了如下展望。

1)完善功能。在现实应用需求的驱动下，研究人员从删除、压缩、扩容等多个维度出发，不断丰富现有过滤器数据结构的功能。随着现实需求复杂化、多样化，结合现实应用中涌现的新需求来完善过滤器数据结构的功能意义重大。

2)提升性能。过滤器数据结构性能的主要衡量指标包括空间消耗、误报率和操作吞吐3个维度。随着海量数据的不断产生，现有过滤器数据结构的性能有待进一步提升。结合硬件指令进一步提升现有过滤器数据结构的性能意义重大。

3)面向应用。随着海量数据的飞速产生，现实世界中将有更多应用场景需要使用过滤器数据结构进行加速。因此，这要求研究人员探索如何在更多应用场景中使用过滤器数据结构，以更好地加速现有应用。

结束语 判断元素是否存在于给定集合是计算机领域的

基本问题之一，过滤器数据结构可以高效地执行近似成员资格查询，因此被广泛应用于现代系统中。本文将现有工作分为基于位图的过滤器数据结构和基于指纹的过滤器数据结构两类，并分别回顾了两类过滤器数据结构的发展历程。最后，本文结合两类过滤器数据结构的发展历史，从功能、性能及应用 3 个角度对过滤器数据结构的未来发展趋势进行了展望。

参 考 文 献

[1] YU M, FABRIKANT A, REXFORD J. BUFFALO: Bloom filter forwarding architecture for large organizations[C]// Proceedings of International Conference on Emerging Networking Experiments and Technologies. 2009: 313-324.

[2] ATHANASSOULIS M, AILAMAKI A. BF-Tree: Approximate tree indexing[J]. Proceedings of the VLDB Endowment, 2014, 7(14): 1881-1892.

[3] LI P, LUO B, ZHU W, et al. Cluster-based distributed dynamic cuckoo filter system forRedis[J]. International Journal of Parallel, Emergent and Distributed Systems, 2020, 35(3): 340-353.

[4] WANG F, CHEN H, LIAO L, et al. The power of betterchoice: Reducing relocations in cuckoo filter[C]// Proceedings of International Conference on Distributed Computing Systems. 2019: 358-367.

[5] GU R, LI S, DAI H, et al. Adaptive online cache capacity optimization via lightweight working set size estimation atscale[C]// Proceedings of USENIX Annual Technical Conference. 2023: 467-484.

[6] DAI H, LI M, LIU A X, et al. Finding persistent items in distributed datasets[J]. IEEE/ACM Transactions on Networking, 2020, 28(1): 1-14.

[7] DAI H, SHAHZAD M, LIU A X, et al. Identifying and estimating persistent items in data streams[J]. IEEE/ACM Transactions on Networking, 2018, 26(6): 2429-2442.

[8] XIA R, DAI H, DU Z, et al. Mining robust frequent items in data streams[C]// Proceedings of International Conference on Joint Cloud Computing. 2020: 110-117.

[9] DAI H, LI M, WANG H, et al. Distantly supervised entity linking with selection consistency constraint[C]// Proceedings of International Conference on Database Systems for Advanced Applications. 2023: 784-799.

[10] LIU J, DAI H, XIA R, et al. DUET: A generic framework for finding special quadratic elements in data streams[C]// Proceedings of ACM Web Conference. 2022: 2989-2997.

[11] YU S, LI X, WANG H, et al. C_CART: An instance confidence-based decision tree algorithm for classification[J]. Intelligent Data Analysis, 2021, 25(4): 929-948.

[12] LI M, DAI H, WANG X, et al. Thresholded monitoring in distributed data streams[C]// Proceedings of International Conference on Distributed Computing Systems. 2019: 218-227.

[13] YU S, LI X, WANG H, et al. BIDI: A classification algorithm with instance difficulty invariance[J]. Expert Systems with Applications, 2021, 165(1): 1-13.

[14] LI M, DAI H, WANG X, et al. Thresholded monitoring in distributed data streams[J]. IEEE/ACM Transactions on Networking, 2020, 28(3): 1033-1046.

[15] DAI H, SHAHZAD M, LIU A X, et al. Finding persistent items in data streams[J]. Proceedings of the VLDB Endowment, 2016, 10(4): 289-300.

