Calculating Set Difference Among Sets Stored in Bloom Filters

Neha Sengupta Pratham Shah Shubham Jain

November 26, 2023

1 Introduction

Bloom Filters [BLO70] are data structures that can efficiently answer set-membership queries. Swamidass & Baldi (2007) [SB07] demonstrated how size of union and intersection of two sets represented using Bloom Filters can be estimated. However, a very fundamental problem of calculating set difference among two sets stored as Bloom Filters remains solved. In this paper, we give a method to estimate elements in the set difference among two sets represented as Bloom Filters. *Problem Statement.* If we are given two sets $S1$ and $S2$ from a namespace U stored in Bloom Filters $B1$ and $B2$ respectively, then we try to estimate elements that are in $S1$ but not in $S2$. $SolutionOverview$. We exploit Bloom Sample Tree [SBBR17] to represent the entire namespace. We go down the tree estimating number of common elements between nodes of the tree and $S1$ and $S2$. If no common element with $S1$ is found, search along that path is pruned, if no common element with $S2$ is found, the node is added into the result and if there are common elements with both sets are found, we move a level down on the path.

2 Algorithm

The following algorithm is used to estimate elements in the set difference among two sets S and C represented as Bloom Filters using Bloom Sample Tree B which represents namespace where B_{ij} represents j^{th} node of i^{th} level.

Algorithm 1 $Set\_diff(B_{ij}, S, C)$

```
procedure SET_DIFF($B_{ij}, S, C$)
    if $isLeaf(B_{ij})$ then
        return all elements of $B_{ij}$ that are in $S$ and not in $C$
    else if $B_{ij} \cap S = \phi$ then
        return $\phi$
    else if $B_{ij} \cap C = \phi$ then
        return $S$
    else
        return $Set\_diff(B_{i+1,2j}, S, C) \cup Set\_diff(B_{i+1,2j+1}, S, C)$
```

Algorithm 2 $Set\_difference(B_{ij}, S, C)$

```
procedure SET_DIFFERENCE($B_{ij}, S, C$)
    return $Set\_diff(B_{0,0}, S, C)$
```

3 Experimental Evaluation

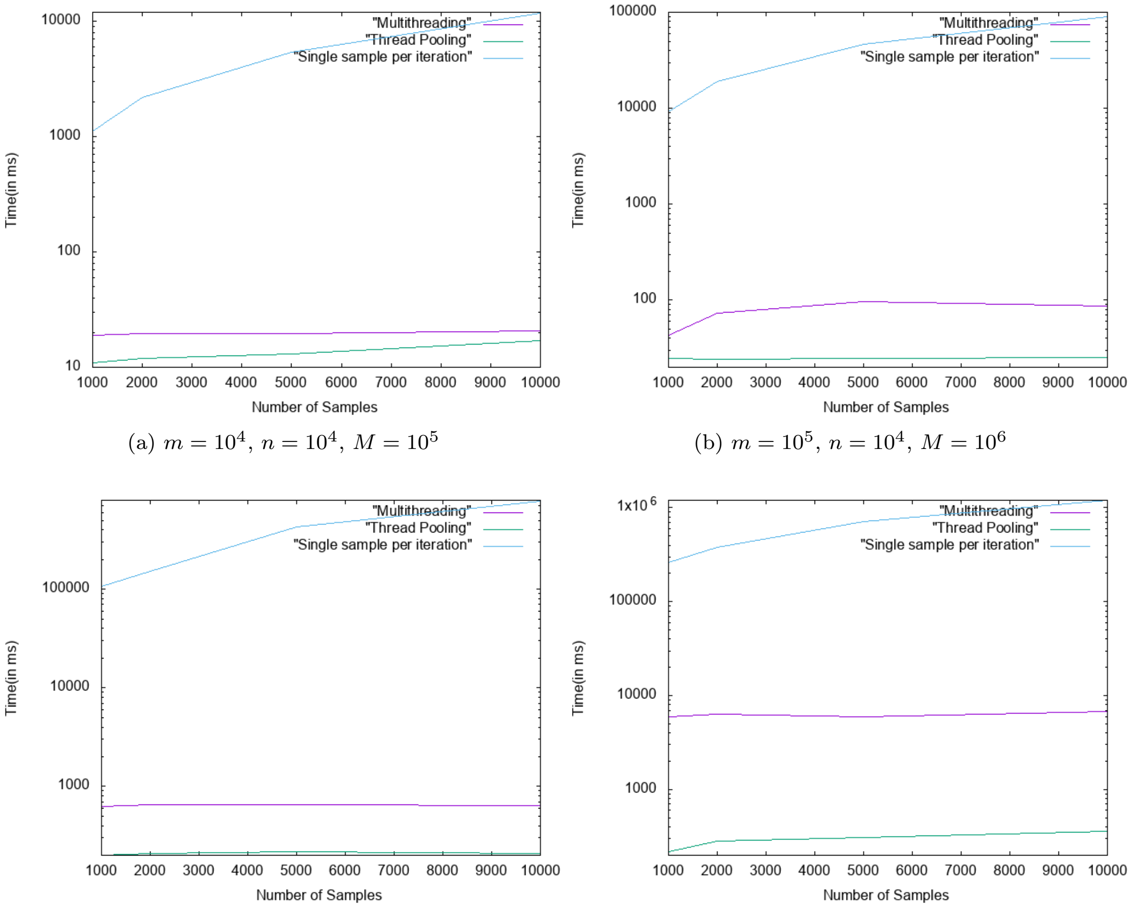
In this section, we describe the experiments we did with different approaches using static namespace. We compared running time of different approaches to generating a single sample required number of times.

3.1 Setup

We used a static synthetic namespace to compare running time of the different approaches. We varied our namespace from 10^5 to 10^7. We also varied number of samples to be generated from 1000 to 10000.In the combined approach, we have used 256 threads for the below experiments.
Query Sets In the experiments we performed, we generated query sets uniformly at random from the the entire namespace.
Algorithms We compared running time of approaches 2 and 3 and compare it to running time of generating a single sample required number of times.
Metrics We report the running time of algorithms by running them multiple times and taking average over the running times.

4 Results

Figure 1 shows the average time taken to sample multiple elements from a set using a Bloom Sample Tree using different approaches. The experiment demonstrate that creating multiple threads in advance and then using each to handle multiple nodes is the best strategy of the proposed ones.



(a) $m = 10^4$, $n = 10^4$, $M = 10^5$

(b) $m = 10^5$, $n = 10^4$, $M = 10^6$

(c) $m = 10^6$, $n = 10^4$, $M = 10^7$

(d) $m = 10^6$, $n = 10^4$, $M = 10^8$

Figure 1: Time taken for different approaches

5 Conclusions

In this paper, we described different methods of sampling multiple elements using a Bloom Sample Tree and compared the time taken by them in different settings. Our experiments demonstrated that the most suitable approach out of the approaches mentioned is to create multiple threads prior to sampling, and then using each of the threads to handle multiple nodes in the Bloom Sample Tree.

References

[BLO70] BURTON H. BLOOM. Space/time trade-offs in hash coding with allowable errors. 1970.

[SB07] S. Joshua Swamidass and Pierre Baldi. Mathematical correction for fingerprint similarity measures to improve chemical retrieval. 2007.

[SBBR17] N. Sengupta, A. Bagchi, S. Bedathur, and M. Ramanath. Sampling and reconstruction using bloom filters. *Data Engineering (ICDE), 2017 IEEE 33rd International Conference*, pages 195–198, 2017.