Bloom Filters Benchmarking

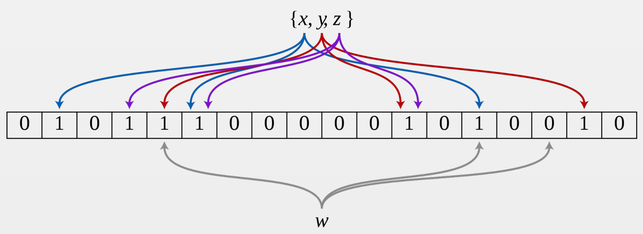
Tarun Khajuria, Mohamed Abdelrahman
University of Tartu, Institute of Computer Science

Project Repository

Introduction

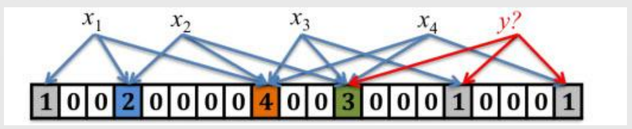
Bloom filter is a probabilistic data structure that is similar to hash tables and allow the trade-off between false positive rate of querying data and the storage space reduction.

Figure below shows the design of a **standard bloom filter** where the **size** of the table is equal to *18*, three **hash functions x, y, z** are used to map and store values in the table, and element w is queried, where one of the three output values of the hashing functions isn't marked in the table. So, element isn't stored in filter.



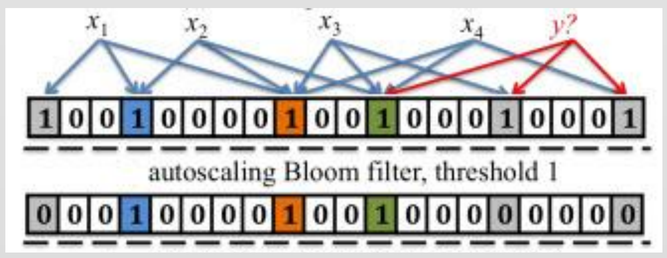
Counting Bloom Filter

This filter supports deletion by allowing value of each cell in the table to be > 1. So, mapped values by the hashing functions are incremented by 1 in case of element storage, or decreased in case of deletion.



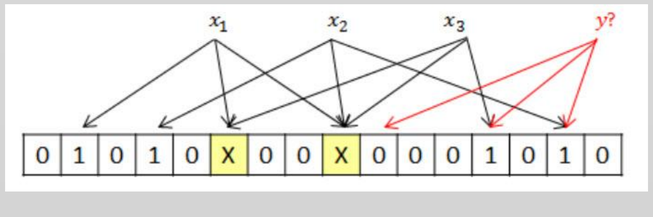
Auto-Scaling Bloom Filter

It is a generalized Counting Filter with introducing two new parameters which are the **binarization threshold, and decision threshold**. it is designed with a fixed size hash table and fixed number of hash function but for a wide range number of elements to be stored keeping almost a fixed ratio of False positives and False Negatives.



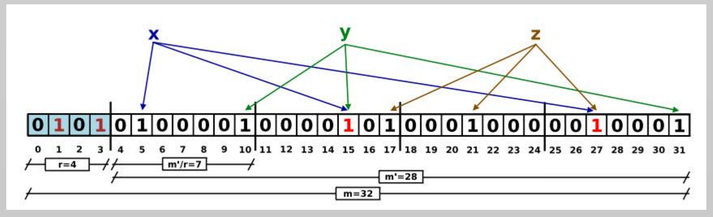
Ternary Bloom Filter

It is an extension over Counting filter, where each cell takes values 0, 1, and **X** for any value >1. It defines new **indeterminable** (can't be identified as positive or negative) events as a result of querying items when all cells corresponding to queried item are having **X**. However, it is very efficient in memory.



Deletable Bloom Filter

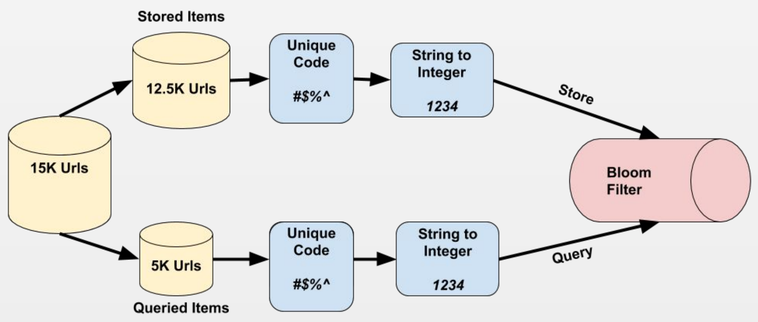
It is an extension of the simple bloom filter with **a separate section** of the table used to maintain information about the sections having no collisions, **allowing deletions** in those sections by simply unsetting a bit. This ensures deletion with **no false negatives**. This implementation is efficient in memory over counting type filter but the allowed deletions decreases as the elements in the table increases.



