H-Sketch: Top-k Traffic Inspection with High-precision and High-throughput in High-speed Networks

Jin Wang [1,2], Menghuan Du [1] and Jinbin Hu [1]

[1] Changsha University of Science and Technology, Changsha 410114, China
[2] Hunan University of Science and Technology, Xiangtan 411201, China
jinwang@hnust.edu.cn,
dmh@stu.csust.edu.cn, jinbinhu@csust.edu.cn

Abstract. Efficient real-time top-K traffic inspection is critical in the field of network measurement, plays a significant role in enhancing network performance and security. With the network traffic surges, especially in high-speed network environments, real-time monitoring and analysis of traffic data has become more and more complex. The existing methods of detecting top-k are often difficult to balance accuracy and memory overhead, and the data processing efficiency in high-speed network environments is difficult to reach the ideal level. Motivated by this, we propose H-Sketch, a new method with high precision and high throughput. H-Sketch can reduce the frequent replacement of flows due to hash conflicts, and record only the complete ID of large flows. The experimental results show that H-Sketch effectively improves processing efficiency by up to 40% and reduces memory consumption while maintaining accuracy.

Keywords: Sketch, Top-k, Network Measurement.

1 Introduction

Top-K detection in high-speed networks is a core task of network measurement [1-3], and has a wide range of application prospects in network management, traffic engineering, anomaly detection, and security monitoring. Top-K detection refers to the accurate identification of the top-K data flows that occur most frequently in a large volume of network traffic, which is essential for efficient congestion control, load balancing, and DDoS attack detection [4-6]. However, with the rapid growth of network traffic, real-time Top-K detection faces the dilemma of balancing accuracy and memory overhead, and existing methods often struggle to ensure detection accuracy while processing traffic efficiently.

Counter-based methods such as Space-Saving [7] and Lossy Counting [8], which track the size of the stream by maintaining counters, tend to overestimate errors when memory is limited, reducing accuracy, although the update speed is fast. Sketch-based methods such as Cuckoo Counter [9] and Waving Sketch[10] use probabilistic data

[1]Jinbin Hu is the corresponding author.

structures and hashing techniques to alleviate memory pressure, but under high traffic loads, frequent hash collisions may lead to frequent stream replacements, which seriously affect the estimation accuracy and processing efficiency. In addition, high-speed network links have strict limits on the number of memory accesses[11], and existing methods such as HeavyKeeper[12] and Elastic Sketch[13] are difficult to meet throughput requirements due to excessive memory accesses. Overall, existing methods struggle to balance throughput, accuracy, and memory efficiency.

To address the above issues, we propose a new detection method, H-Sketch, which significantly reduces memory overhead by recording only the complete ID of the large stream and compressing the stream IDs of the small streams. A dynamic positioning mechanism is utilized to store large streams in large entries of memory allocation, thereby reducing unnecessary memory access and improving processing efficiency. Only the first entry of the bucket and the conflicting items need to be accessed, avoiding the inefficiencies caused by multiple traversals in traditional methods. This design not only ensures high throughput but also significantly enhances memory efficiency and detection accuracy, effectively tackling the challenges of top-k detection in high-speed network environments.

The rest of this document is organized as follows: In Section 2, we describe the design motivations. In Section 3, we detail the details of our design. In section 4, we conduct experiments and analyze the results. In Section 5, we summarize the paper.

2 Motivation

The existing top-k detection methods can be divided into heap-based tracking methods and bucket array-based Sketch methods. The minimum heap-based approach uses a small heap with k slots to track the largest k flows, traversing k heap entries each time a packet arrives, resulting in limited throughput. Bucket array-based approaches, such as Space-Saving and Waving Sketch, map streams to specific buckets in an array and record the stream's ID and size in the buckets, but due to the limited bucket capacity, stream entries are frequently replaced under high loads, resulting in reduced accuracy. Cuckoo Counter combines the characteristics of bucket array and min-heap, records flow information through bucket array, and uses the minimum heap to assist in filtering top-k traffic, but frequent structural adjustment will lead to more memory access and computational overhead, reducing throughput.

