Fast Dynamic Multiset Membership Testing Using Combinatorial Bloom Filters

Fang Hao Murali Kodialam T. V. Lakshman Haoyu Song
Bell Laboratories, Alcatel Lucent
791 Holmdel-Keyport Road, Holmdel, NJ 07733, USA
INFOCOM 2009
Outline

- Problem Description and Its Application
- Preliminary - Bloom Filter
- Related Work
- Approaches Using Multiple Filters
  - Insertion
  - Checking
- Theoretical Analysis
Let $S$ denote a set comprising of $n$ elements.

Each element of the set belongs to one of $\gamma$ groups.

Given some $x \in S$ we use $g(x) \in G$ to represent the group that contains $x$.

Objective of this paper is:

- Given any $x$ we have to determine $g(x)$.
- If $x \not\in S$ then we want to return $g(x) = \perp$
An Example

- \( S = \{2, 4, 6, 8, 10, 1, 3, 5, 7, 9\} \)
- \( g_1 = \{2, 4, 6, 8, 10\}, \ g_2 = \{1, 3, 5, 7, 9\} \)
- \( g(8) = 1; \ g(7) = 2; \)

- Given \( x = 11 \), since \( x \notin S \), then return \( g(11) = \bot \)

- Given \( x = 9 \), then return \( g(9) = 2. \)
Network packet processing, such as forwarding and measurement, often involves some sort of table lookups.

- In high speed networks, these applications require fast and deterministic lookup performance.

Packet classification [3].

Bloom filters play an important role in supporting many of these applications.
What is Bloom Filter?

- Bloom Filter is a bit-based data structure which used to decide set membership.
- It can answer the question of whether an element is in the table (set) or not.
- It is space efficient.
- It may Cause Small Number of False Positives
- Based on Hashing Ideas.
Working-insertion

- $m$ bit vector
- $k$ random hash functions

All elements of a table (set) are compressed into a bit vector.
\( m \) bit vector

\( k \) random hash functions

If all \( V(h_1(y))=V(h_2(y))=\ldots=V(h_k(y))=1 \), then answer \( y \) is in the table (set), Otherwise \( y \) is not in the table.
$x$ is not in the table (set). However, its bits are already set, the answer is: $x$ is in the table (set), this is wrong. We call this *false positive*.
Analysis

- $k$ is the number of hash functions.
- $n$ is the number of elements in a table (set).
- $m$ is the number of bits in vector.
- The probability that one bit (position) is zero is:

\[
(1 - 1/m)^{kn} \approx e^{-kn/m} = e^{-\gamma}, \quad \gamma \equiv \frac{nk}{m}
\]

- Probability of a false positive is: $(1 - e^{-\gamma})^k$
Given a false positive error bound $\delta$, Find a proper $n$, such that probability of a false positive $(1-e^{-\gamma})^k < \delta$, the $n$ is called capacity of the BloomFilter with $m$ bits memory.

$$(1-1/m)^{kn} \approx e^{-kn/m} = e^{-\gamma}, \quad \gamma \equiv \frac{nk}{m}$$
Choose proper $k$

If $n$, the number of elements of a table (set) is known in advance, we can choose a proper $k$ such that the probability of a false positive $(1-e^{-\gamma})^k$ is minimized.

$$k = (m/n)\ln(2)$$

$$(1-1/m)^{kn} \approx e^{-kn/m} = e^{-\gamma} , \quad \gamma \equiv \frac{nk}{m}$$
One approach to address this problem is to have a separate Bloom filter for each group id. There are two drawbacks with this approach.

- We have to know the number of elements that will hash into each filter. Since this value is not known in advance, there is no good way to partition the available memory.
- When we are checking, we have to check against all $t = |G|$ Bloom filters. This can be very expensive if the number of groups is large.
For each element, we must know which Group it belongs to when inserting.
For example, $g(y)=1$; $g(x)=2$;
When we are checking, we have to check against all $t = |G|$ Bloom filters. For example, give $y$, then we answer $g(y) = 1$. 