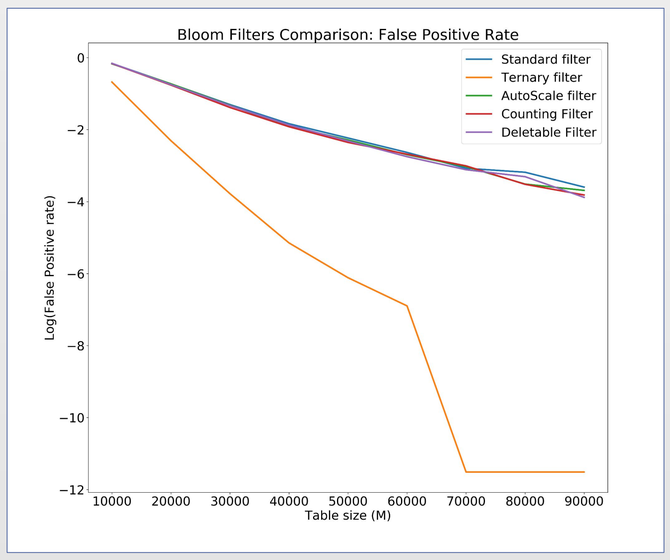
Methodology

As our application is related to URL shortening:

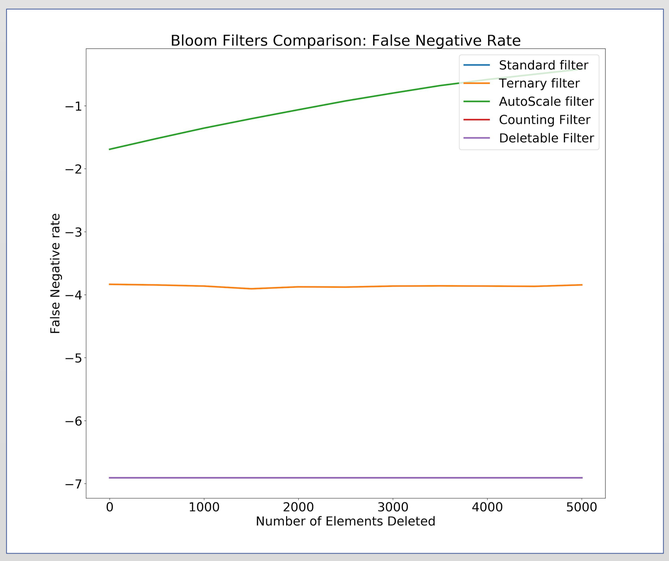
1) a dataset of **15k URLs** were collected.
2) Unique Random Codes are generated for each URL.
3) Each code is converted to a unique integer using ASCII values of characters in code.
4) **12.5k codes are going to be stored** in each filter and **5k codes are going to be queried** in the filter where 50% of them only are already stored in the filter to count false negatives while the other 50% are used to count false positives.

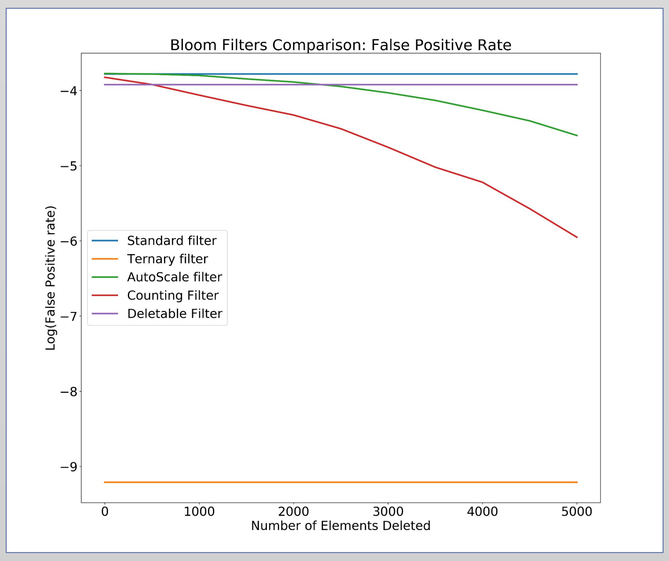


The plot below shows logarithm of the number of false positives from each filter type Vs the filter size.



The plots below shows logarithm of the number of false positives and false negatives from each filter type Vs number of elements deleted from filters.

