A Position-Aware Cuckoo Filter for Robust Lookup

Vijay Shankar Annadurai
Va3463@rit.edu
Rochester Institute of Technology
Rochester, NY 14623
USA

Abstract—A cuckoo filter is a compact data structure used in set membership tests. It overcomes the certain drawbacks of bloom filters like deletion and space overhead. Unlike bloom filters, cuckoo filters store the hash values of the numbers and not the actual numbers. However, while searching for an element, a cuckoo filter searches in all the cells of the bucket which increases the lookup time and the false positive rate (FPR). A new approach is proposed to overcome the drawbacks of a cuckoo filter called a position-aware cuckoo filter (PACF), by introducing an extra cell called position cell in each bucket to count the number of elements inserted in their second hash location.

I. INTRODUCTION

Software-based IP packet forwarding techniques play a vital role in the overall performance of the internet. This performance can be achieved with the help of approximate set membership tests to find whether the IP network prefix is present in the large set of IP network prefixes. A cuckoo filter is a data structure used for set membership queries. It uses a probability-based hashing technique to address the hash collisions. It is used in many real-time applications for faster lookup as the time complexity for lookup is $O(1)$. Due to the possibility that the performance of a cuckoo filter can be enhanced [2], a new filter called a position-aware cuckoo filter is proposed.

PACF is a variant of a cuckoo filter which reduces the overall false positive rate by using extra bits called position bits in each bucket. Each element can be inserted in one of the two hash locations in the filter and position bits will keep track of the number of elements inserted in their second hash position. While searching for an element, the element in its first position will look only the elements inserted in their first positions based on the position bits information and another set of elements are not searched and similarly, element in its second position will only check for elements inserted in their second positions. This considerably reduces the overall FPR as we are searching only one set of elements and ignoring another set of elements.

PACF and the traditional cuckoo filter is implemented for the filters containing 2K and 16K buckets and each bucket has four cells to store the fingerprints of the elements and a position cell to store position bits in PACF. The simulation is carried out for different fingerprint size and different number of input elements and their FPR for each simulation is measured and compared with the traditional cuckoo filter. The performance of PACF is analyzed based on FPR for the various simulations and it should be half that of the traditional cuckoo filter.

II. TRADITIONAL CUCKOO FILTER (TDCF)

A cuckoo filter is a probabilistic filter which uses a hashing technique to store and retrieve the elements. It answers the question, “is item $X$ in set $Y$” by:

definitely no", or

“probably yes” with probability $\epsilon$ to be wrong $\epsilon$ – false positive rate.

In a cuckoo filter, the hash function is used to calculate each element’s hash value called fingerprint. Each fingerprint can be inserted in either one of the two hash locations in the filter. Since the filter stores only the hash value, the original value cannot be retrieved. So, to find out the alternate location of an element the cuckoo filter uses partial key cuckoo hashing technique [3].

If the first location is occupied by other fingerprints, then the second location can be calculated by taking the hash value of the fingerprint xor with the first location. If both the locations are occupied, then the preoccupied fingerprint from one of the locations will be kicked out to its another location so that the new fingerprint will be inserted. Else, this procedure continues till the kicked-out fingerprint is inserted or max kickoff value is reached and the fingerprint is not inserted.

![Figure 1: Element Insertion in the cuckoo filter.](image)

The table structure of a traditional cuckoo filter is shown in Figure 2. Each row in the table is called bucket which contains four cells which are initially marked as empty cells. When an element arrives, the hash value of that element is calculated and it is inserted into one of the empty cells in either of its first or second position.
While searching for an element, the cuckoo filter searches all the fingerprints of the bucket in its first and second location till it finds the required fingerprint. This increases the lookup time to find an element as well as FPR of the filter.

**III. POSITION-AWARE CUCKOO FILTER (PACF)**

In PACF, the element insertion is same as the traditional cuckoo filter by calculating the fingerprint of an element using the hash function and inserting it in either one of the two locations in the filter. In addition, each bucket contains an extra cell called position cell which keeps count of the number of fingerprints inserted into the bucket in their second hash location.

During lookup, the search element’s fingerprint is calculated and the fingerprint in its first hash location will look for the elements in the corresponding bucket that are inserted in their first position based on the position cell information and so the elements that are inserted in their second position will not be searched. Similarly, the search element in its second position will only look for the elements inserted in their second position and another set of elements is ignored.

**IV. IMPLEMENTATION**

The traditional cuckoo filter and PACF is implemented as a proof of concept. The programming language used is Java and the JDK version is 8. The bucket size is set as 2,048 and 16,384 with each bucket containing 4 cells in which elements can be inserted and a position cell in PACF to count the number of elements inserted in their second position. The size of each cell depends on the size of the fingerprint to be stored. The size of the position cell is determined as \(\log_2(X)\) bits per bucket where \(X\) is the number of fingerprints that can be inserted per bucket.

