Distributed Sketching with Traffic-Aware Summaries

Dor Harris[§]
Technion
Haifa, Israel
dorharris@cs.technion.ac.il

Arik Rinberg[§]
Technion
Haifa, Israel
arikrinberg@campus.technion.ac.il

Ori Rottenstreich
Technion
Haifa, Israel
or@technion.ac.il

***Abstract*—Network measurements are important for identifying congestion, DDoS attacks, and more. To support real-time analytics, stream ingestion is performed jointly by multiple nodes, each observing part of the traffic, periodically reporting its measurements to a single centralized server that aggregates them. To avoid communication congestion, each node reports a compressed version of its collected measurements. Traditionally, nodes symmetrically report summaries of the same size computed on their data. We explain that to maximize the accuracy of the joint measurement, nodes should imply various compression ratios on their measurements based on the amount of traffic observed by each node. We illustrate the approach for two common sketches: The Count-Min sketch (CM), which estimates flow frequencies, and the K-minimum-values (KMV) sketch, which estimates the number of distinct flows. For each sketch, we compute node compression ratios based on the traffic distribution. We perform extensive simulations for the sketches and analytically show that, under real-world scenarios, our sketches send smaller summaries than traditional ones while retaining similar error bounds.**

***Index Terms*—Distributed Sketching, Measurement, Stream-Aware, Network Algorithms**

I. Introduction

Network designers need to gather analytics about the performance of the network to better understand what is happening behind the curtain. They rely on them to design traffic engineering, reach a higher utilization of the infrastructure, lower link congestion, and find anomalies when happen. Switches generally do not have sufficient memory to hold entire measurement models for large data streams, therefore the network designer generally sacrifices accuracy for a lower memory footprint. Data sketching algorithms, or *sketches* for short [8], are an indispensable tool for such high-speed low-memory-footprint computations. Sketches estimate some function of a large stream, for example, the frequency of certain items [9], number of observed unique items [10], [15], the top-k most common items [28] or detect frequency changes [24]. They are supported by many data analytics platforms such as PowerDrill [19], Druid [12], Hillview [20], and Presto [29] as well as standalone toolkits [30].

Measurements are conducted in switches and routers all over the network; however, to successfully analyze network behavior, measurement data needs to be gathered in a centralized server that can see the full picture. This is modeled by treating the measured network traffic as a single stream, which is split and ingested by multiple *ingestion nodes*. Once the nodes have a sketch ready to be sent (e.g., they periodically receive a packet signaling the end of their current stream and start a new one) they propagate their local sketch to a *central node* [21], [25], [27]. This is illustrated in Figure 1. The network has to handle the trade-off of sending data packets vs. sending crucial control packets [5], [37]. A way to reduce these control packets is by compressing the sketches before transferring them to a central analytics node.

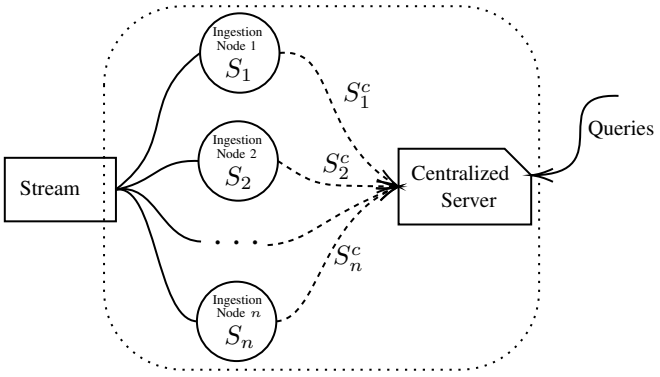

Figure 1. Network-wide measurement over n ingestion nodes: Queries are answered by a centralized server collecting summaries $S_1^c, \ldots, S_n^c$ of the local sketches $S_1, \ldots, S_n$ (e.g., the Count-Min sketch (CM) or the K-minimum-values (KMV) sketch) in the ingestion nodes.

A framework for sketch compression was suggested by Yang et al. [35]. A major component of this framework, named *Maximum Merging Algorithm (**MM**)*, compresses a Count-Min sketch (CM) by utilizing a MAX function and merging multiple cells in one CM to generate a new, smaller sketch. The first limitation of that approach is that it compresses a sketch only into smaller sketches of particular sizes, those that have a common divisor with the size of the original sketch. Moreover, the approach does not provide insights for the selection of the various compression ratios that should be implied for a distributed measurement in multiple nodes. In particular, how the traffic distribution among the nodes should be considered.

A clear drawback of compression methods is the accuracy

[§]The first two authors contributed equally.

