Resilient Disaggregated Network Flow Monitoring

Yongquan Fu
Science and Technology Laboratory of Parallel and Distributed Processing, College of Computer Science, National University of Defense Technology

Dongsheng Li
Science and Technology Laboratory of Parallel and Distributed Processing, College of Computer Science, National University of Defense Technology

Siqi Shen
Science and Technology Laboratory of Parallel and Distributed Processing, College of Computer Science, National University of Defense Technology

Yiming Zhang
Science and Technology Laboratory of Parallel and Distributed Processing, College of Computer Science, National University of Defense Technology

Kai Chen
SING Lab, Hong Kong University of Science and Technology

ABSTRACT

Sketch has been extensively used for scalable network monitoring, which unfortunately, is sensitive to hash collisions. Deploying the sketch involves fine-grained performance control and instrumentation. This paper presents a new class of sketch structure that proactively minimizes the estimation error and reduces the error variance. We develop a disaggregated monitoring application that natively scales the sketching deployment. Testbed and real-world trace-driven simulations show that LSS achieves close-to-optimal performance under hash collisions.

CCS CONCEPTS

• **Networks** → **Network monitoring**; *Network services*.

KEYWORDS

Network flow, sketch, hash function, disaggregated application

ACM Reference Format:
Yongquan Fu, Dongsheng Li, Siqi Shen, Yiming Zhang, and Kai Chen. 2019. Resilient Disaggregated Network Flow Monitoring. In *SIGCOMM '19: ACM SIGCOMM 2019 Conference (SIGCOMM Posters and Demos '19), August 19–23, 2019, Beijing, China.* ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3342280.3342305

1 INTRODUCTION

Network flow monitoring needs a variety of traffic measurement, such as flow delay, flow queueing time, flow frequency, flow entropy, and heavy hitters [2, 5, 10, 15, 21, 28–30, 33, 34]. Traffic statistics tasks require advanced data structures and traffic statistical algorithms due to ever-increasing line rates, massive traffic volumes, and large numbers of active flows [16, 25, 26, 35]. Many space- and time-efficient approaches have been studied, e.g., traffic sampling, traffic counting, traffic sketching [3, 8, 11, 13, 14, 18–20, 24]. Compared to other approaches, the sketch has received extensive attentions due to their competitive trade-off between resource consumption and performance guarantees [1, 22, 32], which is widely used in traffic engineering, network diagnosis, network forensics, intrusion detection and prevention. The sketch internally builds a memory-efficient and constant-speed bucket array to accumulate incoming flow counters [4, 6, 7, 23, 31]. A fundamental question is that, the sketch is sensitive to hash collisions, where multiple keys are mapped to the same bucket, as this noisy bucket no longer returns exact results for any of inserted keys. Recently, ElasticSketch [31] and SketchLearn [17] separates large flows from the sketch structure but need to allocate dedicated space for new items.

We present a new class of sketch called locality-sensitive sketch LSS that is resilient to hash collisions with a constant-size data structure. Real-world deployment experiments and trace-driven simulation confirms that LSS dramatically reduces the estimation error under the same memory footprint. More details are provided in the technical report [12].

2 NETWORK FLOW MONITORING

LSS turns from passively tolerating noisy buckets to proactively recovering the noisy buckets. A bucket array consists of a number of buckets, where each bucket has two fields: (i) A ValSum field that records the sum of values; (ii) A KeyCount filed that records the number of unique keys inserted to this bucket. LSS maps each item to only one bucket array that corresponds to the nearest cluster center for this item. Each bucket array corresponds to a cluster of similar items. LSS averages the bucket's counter to produce an unbiased estimator for noisy buckets. Figure 1 shows the main processes in LSS.

Clustering: We represent the clusters with k cluster centers, thus an LSS is organized as a number k of bucket arrays. The distance from the item to a cluster center is defined as the one-dimensional absolute difference between the item's value and the cluster center. Thus the overall time to find the nearest cluster centers is $O(k)$.

LSS learns the cluster structure based on transferred learning from network flow samples. We initialize the clustering model with

SIGCOMM Posters and Demos '19, August 19–23, 2019, Beijing, China

ACM ISBN 978-1-4503-6886-5/19/08…$15.00
https://doi.org/10.1145/3342280.3342305

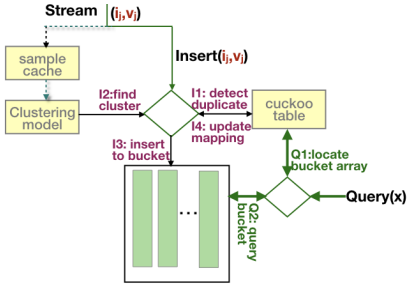
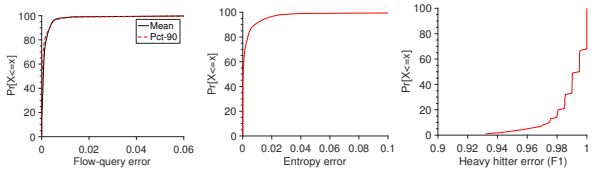


Figure 1: Illustration of LSS' main processes.



