Position-Aware Cuckoo Filters

Minseok Kwon
Rochester Institute of Technology, USA
jmk@cs.rit.edu

Vijay Shankar
Rochester Institute of Technology, USA
va3463@rit.edu

Pedro Reviriego
Universidad Antonio de Nebrija, Spain
previrie@nebrija.es

ABSTRACT

Cuckoo filters have been recently proposed as an efficient structure to perform approximate membership checks. In this paper, it is shown that the false positive rate of a cuckoo filter can be reduced by storing in addition to the fingerprint the information that tell us if a given fingerprint has been inserted in the first or the second bucket. This improvement of the cuckoo filter is denoted as Position Aware (PA) cuckoo filter.

CCS CONCEPTS

• **Networks → Network algorithms**; **Data path algorithms**; *Packet classification*;

KEYWORDS

Cuckoo filters, Bloom filters, packet classification, routers

ACM Reference Format:
Minseok Kwon, Vijay Shankar, and Pedro Reviriego. 2018. Position-Aware Cuckoo Filters. In *ANCS '18: Symposium on Architectures for Networking and Communications Systems, July 23–24, 2018, Ithaca, NY, USA.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3230718.3232103

1 INTRODUCTION

A cuckoo filter enables approximate membership checks by performing partial key cuckoo hashing [3–5]. The filter inserts the fingerprint of an element into one of the two different positions indexed by hash values computed based on the element and the fingerprint. The lookup of the element can be performed using the same two hash values. The filter also supports element removal, which is not supported in Bloom filters. Cuckoo filters achieve lower false positive rates, especially at high occupancies, making cuckoo filters an effective alternative to the original Bloom filters or its variants, for applications like IP lookup and information retrieval [1, 2].

A cuckoo filter is composed of an array of m positions each of which has b cells that can store fingerprints [3]. To search for an element x, the fingerprint of the element is first obtained as $fp = h_f(x)$, and two hash values are computed as $h_a(x)$ and $h_b(fp)$ where $h_f(x)$, $h_a(x)$ and $h_b(x)$ are hash functions. The element is then searched in the buckets indicated by the indexes $a_1 = h_a(x)$ and $a_2 = a_1 \oplus h_b(fp)$. Specifically, the elements in those buckets are read, and their fingerprints are compared with fp. If any of the elements are matched, the search returns positive; otherwise, the search fails. An element can be removed similarly: search for the element first, and once found, the fingerprint is removed. The most complex operation in a cuckoo filter is insertion and is similar to the one defined in cuckoo hashing [4].

The number of cells per bucket is typically $b = 4$. Then, on average, a search compares a fingerprint against other $8 \cdot o$ fingerprints when the filter occupancy is o as there are a total of eight cells in the two buckets searched for. Each fingerprint has a probability of giving a false positive of approximately 2^{-f} when f bits are used for a fingerprint. Therefore, the false positive rate, fpr, of the cuckoo filter is approximated to be:

$$fpr \approx \frac{8 \cdot o}{2^f} \quad (1)$$

In this paper, we present the Position Aware (PA) cuckoo filter that efficiently encodes whether a fingerprint is stored in its first or second position. This is used then to reduce the false positive rate compared to a cuckoo filter.

2 POSITION-AWARE CUCKOO FILTERS

Suppose that each fingerprint is marked with one bit that indicates whether the element is inserted using a_1 or a_2, e.g., 0 for a_1 and 1 for a_2, the first and second positions, respectively. With this extra one bit, if we are accessing bucket a_1, we only need to compare elements in the first position, and if we are accessing bucket a_2, we only need to compare elements inserted in the second position. This change reduces the false positive rate by half as we compare with only a half of the elements. It, however, comes at the expense of one additional bit per fingerprint, which is equivalent to increasing the false positive rate by two times (see Eq. 1). Luckily,

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).
ANCS '18, July 23–24, 2018, Ithaca, NY, USA
© 2018 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-5902-3/18/07.
https://doi.org/10.1145/3230718.3232103

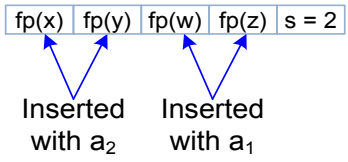

Figure 1: Marking of elements on a bucket in the position-aware cuckoo filter.

the position information can be encoded more efficiently as the elements can be grouped together depending on the hash function used to insert them. For instance, the elements of the first position are grouped to the right, and the ones inserted in the second position to the left. More formally, the search algorithm runs as follows. Let s be the variable used to store the position information. We first read s, and if we are accessing the elements in the first position, we only compare the fingerprints from $s+1$ to 4, where bucket size is four. If we are accessing the second position, then we only compare the first s fingerprints. This is functionally equivalent to having a bit per fingerprint to mark the position used to insert the element, but requires fewer bits. For example, we can code 0, 1, 2, and 3 elements inserted using a_2 when using two bits for s, and the only case not covered is having four elements inserted with a_2, which rarely arises. In Fig. 1, we illustrate an example of a bucket in the filter when two bits are used for s and $s = 2$. In the example, $fp(x)$ and $fp(y)$ are inserted using a_2, and $fp(w)$ and $fp(z)$ using a_1. If the bucket is checked on the first access of a query, we only need to read $fp(w)$ and $fp(z)$; if the bucket is accessed on the second, we only need to read $fp(x)$ and $fp(y)$.