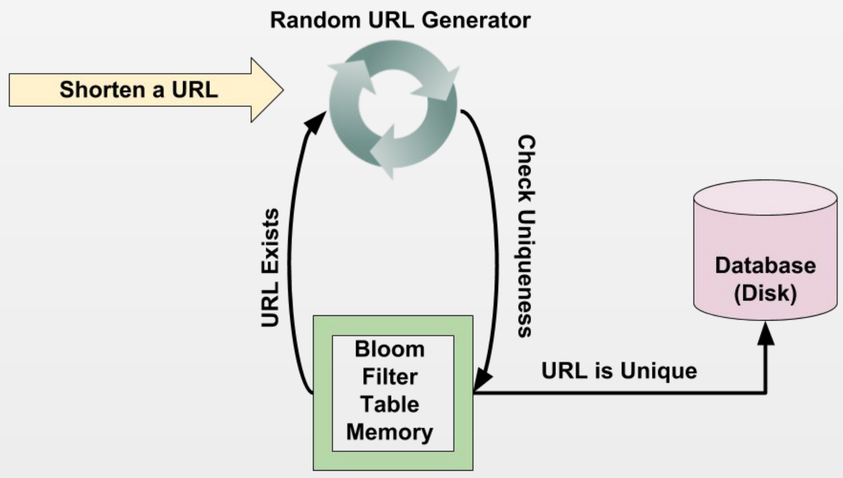




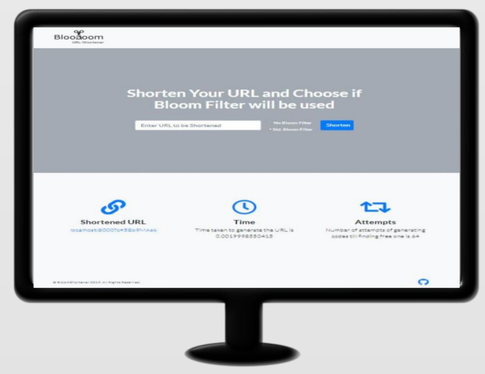
- Ternary Filter has many **indeterminable events** which leads to least FPR, and FNR.
- Standard Filter has almost similar FPR like other filters and it doesn't support deletion. So, no False negatives.
- AutoScaling Filter has the largest FNR but by deleting more elements its FRP decreases. This can be tuned by having different decision threshold, and binarization values.
- Counting Filter has no FNR, and also least FPR by deleting more elements from the filter. However, the filter size is the largest.
- Deletable Filter has no false negatives and almost similar FPR as Standard Filter. The allowed deletions are smaller than other filters but it has less size than Counting Filter.

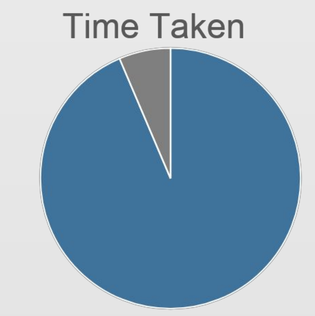
URL Shortening

One of the applications that can use Bloom Filters is the URL Shortening Applications. The process is to make a call to the server, which generates a fresh URL and sends it back. A bloom filter can be used to tell if this URL has already been generated earlier, and keep generating new ones till it returns false. As the filter is in memory, this tends to be cheaper than querying a database.



We have implemented a Web Application for URL shortening. You can try it in both cases of using Standard Bloom Filter or directly accessing the database and measure the time difference. Our application shows that we get faster performance using Std. Bloom Filter by around 14.6 times which will get much more by filling the database more with URLs.





Conclusion

As each type of the filter has its own characteristics, they can be used in wide range of applications according to allowed memory sizes, FNR, FPR, etc.

Filter Type	Table Size	Deletion	Indeterminable Events	Errors Characteristics
Standard	N bits	NO	NO	No FNs
Ternary	2N bits	YES with a limit	YES	Least FPs, FNs
Counting	O(N) C.N bits	YES	NO	No FNs
Deletable	O(N) D.N bits D < C	YES with a limit	NO	No FNs
Auto-Scaling	O(N) A.N bits A < C	YES	NO	Tuned FPs VS FNs relation

In case of URL Shortening, we have found that best filters to be used is Standard filter to avoid FNs, have large table size with less memory and there is no essential need for supporting deletions.

References

- Dillinger P.C., Manolios P., ***Bloom Filters in Probabilistic Verification. In: Hu A.J., Martin A.K. (eds) Formal Methods in Computer-Aided Design.*** FMCAD 2004. Lecture Notes in Computer Science, vol 3312. Springer, Berlin, Heidelberg
- Lim H, Lee J, Byun H, Yim C. ***Ternary Bloom Filter Replacing Counting Bloom Filter***. IEEE Communications Letters. 2017; 21: 278–281.
- Denis Kleyko, Abbas Rahimi, Evgeny Osipov: ***Autoscaling Bloom Filter: Controlling Trade-off Between True and False Positives.*** CoRR abs/1705.03934, 2017
- C. E. Rothenberg, C. A. B. Macapuna, F. L. Verdi, and M. F. Magalhaes, ***"The deletable bloom filter: a new member of the bloom family"***, IEEE Communications Letters, vol. 14, no. 6, pp. 557–559, 2010