(a) Flow-size query (b) Flow entropy (c) Heavy hitters

Figure 2: Testbed performance metrics.

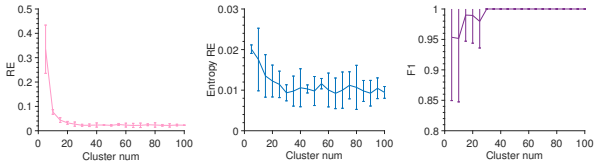
a set of sampled items from the stream. The clustering model is then fed to the LSS instance to produce the clusters. Next, we periodically refresh the clustering model with up-to-date sample cache to adapt to the variations of stream distributions. We select the well-studied K-means clustering method that clusters samples to minimizes the intra-cluster variance.

Insertion: To insert a key-value pair, LSS maps the value to the nearest cluster center, and accumulates the value to the corresponding bucket array. We increment the KeyCount field only once for each unique key to correctly count the number of keys. Further, we adapt to packets from the same network flow with a cuckoo table [9, 27, 36] that dynamically keeps the counters with the same key.

Query: To query the value of a key on the LSS, we first locate the bucket array with a Cuckoo hash table with the input key to get the cluster index of this key. Finally, we return the weighted value $\frac{ValSum}{KeyCount}$ as the approximated result.

Disaggregated Monitoring Framework: The proposed monitoring architecture splits the monitoring application into non-coherent ingestion, sketching, query runtime functions that can be horizontally scaled in the data center. A publish/subscribe (Pub/Sub for short) framework delivers ordered streaming messages across monitoring functions. We choose the Pulsar messaging system originally created at Yahoo as the Pub/Sub underlay.

Evaluation: We ran experiments on ten servers in two racks connected by a 10 Gbps switch. We choose three representative monitoring tasks to evaluate the sketch's performance, namely the flow-size query, the flow-entropy query, and the heavy-hitter query. We quantify the performance of the first two tasks with the relative error metric: defined as $|x_r - x_e| / (x_r)$, where x_r and x_e denoted the ground-truth metric and the estimated metric, respectively, and the last task based on the F1 score defined as the harmonic mean of the precision and the recall values.



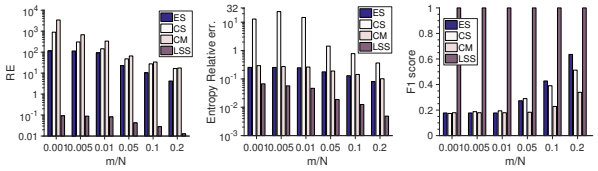
(a) Flow query (b) Entropy (c) Heavy hitters

Figure 3: LSS performance as a function of cluster numbers.

Figure 2(a) plots the Cumulative Distribution Function (CDF) of the mean relative errors. The relative errors of over 90% of all estimations are smaller than 0.01. Over 90% of estimations are smaller than 0.06, because of accurate estimations of flow sizes. Over 90% of tests are greater than 0.95. As LSS captures fine-grained flow distributions with clustered bucket arrays.

Next, we performed a trace-driven study that consists of 8.4 million flows collected on February 18, 2016 at the Equinix-Chicago monitor by CAIDA [31]. We evaluate LSS' accuracy with respect to the number of clusters. Figure 3 plots the variation of the estimation accuracy. We see that the estimation accuracy improves steadily with increasing numbers of clusters from two to ten. The diminishing returns occur when the number of cluster reaches 30.

Further, we compare LSS with count-min (CM) [7], count-sketch (CS) [4], and Elastic Sketch (ES) [31] that are most related with our work. We set the same memory footprint for all compared sketch structures. Figure 4 plots the performance of the flow-size, flow-entropy, and heavy-hitter query tasks, as we vary the ratio between the number of buckets in LSS and the number of unique flows. We see that LSS significantly outperforms other sketch structures in all cases.



(a) Flow-size query (b) Flow entropy (c) Heavy hitters

Figure 4: Simulation performance metrics.

3 CONCLUSION

We have proposed a new class of sketch that is resilient to hash collisions. We present a disaggregated monitoring application that allows for non-intrusive sketch deployment and native network-wide analytics. Extensive evaluation shows that LSS achieves close to optimal performance with a tiny memory footprint.

ACKNOWLEDGMENTS

This work was sponsored in part by National Key R&D Program of China under Grant No. 2018YFB0204300, and the National Natural Science Foundation of China (NSFC) under Grant No. 61602500, 61402509, 61772541, 61872376.