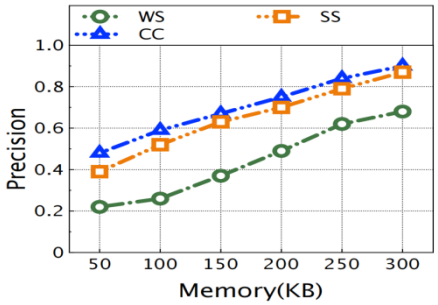

Fig. 1 Precision

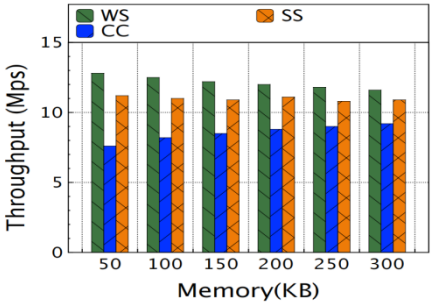

Fig. 2 Throughput

In order to verify the performance of existing methods in high-speed network environments, we conducted experiments based on the CAIDA real IP dataset. The experiments simulated high-speed network traffic scenarios to test the performance of existing methods when facing large-scale traffic. In the experiments, we set different memory sizes ranging from 50KB to 300KB to observe the impact of memory overhead on throughput and detection accuracy. The experiments compared three mainstream methods: Waving Sketch (WS), Cuckoo Counter (CC), and Space-Saving (SS), aiming to evaluate their overall performance in high-speed network environments.

The experimental results are shown in Figure 1 and Figure 2, and it is difficult for existing methods to achieve high throughput, high accuracy and low memory overhead at the same time in a high-speed network environment. WS has high throughput but needs to be improved in terms of measurement accuracy, CC has high measurement accuracy but limited throughput, and SS has balanced throughput and measurement accuracy but poor performance at low memory. These limitations indicate the shortcomings of existing methods in high-speed network scenarios, and a new solution needs to be proposed.

3 Design

3.1 Design Overview

The core idea of H-Sketch is to achieve collaborative optimization of memory efficiency, detection accuracy, and throughput in high-speed network environments through hierarchical storage policies and dynamic conflict management mechanisms. In view of the long-tail distribution characteristics of network traffic (i.e., the vast majority of them are low-frequency mouse streams, and a few are elephant streams), H-Sketch adopts differentiated memory allocation: only the space for recording the full ID is allocated for large streams, while the stream ID is compressed for small streams, and only part of the ID (fingerprint) is stored. At the same time, the global threshold update and conflict flow recording mechanism are used to reduce the accuracy loss caused by hash conflicts, so as to achieve efficient and accurate top-K flow detection under limited memory conditions.

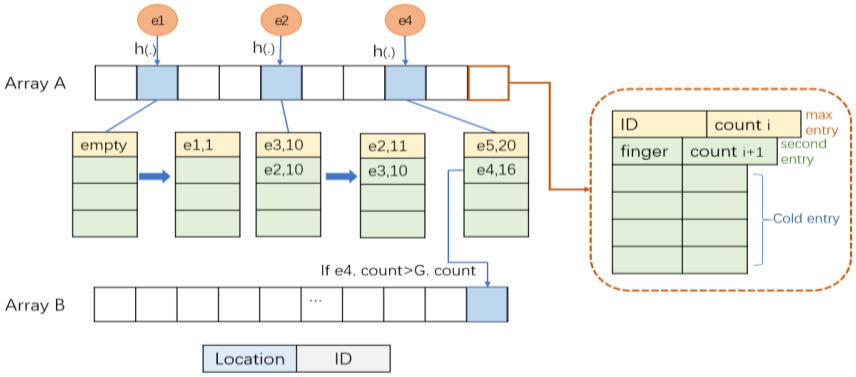

Fig. 3 Structure of the H-Sketch

H-Sketch consists of two parts, as shown in Figure 3, the first layer of array A contains w buckets, each bucket is used as a basic access unit, containing multiple entries, and the entries are the basic storage units. Entries are divided into full ID storage (only the first entry) and fingerprint storage (other entries), and the space utilization is optimized by combining the dynamic memory allocation strategy. Array B in the second layer consists of w-1 buckets, which record the complete ID and location information of the conflicting stream in array A, which is used to correct the detection error caused by hash conflicts. The global variable G is set to track the frequency of the Kth stream in array A in real time, and the collision range is dynamically adjusted to ensure the stable tracking of the high-frequency stream.

