Project Proposal

CSCI 6057 Advanced Data Structures

Balaji Sukumaran (B00948977) bl664064@dal.ca

Probabilistic Data Structures to Store Web Traffic Data

Description:

Probabilistic data structures [1], uses randomized algorithms to deliver approximate solutions with a certain probability of error, are essential when working with large datasets where exactness may be impractical. These structures excel in optimizing memory usage and boosting query performance, particularly when managing vast, sparse, and continuously expanding data streams. Accepting a certain error rate, such as false positives, becomes a calculated trade-off for the benefits of efficiency and scalability.

In this project I indend to investigate three probabilistic data structures: Concurrent Skip Lists, Bloom Filters, and Cuckoo Filters. The focus will be on their capability to store and manage clickstream data [2], with an emphasis on evaluating their efficiency through operations such as insertion, and existence checks.

The reason for selecting website traffic as the application domain is, website traffic data embodies the characteristics that probabilistic data structures handle well—it is vast, unending, and often sparse, making it an ideal candidate for a system that can accommodate imprecision. Secondly, the analysis of website traffic is imperative. By examining patterns in traffic sources, bounce rates, durations on site, and exit rates, we can enhance our understanding of user interactions, tailor marketing strategies, and refine website design to improve overall performance and conversion rates. Applying probabilistic data structures to the analysis of website traffic allows us to maintain the balance between the depth of insight garnered and the system's efficiency.

Summary of results:

Skip lists are data structures that provide an efficient and practical way to handle concurrent access and updates to a dynamic search structure [3][4]. Skip lists have been shown to operate as fast or faster than balanced trees in non-concurrent environments, and they handle contention without significant slowdown. They are a probabilistic alternative to balanced trees and are as efficient as any possible concurrent balanced tree implementation. A novel concurrent skip list algorithm has been developed that uses optimistic synchronization, allowing searches without locks and only using locks for validation before modifying the structure. This method, which preserves the skip list properties at all times, is simple and scalable, performing as well as the best previously known algorithms under most conditions. Future work may explore improvements to handle high levels of contention more effectively.

Bloom filters are employed in networking for high-speed, low-cost set membership tests, and their use enables various hardware algorithms. The innovation of variable-length signatures allows Bloom filters to perform flow deletions, solving a known problem with the standard implementation. Additionally, a bank of Bloom filters has been proposed for dynamically recording flow states or identifying actions for packets, presenting a cost-effective alternative to expensive hardware lookups. This paper has introduced the idea of variable-length

signatures and a bank of Bloom filters to improve the efficiency of tracking time-varying flow tables and building flow memories [5]. A content-based strategy has been identified as a practical solution to reduce the false positive rate and memory usage of traditional Bloom filters, with evaluations demonstrating a potential reduction in false positives by up to 79.83% [6].

Cuckoo filters [7], a new data structure, have been introduced to replace Bloom filters for approximate set membership tests. They support dynamic item addition and removal, achieving higher performance and space efficiency than Bloom filters, especially for applications with many items and moderate false positive rates. Cuckoo filters are more straightforward to implement than alternatives like quotient filters and use less space for practical applications when the false positive rate is moderate. They have been shown to outperform data structures extending Bloom filters to support deletions, both in time and space. In the context of cybersecurity, Cuckoo filters have been suggested as a valuable tool for password cracking processes. They provide significant memory usage improvements without direct reduction in time, enabling two orders of magnitude more efficiency in size/usage compared to other data structures [8].

Papers to read

1. A Separation Logic for Concurrent Randomized Programs [9]
2. The Splay-List: A Distribution-Adaptive Concurrent Skip-List [10]
3. NUMASK: High Performance Scalable Skip List for NUMA [11]
4. BloomFlash: Bloom Filter on Flash-Based Storage [12]
5. Cache-, hash-, and space-efficient bloom filters [13]
6. Cuckoo Filter: Simplification and Analysis [14]
7. Cuckoo Filters for Faster Triangle Listing and Set Intersection [15]
8. Adaptive Cuckoo Filters [16]

Plan:

1. Perform a detailed literature survey on the papers mentioned above.
2. Develop implementations of Concurrent Skip Lists, Bloom Filters, and Cuckoo Filters.
3. Search the internet for clickstream data.
4. Create a testing framework to evaluate the performance and efficiency of the data structures, particularly for insertion and existence checks.
5. Generate graphs to demonstrate the efficiency and performance differences among the probabilistic data structures.

