Cuckoo Bloom Filter

Ju Hyoung Mun and Hyesook Lim*
Dept. of Electronic and Electrical Engineering
Ewha Womans University
Seoul, Korea
*Email: hlim@ewha.ac.kr

***Abstract*—A membership identification is a key functionality in many network applications. Various data structures have been introduced in order to support the efficient membership identification. Since a Bloom filter can provide simple but efficient membership checking, it is widely used in many network applications. However, the query results of Bloom filters can have false positives, which can degrade the search performance. Thus, reducing false positives of Bloom filters is challenging. In order to reduce the false positive rate, this paper proposes to use two sets of hash functions: primary and secondary. When the cell referenced by the primary hash function is occupied, the value of that cell is relocated to a cell referenced by the secondary hash functions like in the cuckoo hash. The proposed structure is evaluated using various sets, and the simulation results show that the proposed scheme can reduce the false positive rates.**

I. Introduction

The membership identification is an important task for many network applications. Various algorithms have been proposed to find the membership efficiently [1]–[3]. Particularly, a Bloom filter [1] can provide simple but efficient membership checking, it is widely used in many network applications, and many variants of the Bloom filter have been introduced for various purposes [4], [5], [7].

A Bloom filter [1] can identify whether an input is the member of a given set. Because of its simple structure, it has been widely employed in many network applications [4]. The Bloom filter is m-bit vector which programs the elements of a given set S by setting the bits indicated k hash functions. The membership checking process, which is called querying, can have the result of positive or negatives. Using the same k hash functions which are used in programming, the values of the bits are checked. If there is at least one 0 of the referenced bits, the result is negative, which means that the given input does not belong to the given set S. If all indicated bits are 1, the result is positive, which means that the input is the member of the set S. This membership identification result can be false only when the query result is positive. In order to find the correct membership, the original database should be accessed. Let n be the number of elements programmed to a m-bit Bloom filter, and then the false positive rate is obtained from Eq. (1). As shown in the equation, the false positive rate can be reduced by adjusting the size of the filter, the number of hash functions, and the number of the programmed elements.

$$f_s = (1-p)^k = \left(1 - (1 - \frac{1}{m})^{kn}\right)^k \tag{1}$$

A cuckoo hash table [2] is to store the set information using two hash functions, $h_1(x)$ and $h_2(x)$. When the bucket indicated by the first hash function $bucket[h_1(x)]$ is empty, then the input x is stored in that bucket. When the bucket $bucket[h_1(x)]$ is occupied by the element y, x is stored in that bucket and the element y is kicked out and try to find a new bucking using second hash function $h_2(y)$. When looking up the cuckoo hash table, both buckets are looked up and if one of the buckets stores the data then the input is matched. The cuckoo hash table relocates the previously occupied elements into a new location. A cuckoo filter [3] is a variant of the cuckoo hash table focuses on the membership identification. Instead of storing the full key, it only stores the fixed-length fingerprints of elements for membership checking. Moreover, each bucket consists of multiple entries. This cuckoo filter can identify the membership using small memory consumption.

In order to reduce the false positive rate, this paper proposes to apply the relocating method of cuckoo hashing to the Bloom filter. This paper proposes to use two sets of hash functions: primary and secondary. When the cell referenced by the primary hash function is occupied, the value of that cell is relocated to a cell referenced by the secondary hash functions like in the cuckoo hash. The proposed structure is evaluated using various sets, and the simulation results show that the proposed scheme can reduce the false positive rates.

II. The Proposed Work

The proposed cuckoo Bloom filter is to check the membership of a given input. The two functionalities are programming and querying same as the standard Bloom filter. The difference between the standard Bloom filter and the proposed structure is that the proposed structure uses two sets of hash functions: primary and secondary hash functions h_p, h_s. Each set of hash functions consists of k hash functions, therefore the total number of hash functions is $2k$. The secondary hash function is calculated using the hash value of the primary hash function and the fingerprint of an input as shown in Eq 2.

$$\begin{aligned} h_{pi}(x) &= hash_i(x) \\ h_{si}(x) &= h_{pi}(x) \oplus hash(fp_x) \end{aligned} \tag{2}$$

The example of querying is described in Figure 2. Same as programming, the k hash indices are computed by using the primary hash functions $h_{p1}(x), \cdots h_{pk}(x)$. If all referenced cells are not identical with the fingerprint of an input, fp_x.

