Concurrent Skip List

Abstract ([pugh-skiplists1990.pdf (cmu.edu)](https://15721.courses.cs.cmu.edu/spring2016/papers/pugh-skiplists1990.pdf))

This paper describes a new approach to providing efficient concurrent access to a dynamic search structure. Previous approaches have attempted to solve this problem using search trees (either balanced or unbalanced). We describe methods for performing concurrent access and updates using skip lists. Skip lists are a probabilistic alternative to balanced trees that provide much of the simplicity of unbalanced trees, together with good worst-case expected performance. In this paper, we briefly review skip lists, describe simple methods for concurrent maintenance of sorted linked lists, formally prove the correctness of those methods, and show how they can be extended to provide simple and efficient algorithms for concurrent maintenance of skip lists.

Conclusions The concurrent skip list algorithms described in this paper provide an efficient and practical method of allowing concurrent access and updates to a search structure in shared memory. Since skip lists are roughly as fast or faster than balanced trees in a non-concurrent environment and contention does not significantly slow down concurrent skip lists, I conjecture that the concurrent skip list algorithms described in this paper are at least as efficient as any possible concurrent balanced tree implementation. It might be possible to design concurrent balanced tree algorithms that allowed O(n) busy writers with high efficiency, but the complexity of such algorithms probably would make their implementation prohibitive.

Abstract ([main.dvi (psu.edu)](https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=95382d2d8adc76d19151d6ed80990f3879d9fc5a))

Abstract. We propose a new concurrent skip list algorithm distinguished by a combination of simplicity and scalability. The algorithm employs optimistic synchronization, searching without acquiring locks, followed by short lock-based validation before adding or removing nodes. It also logically removes an item before physically unlinking it. Unlike some other concurrent skip list algorithms, this algorithm preserves the skip list properties at all times, which facilitates reasoning about its correctness. Experimental evidence shows that this algorithm performs as well as the best previously known algorithm under most circumstances.

Conclusions We have shown how to construct a scalable, highly concurrent skip list using a remarkably simple algorithm. The principal open question is whether we can improve the algorithm’s performance at high levels of contention. One simple approach is to use a more sophisticated back-off scheme when synchronization conflicts are detected. Another intriguing approach is to allow the randomness of the skip-list’s height to be compromised under high contention. In the algorithm presented here, when a thread adds a new item, it gives up and retries if it encounters a synchronization conflict when linking the item at any level. In principle, a thread that encounters a conflict when linking the item at layer ` > 0 could simply stop there, leaving the item linked at levels zero to ` − 1, acting as if it had randomly chosen ` − 1. The resulting data structure, while structurally still a skip list, would be a little flatter than it should be. Whether this approach is effective is the subject of future work.

Bloom Filters:

Abstrct: [bloom.pdf (stanford.edu)](https://web.stanford.edu/~balaji/papers/bloom.pdf)

Bloom filters have been very interesting in networking because they enable the high speed, low cost implementation of various hardware algorithms. The paper introduces the idea of variable-length signatures, as opposed to the current practice of using fixed-length signatures. This idea naturally enables Bloom filters to perform flow deletions, a well-known problem with standard Bloom filters. Other uses of this idea are also presented and explored. A second contribution of the paper is the use of a bank of Bloom filters to identify the action that must be applied to the packets of a flow, or to dynamically record the state a flow is in. Our work shows that this approach is a promising alternative to expensive CAM or hash table lookups, and suggests a method of building cheap “fuzzy” flow memories

4 Conclusion The paper introduced two main ideas: variable-length signatures for Bloom filters (or VBFs), and the use of a bank of Bloom filters for building fuzzy flow memories. Due to a shortage of space, we were able to present only preliminary analyses and comparisons of these ideas; a more in-depth study will be presented in forthcoming publications. Our results are, nevertheless, quite encouraging and suggest that both the ideas can lead to better Bloom filters for tracking timevarying flow tables and for building flow memories. In addition, several research avenues have been mentioned as worth pursuing further and we are embarked on this program

Abstract: [Applied Sciences | Free Full-Text | Content-Based Approach for Improving Bloom Filter Efficiency (mdpi.com)](https://www.mdpi.com/2076-3417/13/13/7922)

Bloom filters are a type of data structure that is used to test whether or not an element is a member of a set. They are known for being space-efficient and are commonly employed in various applications, such as network routers, web browsers, and databases. These filters work by allowing a fixed probability of incorrectly identifying an element as being a member of the set, known as the false positive rate (FPR). However, traditional bloom filters suffer from a high FPR and extensive memory usage, which can lead to incorrect query results and a slow performance. Thus, this study indicates that a content-based strategy could be a practical solution for these challenges. Specifically, our approach requires less bloom filter storage, consequently decreasing the probability of false positives. The effectiveness of several hash functions on our strategy’s performance was also evaluated. Experimental evaluations demonstrated that the proposed strategy could potentially decrease false positives by a substantial margin of up to 79.83%. The use of size-based content bits significantly contributes to the decrease in the number of false positives as well. However, as the volume of content bits rises, the impact on time is not considerably noticeable. Moreover, the evidence suggests that the application of a singular approach leads to a more than 50% decrease in false positives.