[16] DAI M, XU S, SHAO S, et al. Blockchain-based reliable fog-cloud service solution for IIoT[J]. Chinese Journal of Electronics, 2021, 30(2): 359-366.

[17] DAI H, LI M, LIU A X, et al. Finding persistent items in distributed datasets[C]// Proceedings of International Conference on Computer Communications. 2018: 1403-1411.

[18] WANG H, DAI H, LI M, et al. Bamboo filters: Make resizing smooth[C]// Proceedings of International Conference on Data Engineering. 2022: 979-991.

[19] LI M, CHEN D, DAI H, et al. Seesaw counting filter: An efficient guardian for vulnerable negative keys during dynamic filtering[C]// Proceedings of the ACM Web Conference. 2022: 2759-2767.

[20] LI M, XIE R, CHEN D, et al. A pareto optimal bloom filter family with hash adaptivity[J]. The VLDB Journal, 2023, 32(3): 525-548.

[21] WANG H, DAI H, GU R, et al. Wormhole filters: Caching your hash on persistent memory[C]// Proceedings of European Conference on Computer Systems. 2024: 1-16.

[22] LI Y, WANG Z, YANG R, et al. Learned bloom filter for multi-key membership testing[C]// Proceedings of the International Conference on Database Systems for Advanced Applications. 2023: 62-79.

[23] DAI H, YU J, LI M, et al. Bloom filter with noisy coding framework for multi-set membership testing[J]. IEEE Transactions on Knowledge and Data Engineering, 2022, 35(7): 6710-6724.

[24] LI L, WU J, HU H, et al. Secure cloud architecture for 5G core network[J]. Chinese Journal of Electronics, 2021, 30(3): 516-522.

[25] DAI H, ZHONG Y, LIU A X, et al. Noisy bloom filters for multi-set membership testing[C]// Proceedings of International Conference on Measurement and Modeling of Computer Science. 2016: 139-151.

[26] LI P, YU X, XU H, et al. Secure localization technology based on dynamic trust management in wireless sensor networks[J]. Chinese Journal of Electronics, 2021, 30(4): 759-768.

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

[28] FAN L, CAO P, ALMEIDA J M, et al. Summary cache: A scalable wide area web cache sharing protocol[J]. IEEE/ACM Transactions on Networking, 2000, 8(3): 281-293.

[29] BONOMI F, MITZENMACHER M, PANIGRAHY R, et al. An improved construction for counting bloom filters[C]// Proceedings of the Annual European Symposium. 2006: 684-695.

[30] MITZENMACHER M. Compressed bloom filters [J]. IEEE/ACM Transactions on Networking, 2002, 10(5): 604-612.

[31] GUO D, WU J, CHEN H, et al. The dynamic bloom filters[J]. IEEE Transactions on Knowledge and Data Engineering, 2009, 22(1): 120-133.