**V. RESULTS**

Several simulations are performed for PACF by varying the parameters like fingerprint size, bucket size, the number of elements occupied in the second hash position and their false positive rate is analyzed for each simulation and compared to the traditional cuckoo filter.

**Test 1:** An input file containing 6,500 elements are inserted into both the filters and their FPR is recorded by looking at one million elements.

<table>
<thead>
<tr>
<th></th>
<th>Traditional CF</th>
<th>PACF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inserted Elements</td>
<td>6,499</td>
<td>6,499</td>
</tr>
<tr>
<td>Lookup Elements</td>
<td>1 Million</td>
<td>1 Million</td>
</tr>
<tr>
<td># of False Positives</td>
<td>50,125</td>
<td>25,752</td>
</tr>
<tr>
<td>False Positive Rate</td>
<td>0.057</td>
<td>0.025</td>
</tr>
</tbody>
</table>

Table 1: FPR comparison between PACF vs. the traditional cuckoo filter.

Table 1 shows the false positive rate of PACF and the traditional cuckoo filter obtained from setting the fingerprint size of seven bits and max kickoff value as 100.

**Test 2:** The bucket size is kept as 2,048 with each bucket containing 4 cells to store fingerprints and one position cell and setting the fingerprint size to be seven bits. In each simulation, a range of inputs from 10% until 90% of 8,192 elements in 10% increments is inserted and is looked at one million elements and their FPR is recorded. The max kickoff value is set as 1,000 so that the occupancy of the filter can be increased.
Test 3: Test 2 is repeated with the fingerprint size of eight bits. By increasing the fingerprint size to eight bits, the hash value of an element can be in the range of 0 to 254 (255 to mark empty cells).

Hence, the false positive rate for a fingerprint size of seven bits (Fig. 4) is twice as much as the false positive rate for a fingerprint size of eight bits (Fig. 5) for the same number of input elements.

Test 4: The false positive rate and the occupancy percentage of PACF are measured for the different number of elements occupied in the second position such as 3, 2, 1 and 0 respectively by setting the fingerprint size to eight bits for the bucket size of 2,048 and 16,384 and compared with the traditional cuckoo filter.

![Figure 4: FPR vs. the number of input elements comparison for PACF and the traditional cuckoo filter for the fingerprint size of seven bits.](image)

![Figure 5: FPR vs. the number of input elements comparison for PACF and the traditional cuckoo filter for the fingerprint size of eight bits.](image)

<table>
<thead>
<tr>
<th>Elements in second position</th>
<th>Finger print size: 8 bits</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Bucket Size: 2K</td>
</tr>
<tr>
<td></td>
<td>Occupancy avg.</td>
</tr>
<tr>
<td>3</td>
<td>97.05%</td>
</tr>
<tr>
<td>2</td>
<td>96.72%</td>
</tr>
<tr>
<td>1</td>
<td>94.04%</td>
</tr>
<tr>
<td>0</td>
<td>13.48%</td>
</tr>
<tr>
<td>TDCF</td>
<td>97.14%</td>
</tr>
</tbody>
</table>

Table 2: FPR and the occupancy percentage of PACF and TDCF for the fingerprint size of eight bits and the bucket size of 2K and 16K.

The test 4 and 5 contains a total of 20 simulations of 1 million executions each with the max kickoff value of 1,000 for all the simulations. The results obtained from the simulations (Table 3 and 4) shows that the false positive rate is roughly same irrespective of the bucket size and it varies based on the fingerprint size. Also, the false positive rate of PACF is reduced to half that of the traditional cuckoo filter.

VI. FUTURE WORK

One of the applications of a cuckoo filter is faster IP lookup because of its hashing principle [1]. As part of the future work, PACF will be implemented in the real-time applications such as a router for IP network prefix lookup. Initially, PACF will be configured in open source router and its metrics are obtained. These metrics include memory usage, false positives, lookup time of the forwarding table [2]. Once the measurements of the performance metrics are analyzed, then PACF will be deployed in the real-time router.

VII. CONCLUSION

The main objective of reducing the false positive rate of the traditional cuckoo filter to half is achieved by introducing an extra cell called position cell in each bucket to store the count of the number of elements inserted into that bucket in their second position. The extra space overhead of log₂(X) bits per bucket where X is the number of elements per bucket is negligible when compared to the overall memory usage of a cuckoo filter.

VIII. REFERENCES