The primary benefit of the PA cuckoo filter is to reduce the false positive rate in half with only additional two bits for s per bucket. One potential drawback is the restriction that four elements cannot be inserted with a_2 in each bucket. This however, is unlikely to occur and has a negligible impact as seen in the results presented in the following section.

3 EVALUATION

The PA cuckoo filter as well as a standard cuckoo filter were implemented to test their performance. In our implementation, we set the number of elements allowed in the second position to be three.The parameter values in our experiments were 2000 or 16000 buckets for the table size, eight or twelve bits for the fingerprint, one thousand cuckoo kickoffs to declare an insertion failure and four cells per bucket.

In the first experiment, the false positive rates of the PA cuckoo filter and a standard cuckoo filter are measured. For

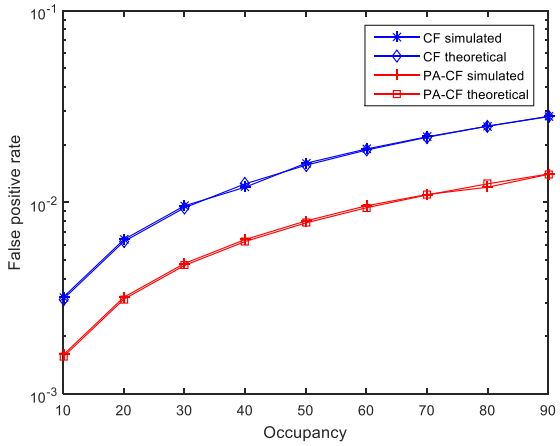

Figure 2: False positive rates for the position-aware and standard cuckoo filters when f = 8 bits.

this, the tables are filled with elements until a specified occupancy value is reached. Then searches for elements not present in the filters are performed 1,000 times, and the false positive rates are computed. For each configuration, the test is repeated one million times, and the average results are reported. Fig. 2 shows the false positive rates for $fp = 8$ bits per fingerprint for tables with 16,000 buckets for a standard cuckoo filter and the PA cuckoo filter with three elements allowed in the second position. As shown in the figure, the PA filter has approximately a half of the false positive rate of the cuckoo filter. These false positive rates also match the theoretical estimates. Similar results were obtained for the rest of the configurations. As implied previously, one concern with the PA cuckoo filter is the possibility of table underutilization due to the limit of the position bits. To examine this concern, in a second experiment, we insert elements until an insertion fails. When the failure occurs, the number of elements in the table is recorded. We run the experiment repeatedly again one million times for randomly generated input data. The largest difference in occupancy observed was of less than 0.2% for the table and fingerprint sizes tested. This confirms that the impact on occupancy is small.

4 CONCLUSIONS

This paper has presented the Position-Aware (PA) cuckoo filter, an enhancement to the standard cuckoo filter to reduce its false positive rate. The PA filter encodes the two positions that are used to store fingerprints for each bucket, this, on the average, reduces the number the false positive rate by half. The PA cuckoo filter has been implemented and tested to demonstrate that it reduces the false positive rate with a negligible impact on memory occupancy.

REFERENCES

[1] B. Bloom. 1970. Space/Time Tradeoffs in Hash Coding with Allowable Errors. *Communications of the ACM* 13 (1970), 422–426. Issue 7.

[2] A. Broder and M. Mitzenmacher. 2004. Network Applications of Bloom Filters: A Survey. *Internet Mathematics* 1 (January 2004), 485–509. Issue 4.

[3] B. Fan, D. Andersen, M. Kaminsky, and M. Mitzenmacher. 2014. Cuckoo Filter: Practically Better Than Bloom. In *Proc. of ACM CoNEXT*. 75–88. DOI: http://dx.doi.org/10.1145/2674005.2674994

[4] R. Pagh and F. Rodler. 2004. Cuckoo Hashing. *J. Algorithms* 51, 2 (May 2004), 122–144.

[5] S. Pontarelli and P. Reviriego. 2016. Cuckoo Cache: A Technique to Improve Flow Monitoring Throughput. *IEEE Internet Computing* 20 (2016), 46–53. Issue 4.