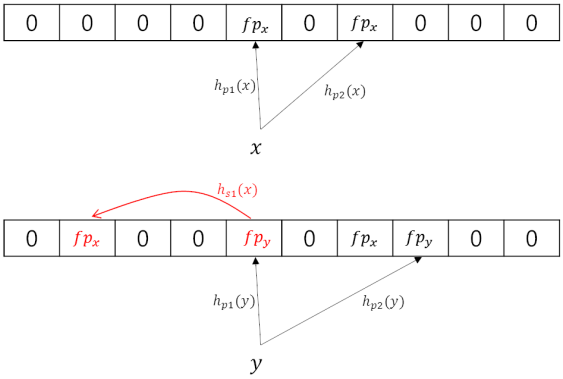

Fig. 1. The example of programming

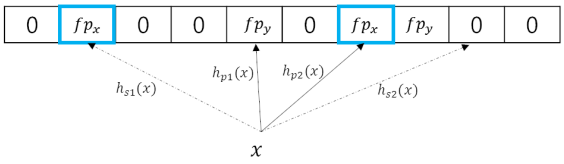

Fig. 2. The example of querying

Algorithm 1: The query procedure of the proposed structure

```
Function Query (x)
    for i=1 to k do
        if BF[h_pi(x)]!=fp(x) || BF[h_si(x)]!=fp(x)
         then
            return negative;
        end
    end
    return positive;
end
```

If the referenced cell has the different fingerprint, then the secondary hash function is used. When the cell referenced by the secondary hash function is identical with the fingerprint, then the input is identified as a positive. When the fingerprints of the cells indicated by the primary and secondary hash functions are not identical with the fingerprint of the input, then the input is negative. More detailed querying process is explained in Algorithm 1.

III. Performance Evaluation

The proposed cuckoo Bloom filter has been evaluated using URL data from ALEXA [8]. The sizes of the programming sets S are 2^{13}, 2^{15}, and 2^{17}, and the sizes of the querying sets U are the triple of the programming sets.

The performance of the proposed scheme is compared with the standard Bloom filter. The comparison of the false positive rate is shown in Table I. The false positive rate is an important metric because both the standard Bloom filter and the proposed cuckoo filter only have false positive errors. As shown in the table, the proposed cuckoo Bloom filter can reduce the false positive rate.

TABLE I
The false positive rate

$\|S\|$	Bloom filter	Proposed
2^{13}	0.091	0.080
2^{15}	0.093	0.083
2^{17}	0.093	0.083

IV. Conclusion

A membership identification is an important task in many network applications, and the Bloom filter has been widely employed because of its simple and compact characteristics. Since the Bloom filter can produce false positives results, reducing false positives is challenging. This paper proposes to apply the relocating technique of the cuckoo hashing . When the cell referenced by primary hash function is occupied, the value of that cell is relocated to a cell referenced by the secondary hash functions like in the cuckoo hash. The proposed structure is evaluated using various sets, and the simulation results show that the proposed scheme can reduced the false positive rates.

Acknowledgment

This research was supported by a National Research Foundation of Korea (NRF), 2017R1A2B4011254.

References

[1] B. Bloom, “Space/Time Tradeoffs in Hash Coding with Allowable Errors", Communications of the ACM 13:7 1970, pp.422-426
[2] R. Pagh, F. F. Rodler, “Cuckoo hashing," *Journal of Algorithms* vol. 51, no. 2, pp. 122-144, May 2004
[3] B. Fan, D.G. Andersen, M. Kaminsky, M.D. Mitzenmacher, “Cuckoo filter: practically better than bloom," *ACM CoNEXT*, 2014, pp. 75-88.
[4] S. Tarkoma, C. E. Rothenberg, and E. Lagerspetz, “Theory and Practice of Bloom Filters for Distributed Systems," *IEEE Communications Surveys and Tutorials*, vol. 14, no. 1, pp. 131-155, First Quarter, 2012.
[5] K.-Y. Whang, B. T. Vander-Zanden, and H. M. Taylor, “A lineartime probabilistic counting algorithm for database applications," *ACM Transactions on Database Systems*, vol. 15, no. 2, pp. 208-229, 1990.
[6] F. Bonomi, M. Mitzenmacher, R. Panigrah, S. Singh, G. Varghese, “Beyond bloom filters: from approximate membership checks to approximate state machines," *ACM SIGCOMM*, 2006, pp. 315-326.
[7] H. Lim, J. Lee, H. Byun, and C. Yim, “Ternary Bloom Filter Replacing Counting Bloom Filter" *IEEE Communications Letters*, vol. 21, no.2, pp.278-281, Feb. 2017.
[8] Alexa the Web Information Company, http://www.alexa.com/