ISBN 978-3-903176-39-3

reduction they might imply. It is often critical for network managers to have access to measurements with guarantees of their accuracy. In the common case that the network has multiple ingestion nodes that generate the measurements and send the data to a centralized server, it is intuitive to allow each node to report them through an amount of data that is correlative to the amount of traffic it observes. In this paper, we successfully formulate such compression ratios for a network with any given number of ingestion nodes and any distribution of the traffic through these nodes. Moreover, the resize factors also guarantee that the amount of data that is sent over the network is less than compressing all the data from all the nodes to the same size with previously presented methods.

In this paper, we present the following major contributions.

(i) We motivate a compression method for distributed measurement which is traffic aware. We focus on the CM and describe the traffic-aware Count-Min sketch, denoted TA-CM, that computes the ideal compression ratios for the various nodes for reducing the total amount of reported data. We provide guarantees on the error bounds of the approach.

(ii) As a building block, we develop a compression method named CM-SKTC for a single CM that allows general compression ratios.

(iii) We also present Traffic-Aware K-minimum -values (KMV), denoted TA-KMV – a traffic-aware compression ratio for nodes implementing distributed distinct flow count with the KMV sketch.

(iv) We conduct an experimental study based on real-traffic traces for showing the compression methods perform well under realistic scenarios.

The rest of the paper is organized as follows. Section II discusses the background and related work. In Section III we present the CM-SKTC compression method for a single CM, bound its error, and describe how to decrease the data sent from ingestion nodes to the centralized server. The TA-CM for traffic-aware CM compression for multiple nodes is described in Section IV. Section V presents the TA-KMV for cardinality estimation. Experimental evaluation of the methods is provided in Section VI. Finally, in Section VII we conclude and suggest directions for future work.

II. Background

In this section, we present the Count-Min sketch and the KMV sketch. CM-SKTC and TA-KMV are methods to compress these two sketches. Likewise, we detail related work.

A. The Count-Min Sketch (CM)

Estimating flow size is a required capability in many networking applications, in fields as diverse as accounting, monitoring, load balancing, and filtering, and even beyond networking. Counting the exact size for every flow is often challenging due to a typically large number of active flows at a specific time, making it difficult to maintain a counter-per-flow within a memory accessible at the line rate. There can be two types of errors in the estimation of a flow size: Overestimations and underestimations. The state-of-the-art data structure for flow size estimation is the Count-Min sketch (CM) suggested by Cormode and Muthukrishnan in 2005 [9].

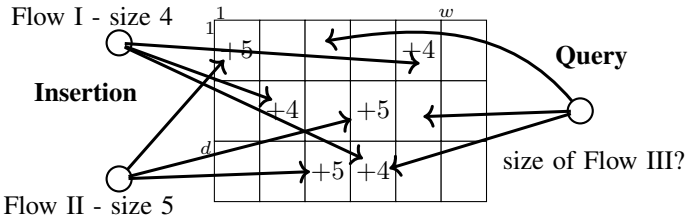

Figure 2. The Count-Min sketch (CM) [9], allowing flow size estimation. A flow size is estimated as the minimum among the counters it is mapped to by a set of hash functions.

The CM relies on a two-dimensional array with w columns and d rows of counters initialized to values of 0s. A set of d hash functions are used to map a flow to d counters, one in each of its rows. Upon a flow arrival, each of these counters is incremented by the length of the flow sequence. To estimate the size of a flow, its d selected counters are considered and the size is estimated as the minimum among these counters. Since multiple flows can contribute to the same counter, the computed value is larger than the exact one in case other flows contributed to all d counters, thus implying an overestimation. CM completely avoids underestimations. A tradeoff exists between the level of accuracy and the allocated memory such that more memory reduces collisions among flows. Similarly, reducing the number of flows improves accuracy.

The CM is illustrated in Figure 2. Flows I, II of size 4, and 5, respectively, are recorded in the sketch (shown on the left side). Each flow increases the value of $d = 3$ counters by its size. The size of Flow III (right side) is estimated by querying the CM as the minimal among the d counters it is mapped to.

While as mentioned the CM can observe overestimations, the guarantee on its accuracy can be described as follows. When using CM with width w and depth d the estimation $\hat{f}$ of flow f satisfies with probability $1 - \delta$

$$\hat{f} \leq f + \epsilon N.$$

Here, N is the number of packets in the measured stream, ϵ holds $w = \lceil \frac{e}{\epsilon} \rceil$ (for Euler's number e) and δ holds $d = \lceil \ln \frac{1}{\delta} \rceil$.

B. KMV (K-minimum-values) Sketch

Another essential capability in network monitoring is the ability to estimate the number of distinct flows, usually called flows' cardinality. Counting the exact number of distinct flows is generally challenging in high traffic rate, making it hard to save some unique data for each flow and quickly identify whether a flow has appeared or not.