[32] WU Y, HE J, YAN S, et al. Elastic bloom filter: Deletable and

expandable filter using elastic fingerprints[J]. IEEE Transactions on Computers, 2021, 71(4): 984-991.
[33] KRASKA T, BEUTEL A, CHI E H, et al. The case for learned index structures[C]// Proceedings of the 2018 International Conference on Management of Data. 2018: 489-504.
[34] PENG Y, GUO J, LI F, et al. Persistent bloom filter: Membership testing for the entire history[C]// Proceedings of the 2018 International Conference on Management of Data. 2018: 1037-1052.
[35] XIE R, LI M, MIAO Z, et al. Hash adaptive bloom filter[C]// Proceedings of the 37th IEEE International Conference on Data Engineering. 2021: 636-647.
[36] ROTHENBERG C E, MACAPUNA C A B, VERDI F L, et al. The deletable bloom filter: a new member of the bloom family [J]. IEEE Communications Letters, 2010, 14(6): 557-559.
[37] MOSHARRAF N, JAYASUMANA A P, RAY I. Compacted bloom filter[C]// Proceedings of the 2nd IEEE International Conference on Collaboration and Internet Computing. 2016: 304-311.
[38] ALMEIDA P S, BAQUERO C, PREGUIÇA N M, et al. Scalable bloom filters[J]. Information Processing Letters, 2007, 101(6): 255-261.
[39] LU J, YANG T, WANG Y, et al. One hashing bloom filter[C]// Proceedings of the International Symposium on Quality of Service. 2015: 289-298.
[40] BHATTACHARYA A, BEDATHUR S, BAGCHI A. Adaptive learned bloom filters under incremental workloads[C]// Proceedings of the India Joint International Conference on Data Science and Management of Data. 2020: 107-115.
[41] LIU Q, ZHENG L, SHEN Y, et al. Stable learnedbloom filters for data streams[J]. Proceedings of the VLDB Endowment, 2020, 13(11): 2355-2367.
[42] GOSWAMI M, PAGH R, SILVESTRI F, et al. Distance sensitive bloom filters without false negatives[C]// Proceedings of the Annual Symposium on Discrete Algorithms. Society for Industrial and Applied Mathematics. 2017: 257-269.
[43] COHEN S, MATIAS Y. Spectral bloom filters[C]// Proceedings of the International Conference on Management of Data. 2003: 241-252.
[44] MATSUMOTO Y, HAZEYAMA H, KADOBAYASHI Y. Adaptive bloom filter: A space-efficient counting algorithm for unpredictable network traffic[J]. IEICE Transactions on Information and Systems, 2008, 91(5): 1292-1299.
[45] FAN B, ANDERSEN D G, KAMINSKY M, et al. Cuckoo filter: Practically better than bloom[C]// Proceedings of International Conference on Emerging Networking Experiments and Technologies. 2014: 75-88.
[46] BENDER M A, FARACHCOLTON M, JOHNSON R, et al. Don't thrash: How to cache your hash on flash[J]. Proceedings of the VLDB Endowment, 2012, 5(11): 1627-1637.
[47] CHEN H, LIAO L, JIN H, et al. The dynamic cuckoo filter [C]// Proceedings of the International Conference on Network Protocols. 2017: 1-10.
[48] BRESLOW A D, JAYASENA N. Morton filters: Faster, space-efficient cuckoo filters via biasing, compression, and decoupled logical sparsity[J]. Proceedings of the VLDB Endowment, 2018, 11(9): 1041-1055.
[49] BRESLOW A D, JAYASENA N. Morton filters: fast, compressed sparse cuckoo filters[J]. The VLDB Journal, 2020, 29(2): 731-754.
[50] WANG M, ZHOU M, SHI S, et al. Vacuum filters: More space-efficient and faster replacement for bloom and cuckoo filters[J]. Proceedings of the VLDB Endowment, 2019, 13(2): 197-210.
[51] GRAF T M, LEMIRE D. Xor filters: Faster and smaller than bloom and cuckoo filters[J]. ACM Journal of Experimental Algorithmics, 2020, 25(1): 1-16.
[52] ZHANG F, CHEN H, JIN H, et al. The logarithmic dynamic cuckoo filter[C]// Proceedings of the 37th IEEE International Conference on Data Engineering. 2021: 948-959.
[53] FU P, LUO L, GUO D, et al. Jump filter: Dynamic sketch design for big data governance[J]. Journal of Software, 2023, 34(3): 1193-1212.
[54] HUANG K, YANG T. Additive and subtractive cuckoo filters [C]// Proceedings of the International Symposium on Quality of Service. 2020: 1-10.
[55] HUANG K, YANG T. Tagged cuckoo filters[C]// Proceedings of the International Conference on Computer Communications and Networks. 2021: 1-10.
[56] PANDEY P, BENDER M A, JOHNSON R, et al. A general-purpose counting filter: Making every bit count[C]// Proceedings of the International Conference on Management of Data. 2017: 775-787.
[57] PANDEY P, CONWAY A, DURIE J, et al. Vector quotient filters: Overcoming the time/space trade-off in filter design[C]// Proceedings of the International Conference on Management of Data. 2021: 1386-1399.
[58] DAYAN N, BERCEA I, REVIRIEGO P, et al. InfiniFilter: Expanding filters to infinity and beyond[J]. Proceedings of the ACM on Management of Data, 2023, 1(2): 1-27.

WANG Hancheng, born in 1998, postgraduate, is a student member of CCF (No. F2622G). His main research interests include data indexing and query optimization.

DAI Haipeng, born in 1985, Ph.D, associate professor, Ph. D supervisor, is a distinguished member of CCF (No. 19625S). His main research interests include data mining and mobile computing.

(责任编辑:喻藜)