REFERENCES

[1] Omid Alipourfard, Masoud Moshref, Yang Zhou, Tong Yang, and Minlan Yu. 2018. A Comparison of Performance and Accuracy of Measurement Algorithms in Software. In *Proceedings of the Symposium on SDN Research, SOSR 2018, Los Angeles, CA, USA, March 28-29, 2018*. 18:1–18:14.

[2] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, and George Varghese. 2014. CONGA: distributed congestion-aware load balancing for datacenters. In *ACM SIGCOMM 2014 Conference, SIGCOMM'14, Chicago, IL, USA, August 17-22, 2014*. 503–514.

[3] Ran Ben-Basat, Gil Einziger, Roy Friedman, Marcelo Caggiani Luizelli, and Erez Waisbard. 2017. Constant Time Updates in Hierarchical Heavy Hitters. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017*. 127–140.

[4] Moses Charikar, Kevin C. Chen, and Martin Farach-Colton. 2002. Finding Frequent Items in Data Streams. In *Proc. of ICALP*. 693–703.

[5] Li Chen, Justinas Lingys, Kai Chen, and Feng Liu. 2018. AuTO: scaling deep reinforcement learning for datacenter-scale automatic traffic optimization. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2018, Budapest, Hungary, August 20-25, 2018*. 191–205.

[6] Graham Cormode. 2017. Data sketching. *Commun. ACM* 60, 9 (2017), 48–55.

[7] Graham Cormode and S. Muthukrishnan. 2004. An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. In *Proc. of LATIN*. 29–38.

[8] Cristian Estan and George Varghese. 2003. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Trans. Comput. Syst.* 21, 3 (2003), 270–313.

[9] Bin Fan, David G. Andersen, Michael Kaminsky, and Michael Mitzenmacher. 2014. Cuckoo Filter: Practically Better Than Bloom. In *Proceedings of the 10th ACM International on Conference on emerging Networking Experiments and Technologies, CoNEXT 2014, Sydney, Australia, December 2-5, 2014*. 75–88.

[10] Nick Feamster and Jennifer Rexford. 2017. Why (and How) Networks Should Run Themselves. *CoRR* abs/1710.11583 (2017). arXiv:1710.11583 http://arxiv.org/abs/1710.11583

[11] Yongquan Fu, Pere Barlet-Ros, and Dongsheng Li. 2018. Every Timestamp Counts: Accurate Tracking of Network Latencies Using Reconcilable Difference Aggregator. *IEEE/ACM Trans. Netw.* 26, 1 (2018), 90–103.

[12] Yongquan Fu, Dongsheng Li, Siqi Shen, Yiming Zhang, and Kai Chen. 2019. Locality-Sensitive Sketching for Resilient Network Flow Monitoring. *CoRR* (2019). https://arxiv.org/abs/1905.03113

[13] Yongquan Fu, Yijie Wang, and Ernst Biersack. 2013. A General Scalable and Accurate Decentralized Level Monitoring Method for Large-scale Dynamic Service Provision in Hybrid Clouds. *Future Generation Comp. Syst.* 29, 5 (2013), 1235–1253.

[14] Yongquan Fu and Xiaoping Xu. 2017. Self-Stabilized Distributed Network Distance Prediction. *IEEE/ACM Trans, Netw.* 25, 1 (Feb 2017), 451–464.

[15] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, David A. Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, Zhi-Wei Lin, and Varugis Kurien. 2015. Pingmesh: A Large-Scale System for Data Center Network Latency Measurement and Analysis. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17-21, 2015*. 139–152.

[16] Arpit Gupta, Rob Harrison, Marco Canini, Nick Feamster, Jennifer Rexford, and Walter Willinger. 2018. Sonata: query-driven streaming network telemetry. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2018, Budapest, Hungary, August 20-25, 2018*. 357–371.

[17] Qun Huang, Patrick P. C. Lee, and Yungang Bao. 2018. Sketchlearn: Relieving User Burdens in Approximate Measurement with Automated Statistical Inference. In *Proc. of SIGCOMM*. 576–590.

[18] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. 2003. Sketch-based change detection: methods, evaluation, and applications. In *Proceedings of the 3rd ACM SIGCOMM Internet Measurement Conference, IMC 2003, Miami Beach, FL, USA, October 27-29, 2003*. 234–247.

[19] Dongsheng Li, Jiannong Cao, Xicheng Lu, and Kaixian Chen. 2009. Efficient Range Query Processing in Peer-to-Peer Systems. *IEEE Trans. Knowl. Data Eng.* 21, 1 (2009), 78–91.