3.2 Design Detail

The first entry of array A stores the full stream ID and allocates a larger counter, the second entry will be recorded to Array B if it has a conflicting full ID, so only part of the ID is stored but also a larger counter is allocated, and the rest of the entries only store the partial ID (fingerprint) and compact counter of the stream. Through the dynamic positioning mechanism, it is ensured that the large volume of data is stored in entries with larger memory allocations. When the sub-large stream in the bucket of array A exceeds the threshold of the global variable G, we consider that a conflict has occurred and record the corresponding flow information to array B to solve the problem of high-frequency stream coverage caused by hash conflict.

Insertion. When the packet p belonging to stream e arrives, we first find the candidate bucket of array A through the hash calculation index, iterate through all the entries in the candidate bucket, and check whether stream e has been recorded. There may be the following three situations：

Case1. If stream e is not currently logged but there are idle entries in the bucket, store the ID or fingerprint of stream e in the free entries and change the counter to 1.

Case2. If stream e already exists in the bucket, perform different operations according to its order in the bucket, if stream e is the current largest stream in the bucket, increase its counter by 1; If stream e is the second largest stream in the bucket, after increasing its counter by 1, compare it with the threshold recorded by the largest stream and the global variable G in the bucket, if the counter of stream e exceeds the maximum stream, it will swap positions with the first entry and become the new largest stream, and if the counter of stream e exceeds G, its full ID and location information will be recorded in array B to avoid conflict override. If stream e is not the first two streams, increase the counter by 1 and compare it to the next largest stream to trigger potential contention.

Case3. If stream e is not recorded and there are no free entries in the bucket, replace the stream with the smallest counter as stream e, and initialize the counter to 1.

Query. When you need to query the current Top-K, traverse the first entry of each bucket in array A and retrieve the conflicting stream information recorded in array B at the same time, and finally select the first K streams as the final result.

4 Evaluation

4.1 Experimental Setup

Datasets. We used the CAIDA 2019 dataset to simulate 100Gbps+ high-speed network traffic, covering traffic processing scenarios at different memory scales, and simulated the characteristics of long-tail distribution in real traffic to verify the performance of the three methods when processing large-scale data.

Platform. We conducted experiments on a server with a quad-core CPU (Intel (R) Core (TM) i5-8265U CPU) that supports both 32-bit and 64-bit operating modes. Each core has three levels of cache: a 128KB L1d cache, a 128KB L1i cache, a 1MB L2 cache, and a 12MB L3 cache. The operating system used is Ubuntu 20.04.

Evaluation Metrics. We record the proportion of accurately identified top-k items out of the total reported items as detection precision, measure the accuracy of traffic size estimation with precision, use throughput to evaluate processing speed, and measure their memory overhead by observing performance at different memory sizes.

4.2 Experimental Results

Throughput. We calculated the throughput of Waving Sketch (WS), Cuckoo Counter (CC), Space-Saving (SS), and H-Sketch (HS) at different memory sizes. As can be seen in Figure 4, HS consistently has the highest throughput and performs better when memory is small, while WS also performs relatively well in terms of throughput, but not as good as HS. Overall, throughput performance is sorted as HS>WS > SS > CC.

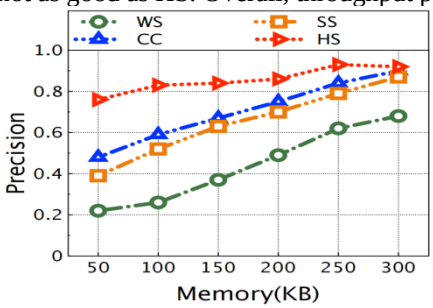

Fig. 4 Precision

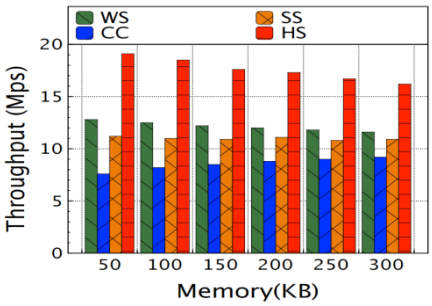

Fig. 5 Throughput

Precision. We evaluated the accuracy of WS, CC, SS, and HS at different memory sizes. As can be seen in Figure 5, HS consistently has a high accuracy rate, while CC is as accurate as HS but poor in terms of throughput. As the memory increases, the accuracy of SS also skyrockets, but it is not as accurate as CC and HS. In general, the order in terms of precision is HS > CC > SS >WS.