References

[1] Anik, “Probabilistic Data Structures Simplified”, *Medium*, Available: https://medium.com/@tech.anikghosh/probabilistic-data-structures-db5d238008eb, [Accessed: Mar 11, 2024]

[2] T. Lai, “Web Traffic Time Series Forecasting”, *Kaggle*, Available: https://www.kaggle.com/datasets/ymlai87416/wiktraffictimeseriesforecast/data, [Accessed: Mar 11, 2024]

[3] W. Pugh, “Concurrent maintenance of skip lists,” in University of Maryland at College Park, USA, 1990. Available: https://dl.acm.org/doi/10.5555/93717

[4] Herlihy, Maurice & Lev, Yossi & Luchangco, Victor & Shavit, Nir, “A Provably Correct Scalable Concurrent Skip List,” 2010, [Online]. Available: https://www.researchgate.net/publication/249902280_A_Provably_Correct_Scalable_Concurrent_Skip_List

[5] Lu, Yi, B. Prabhakar and F. Bonomi, “Bloom filters: Design innovations and novel applications,” 2005, [Online]. Available: https://web.stanford.edu/papers/bloom.pdf

[6] Alsuhaibani M, Khan RU, Qamar AM, Alsuhibany SA. Content-Based Approach for Improving Bloom Filter Efficiency. Applied Sciences. 2023; 13(13):7922. https://doi.org/10.3390/app13137922

[7] Bin Fan, Dave G. Andersen, Michael Kaminsky, and Michael D. Mitzenmacher. 2014. Cuckoo Filter: Practically Better Than Bloom. In Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies (CoNEXT '14). Association for Computing Machinery, New York, NY, USA, 75–88. https://doi.org/10.1145/2674005.2674994

[8] Cano, MD., Villafranca, A. & Tasic, I. Performance evaluation of Cuckoo filters as an enhancement tool for password cracking. Cybersecurity 6, 57 (2023). https://doi.org/10.1186/s42400-023-00193-6

[9] Joseph Tassarotti and Robert Harper. 2019. A Separation Logic for Concurrent Randomized Programs. Proc. ACM Program. Lang. 3, POPL, Article 64 (January 2019), 30 pages. https://doi.org/10.1145/3290377

[10] Vitaly Aksenov, Dan Alistarh, Alexandra Drozdova, and Amirkeivan Mohtashami. 2023. The splay-list: a distribution-adaptive concurrent skip-list. Distrib. Comput. 36, 3 (Sep 2023), 395–418. https://doi.org/10.1007/s00446-022-00441-x

[11] Henry Daly, Ahmed Hassan, Michael F. Spear, and Roberto Palmieri. NUMASK: High Performance Scalable Skip List for NUMA. In 32nd International Symposium on Distributed Computing (DISC 2018). Leibniz International Proceedings in Informatics (LIPIcs), Volume 121, pp. 18:1-18:19, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2018) URL: https://doi.org/10.4230/LIPIcs.DISC.2018.18

[12] B. Debnath, S. Sengupta, J. Li, D. J. Lilja and D. H. C. Du, "BloomFlash: Bloom Filter on Flash-Based Storage," 2011 31st International Conference on Distributed Computing Systems, Minneapolis, MN, USA, 2011, pp. 635-644, doi: 10.1109/ICDCS.2011.44.

[13] Felix Putze, Peter Sanders, and Johannes Singler. 2010. Cache-, hash-, and space-efficient bloom filters. ACM J. Exp. Algorithmics 14, Article 4 (2009), 18 pages. https://doi.org/10.1145/1498698.1594230

[14] David Eppstein. Cuckoo Filter: Simplification and Analysis. In 15th Scandinavian Symposium and Workshops on Algorithm Theory (SWAT 2016). Leibniz International Proceedings in Informatics, Volume 53, pp. 8:1-8:12, Schloss Dagstuhl – Leibniz-Zentrum für Informatik (2016) URL: https://doi.org/10.4230/LIPIcs.SWAT.2016.8

[15] David Eppstein, Michael T. Goodrich, Michael Mitzenmacher, and Manuel R. Torres. 2017. 2-3 Cuckoo Filters for Faster Triangle Listing and Set Intersection. In Proceedings of the 36th ACM SIGMOD-SIGACT-SIGAI Symposium on Principles of Database Systems (PODS '17). Association for Computing Machinery, New York, NY, USA, 247–260. https://doi.org/10.1145/3034786.3056115

[16] Michael Mitzenmacher, Salvatore Pontarelli, and Pedro Reviriego. 2020. Adaptive Cuckoo Filters. ACM J. Exp. Algorithmics 25, Article 1.1 (2020), 20 pages. https://doi.org/10.1145/3339504