6. Conclusions and Outlook

This research has presented a comprehensive examination of the challenges associated with traditional BFs and proposed a novel, content-based approach to addressing these issues. The utilization of a smaller amount of data storage space in the filter has been identified as the key factor in reducing the FPR, resulting in a significant improvement in the accuracy of membership testing. Additionally, the evaluation of different hash functions has played a crucial role in optimizing the performance of our approach. The experimental results obtained from the evaluation of our proposed approach demonstrate its effectiveness and superiority in comparison to traditional BFs in terms of the accuracy and efficiency of membership testing.

In summary, this research has successfully proposed a novel, content-based approach that addresses the challenges of traditional BFs and demonstrates its effectiveness through experimental evaluation. The proposed approach holds promise for practical applications in various domains and opens up new opportunities for future research in the field of BFs.

Cuckoo filter:

Abstract [cuckoo-conext2014.pdf (cmu.edu)](https://www.cs.cmu.edu/~dga/papers/cuckoo-conext2014.pdf)

In many networking systems, Bloom filters are used for highspeed set membership tests. They permit a small fraction of false positive answers with very good space efficiency. However, they do not permit deletion of items from the set, and previous attempts to extend “standard” Bloom filters to support deletion all degrade either space or performance. We propose a new data structure called the cuckoo filter that can replace Bloom filters for approximate set membership tests. Cuckoo filters support adding and removing items dynamically while achieving even higher performance than Bloom filters. For applications that store many items and target moderately low false positive rates, cuckoo filters have lower space overhead than space-optimized Bloom filters. Our experimental results also show that cuckoo filters outperform previous data structures that extend Bloom filters to support deletions substantially in both time and space.

A limitation of standard Bloom filters is that one cannot remove existing items without rebuilding the entire filter (or possibly introducing generally less desirable false negatives). Several approaches extend standard Bloom filters to support deletion, but with significant space or performance overhead. Counting Bloom filters [12] have been suggested for multiple applications [24, 25, 9], but they generally use 3–4× space to retain the same false positive rate as a space-optimized Bloom filter. Other variants include d-left counting Bloom filters [5], which are still 1.5× larger, and quotient filters [2], which provide significantly degraded lookup performance to yield comparable space overhead to Bloom filters. This paper shows that supporting deletion in approximate set membership tests need not impose higher overhead in space or performance compared to standard Bloom filters. We propose the cuckoo filter, a practical data structure that provides four major advantages. 1. It supports adding and removing items dynamically; 2. It provides higher lookup performance than traditional Bloom filters, even when close to full (e.g., 95% space utilized); 3. It is easier to implement than alternatives such as the quotient filter; and 4. It uses less space than Bloom filters in many practical applications, if the target false positive rate is less tha

Abstract: [Performance evaluation of Cuckoo filters as an enhancement tool for password cracking | Cybersecurity | Full Text (springeropen.com)](https://cybersecurity.springeropen.com/articles/10.1186/s42400-023-00193-6)

Abstract

Cyberthreats continue their expansion, becoming more and more complex and varied. However, credentials and passwords are still a critical point in security. Password cracking can be a powerful tool to fight against cyber criminals if used by cybersecurity professionals and red teams, for instance, to evaluate compliance with security policies or in forensic investigations. For particular systems, one crucial step in the password-cracking process is comparison or matchmaking between password-guess hashes and real hashes. We hypothesize that using newer data structures such as Cuckoo filters could optimize this process. Experimental results show that, with a proper configuration, this data structure is two orders of magnitude more efficient in terms of size/usage compared to other data structures while keeping a comparable performance in terms of time.

Conclusion

New data structures, such as the Cuckoo filters, have been proven efficient in several computer network applications. Nevertheless, its use in security has been limited mainly to authentication tasks. In this work, we have introduced a new use of Cuckoo filters as a valuable tool within the password-cracking process. The proposed method is particularly interesting for systems that use NTLM hashes because, in this scenario, the comparison step between generated hashes and target hashes requires a searching algorithm. Results show that whereas there is no a direct reduction in time, the gain in terms of memory usage is of two orders of magnitude compared to commonly employed data structures, which opens the door to further research in this direction.