![Diagram showing Bloom filters with functions $h_1(y)$, $h_2(y)$, and $h_3(y)$, and Bloom filters $g_1$ and $g_2$.]
Related Work: Coded Bloom Filter[6, 9]

- To have a fixed number $f$ of Bloom filters and code the group id into these $f$ Bloom filters.
  - Each group index is coded into a $f$ bit binary vector and an element of the group is hashed into each of the filters where the code has a one.
  - In order to check the set that includes a given element $x$, we hash $x$ into each of the $f$ filters and determine the filters where we get a one. This gives the code of the group that contains $x$. 
Working-insertion

y’s group index code:
y’s group index code:
Drawback-Coded Bloom Filter[6, 9]

- **Sizing the Filters:** The number of elements that will be hashed into a given filter is a function of the sum of the number of elements in the group id that map to that filter. Since this is not known in advance, it is not possible to allocate the given memory between the different filters.

- **Decoding with False Positives:** If an arbitrary coding is used to map group ids into the filters, then there will be misclassification of group id due to false positives in one or more filters.
• **Misclassification:** In this case the data structure outputs the wrong group id for some \( x \in S \). In other words, given some \( x \in S \) whose group id is \( g(x) = k \in G \). The COMB instead outputs \( g(x) = j \neq k \). A particular case of misclassification is when given some \( x \in S \), we declare that \( g(x) = \bot \).
An example for misclassification

y's real group index code: [0 1 0 0]
y's reported group index code: [0 1 1 0]
Approaches Using Multiple Filters

Core idea:

- Coded Bloom Filter, but the code for all the groups have the same number of ones, say $\theta$. 
Working - insertion

y’s group index code:

$\theta = 2$
Working-checking

y’s group index code: \[ \begin{array}{cccc}
\theta &=& 2 \\
0 & 1 & 1 & 0 \\
1 & 2 & 3 & 4 
\end{array} \]
**Checking:** In order to determine the group id of a given $x$, we hash $x$ using all $f$ hash sets ($fh$ hashes in all). We initialize a $f$ bit binary vector $v$ to zero. Bit $i$ in vector $v$ is set to one if all $h$ hash functions in hash set $i$ results in a one. If $w(v) < \theta$ then we declare $g(x) = \bot$. If $w(v) > \theta$, then we declare a classification failure. If $w(y) = \theta$, then the group id of $x$ is then $C^{-1}(v)$. There are $fh$ memory accesses for checking.
Fixed weight codes can eliminate the problem of misclassification.

Note that if \( x \in S \), then there will certainly be \( \theta \) ones when we check for membership.

- If there are additional ones due to false positives, then we immediately declare our inability to determine the group id for \( x \).
An example for misclassification

y is in the table (set)
y’s real group index code: 

y’s reported group index code: 

False positive
Theoretical Analysis

- The false positive probability $p$ for any $\theta$
  Bloom Filter?

- The false positive probability for $(f, \theta)$?

- The probability of classification failure probability?

- How many groups can be represented by $(f, \theta)$?
The false positive probability $p$ for any $\theta$ Bloom Filter

\[ p = \left( 1 - \left( 1 - \frac{1}{m} \right)^{\theta nh} \right)^h \]

\[ \approx \left( 1 - e^{-\frac{\theta nh}{m}} \right)^h \]
The false positive probability for \((f, \theta)\)

False Positive Probability

\[
\begin{align*}
\text{False Positive Probability} & = \binom{f}{\theta} p^\theta (1 - p)^{f - \theta} \\
& \leq \binom{f}{\theta} p^\theta \\
& \leq f^\theta p^\theta = (fp)^\theta
\end{align*}
\]
The probability of classification failure probability

\[
\Pr[\text{Classification Failure}] = 1 - (1 - p)^{f-\theta} \\
\leq (f - \theta)p \leq fp
\]
How many groups can be represented by \((f, \theta)\)?

\[
\rho(f, \theta) = \begin{pmatrix} f \\ \theta \end{pmatrix}.
\]