[20] Xin Li, Fang Bian, Mark Crovella, Christophe Diot, Ramesh Govindan, Gianluca Iannaccone, and Anukool Lakhina. 2006. Detection and identification of network anomalies using sketch subspaces. In *Proceedings of the 6th ACM SIGCOMM Internet Measurement Conference, IMC 2006, Rio de Janeriro, Brazil, October 25-27, 2006*. 147–152.

[21] Ziyang Li, Wei Bai, Kai Chen, Dongsu Han, Yiming Zhang, Dongsheng Li, and Hongfang Yu. 2017. Rate-aware flow scheduling for commodity data center networks. In *2017 IEEE Conference on Computer Communications, INFOCOM 2017, Atlanta, GA, USA, May 1-4, 2017*. 1–9.

[22] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. 2016. One Sketch to Rule Them All: Rethinking Network Flow Monitoring with UnivMon. In *Proceedings of the ACM SIGCOMM 2016 Conference, Florianopolis, Brazil, August 22-26, 2016*. 101–114.

[23] Gurmeet Singh Manku and Rajeev Motwani. 2002. Approximate Frequency Counts over Data Streams. In *VLDB 2002, Proceedings of 28th International Conference on Very Large Data Bases, August 20-23, 2002, Hong Kong, China*. 346–357.

[24] Masoud Moshref, Minlan Yu, Ramesh Govindan, and Amin Vahdat. 2015. SCREAM: sketch resource allocation for software-defined measurement. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies, CoNEXT 2015, Heidelberg, Germany, December 1-4, 2015*. 14:1–14:13.

[25] Srinivas Narayana, Anirudh Sivaraman, Vikram Nathan, Prateesh Goyal, Venkat Arun, Mohammad Alizadeh, Vimalkumar Jeyakumar, and Changhoon Kim. 2017. Language-Directed Hardware Design for Network Performance Monitoring. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017*. 85–98.

[26] Srinivas Narayana, Mina Tahmasbi, Jennifer Rexford, and David Walker. 2016. Compiling Path Queries. In *13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016*. 207–222.

[27] Rasmus Pagh and Flemming Friche Rodler. 2004. Cuckoo hashing. *J. Algorithms* 51, 2 (2004), 122–144.

[28] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. 2015. Inside the Social Network's (Datacenter) Network. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17-21, 2015*. 123–137.

[29] Arjun Roy, Hongyi Zeng, Jasmeet Bagga, and Alex C. Snoeren. 2017. Passive Realtime Datacenter Fault Detection and Localization. In *14th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2017, Boston, MA, USA, March 27-29, 2017*. 595–612.

[30] Haiyang Wang, Tong Li, Ryan Shea, Xiaoqiang Ma, Feng Wang, Jiangchuan Liu, and Ke Xu. 2018. Toward Cloud-Based Distributed Interactive Applications: Measurement, Modeling, and Analysis. *IEEE/ACM Trans. Netw.* 26, 1 (2018), 3–16.

[31] Tong Yang, Jie Jiang, Peng Liu, Qun Huang, Junzhi Gong, Yang Zhou, Rui Miao, Xiaoming Li, and Steve Uhlig. 2018. Elastic Sketch: Adaptive and Fast Network-wide Measurements. In *Proc. of SIGCOMM*. 561–575.

[32] MyungKeun Yoon, Tao Li, Shigang Chen, and Jih-Kwon Peir. 2011. Fit a Compact Spread Estimator in Small High-Speed Memory. *IEEE/ACM Trans. Netw.* 19, 5 (2011), 1253–1264.

[33] Hong Zhang, Li Chen, Bairen Yi, Kai Chen, Mosharaf Chowdhury, and Yanhui Geng. 2016. CODA: Toward Automatically Identifying and Scheduling Coflows in the Dark. In *Proceedings of the ACM SIGCOMM 2016 Conference, Florianopolis, Brazil, August 22-26, 2016*. 160–173.

[34] Hong Zhang, Junxue Zhang, Wei Bai, Kai Chen, and Mosharaf Chowdhury. 2017. Resilient Datacenter Load Balancing in the Wild. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication, SIGCOMM 2017, Los Angeles, CA, USA, August 21-25, 2017*. 253–266.

[35] Yibo Zhu, Nanxi Kang, Jiaxin Cao, Albert G. Greenberg, Guohan Lu, Ratul Mahajan, David A. Maltz, Lihua Yuan, Ming Zhang, Ben Y. Zhao, and Haitao Zheng. 2015. Packet-Level Telemetry in Large Datacenter Networks. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication, SIGCOMM 2015, London, United Kingdom, August 17-21, 2015*. 479–491.

[36] Pengfei Zuo, Yu Hua, and Jie Wu. 2018. Write-Optimized and High-Performance Hashing Index Scheme for Persistent Memory. In *13th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2018, Carlsbad, CA, USA, October 8-10, 2018*. 461–476.