5 Conclusion

This paper presented H-Sketch, an efficient real-time top-k traffic inspection method optimized for high-traffic scenarios in a high-speed network environment. The core of

H-Sketch contains: (1) an efficient traffic detection method based on hierarchical storage strategy and dynamic conflict management mechanism; (2) Differentiated memory distribution strategy for stream long-tail distribution. The simulation results show that compared with the existing research, H-Sketch effectively improves the throughput of the real high-speed network by 40%.

Acknowledgement

This work was supported in part by the National Natural Science Foundation of China under Grant 62473146 and Grant 62472050; in part by the Natural Science Foundation of Hunan Province under Grant 2024JJ3017 and Grant 2025JJ20070; and supported by the Science and Technology Project of Hunan Provincial Department of Water Resources under Grant XSKJ2024064-36.

References

1. L. Gu, Y. Tian, W. Chen, Z. Wei, C. Wang, X. Zhang. Per-flow network measurement with distributed sketch. In Proc. IEEE/ACM Transactions on Networking 32(1), 411-426, 2023.
2. J. Hu, Y. He, W. Luo, J. Huang, J. Wang. Enhancing load balancing with in-network recirculation to prevent packet reordering in lossless data centers. In Proc. IEEE/ACM Transactions on Networking 32(5), 4114 – 4127, 2024.
3. B. Zhao, X. Li, B. Tian, Z. Mei, W. Wu. DHS: Adaptive memory layout organization of sketch slots for fast and accurate data stream processing. In Proc. ACM SIGKDD, 2021.
4. X. Chen, Q. Xiao, H. Liu, Q. Huang. Eagle: Toward Scalable and Near-Optimal Network-Wide Sketch Deployment in Network Measurement. In Proc. ACM SIGCOMM, 2024.
5. Y. Du, H. Huang, Y. E. Sun, S. Chen, G. Gao. Self-adaptive sampling for network traffic measurement. In Proc. IEEE INFOCOM, 2021.
6. Y. E. Sun, H. Huang, C. Ma, S. Chen, Y. Du, Q. Xiao. Online spread estimation with non-duplicate sampling. In Proc. IEEE INFOCOM, 2020.
7. A. Metwally, D. Agrawal, A. El Abbadi. Efficient computation of frequent and top-k elements in data streams. In Proc. Springer ICDT, 2005.
8. G. S. Manku, R. Motwani. Approximate frequency counts over data streams. In Proc. Morgan Kaufmann/ACM VLDB, 2002.
9. Q. Shi, Y. Xu, J. Qi, W. Li, T. Yang, Y. Xu. Cuckoo Counter: Adaptive structure of counters for accurate frequency and top-k estimation. In Proc. IEEE/ACM Transactions on Networking 31(4), 1854-1869, 2023.
10. J. Li, Z. Li, Y. Xu, S. Jiang, T. Yang, et al. WavingSketch: An unbiased and generic sketch for finding top-k items in data streams. In Proc. ACM SIGKDD, 2020.
11. J. Hu, S. Rao, M. Zhu, J. Huang, J. Wang, J. Wang. SRCC: Sub-RTT Congestion Control for Lossless Datacenter Networks. In Proc. IEEE Transactions on Industrial Informatics, 2024, DOI: 10.1109/TII.2024.3495759.
12. T. Yang, H. Zhang, J. Li, J. Gong, S. Uhlig, S. Chen. HeavyKeeper: an accurate algorithm for finding Top-k elephant flows. In Proc. IEEE/ACM Transactions on Networking 27(5), 1845-1858, 2019.
13. T. Yang, J. Jiang, P. Liu, et al. Elastic sketch: Adaptive and fast network-wide measurements. In Proc. ACM SIGCOMM, 2018.