The K-minimum-values sketch (KMV) [4], [16] uses a hash function h that maps every flow uniformly to $[0, 1]$. For a parameter k the sketch maintains the minimal k observed values for flows so far. Let $h_1, ..., h_k$ be those k minimal calculated values such that $h_1 < h_2 < ... < h_k$. The *KMV* sketch provides its cardinality estimation as $\frac{k}{h_k}$. This

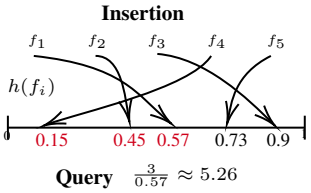

Figure 3. The K-minimum-values sketch (KMV) [4], [16], allowing cardinality estimation. The cardinality is estimated by dividing k with the k'th minimal hashed value, (here $k = 3$ with recorded values shown in red).

estimation is based on the fact that the k order statistics of group uniformly randomized in the range $[0, 1]$ is equal to $\frac{k}{n+1}$. The error of the algorithm is $\frac{1}{\sqrt{k}}$, i.e. for m flows, the KMV sketch estimation m' satisfies $\left|\frac{m-m'}{m}\right| = O(\frac{1}{\sqrt{k}})$.

The KMV is illustrated in Figure 3. There are five flows recorded in the sketch, each maps to a different value in the range of $[0, 1]$. Here $k = 3$ so that only the three minimal values are stored. When flows' cardinality is queried, the sketch finds the largest value that is currently stored, which is 0.57 and returns $\frac{k}{0.57} = \frac{3}{0.57} \approx 5.26$ distinct flows.

C. Related Work

Common sketch solutions focus on the trade-off of speed, accuracy, and memory (e.g., [7], [9], [13], [28], [34] and more). However, they generally consider only a single node ingesting the entire stream. The introduction of software-defined networks (SDN) allows for deploying centralized algorithms for maintaining and collecting information about network operations [36]. These solutions typically have centralized controlled merging of incoming data from ingestion nodes. Network-wide measurements have been widely studied [1], [3], [18], [25]. These solutions, however, do not consider the size of control packets sent to the controller – they do not take into account that these control packets may worsen existing congestion [5], [37].

Recently, [31] suggested methods for accurate flow size estimation in the Count-Min sketch (CM) without overestimation, that apply when the number of flows of non-zero size is bounded. Yang et al. presented the Maximum Merging (**MM**) [35], a method for compressing CM before transmission over the network. The main limitation of this method is that it can only compress the CM to a constant w' which divides the width w. Shrivastava et al. [33] presented time adaptive sketches, and discussed the need for having recent data more accessible. By using the methods shown in this paper, network operators can create larger sketches and thus excess the need in time-reliant sketches. In [17], Harrison et al. presented a network-wide scheme of detecting heavy-hitters, thus while considering the reporting communication overhead. [18] presented a method of heavy-hitters detection that used probabilistic summary reporting to decrease control packets during DDoS attacks. These two works emphasize that minimizing summaries size is critical and can have a major influence on network performance.

Input: A CM S (size $d \times w$), new width w'
Output: Compressed CM-SKTC S^c with size $d \times w'$

```
S^c ← array of 0's of size d × w'
for j = 1; j ≤ d; j++ do
    for i = 1; i ≤ w; i++ do
        l = g_j(i)
        S^c[j][l] = max(S^c[j][l], S[j][i])
    end for
end for
return S^c
```

Algorithm 1: CM-SKTC compression algorithm

Input: A compressed CM-SKTC S^c (size $d \times w'$), a flow f
Output: An estimation of f

```
return min_{j∈[1...d]} {S^c[j][g_j(h_j(f))]}
```

Algorithm 2: CM-SKTC estimation query

UnivMon [27] and NitroSketch [26] summarize streams in a sketch that can later answer multiple measurement tasks. The SKTC compression approach can be generalized to such sketches, while the analysis of the ideal resize factors can be generalized to refer jointly to the accuracy of multiple tasks.

III. The CM-SKTC Compression Method

We present a method that allows compressing any sized CM (d, w) to any new size possible (d, w') for $w' \in [1, w]$. Let $h_1, \ldots, h_d$ be the hash functions used in original CM, such that every function holds $h_i : \{f_1, f_2, \ldots, f_m\} \mapsto \{1, 2, \ldots, w\}$, and let $g_1, ..., g_d$ be hash functions such that $g_i : \{1, 2, \ldots, w\} \mapsto \{1, 2, \ldots, w'\}$ for every i. Algorithm 1 presents the CM-SKTC compression method.

The CM-SKTC compression method works as follows. For each array $S^c[j]$ in the sketch, the value of the l's cell $S^c[j][l]$ is the maximum value over all cells in the corresponding array of the original sketch $S[j]$ that by the hash function g_j are hashed to the l'th cell (i.e. $S^c[j][l] \leftarrow \max_{i:g_j(i)=l} \{S[j][i]\}$).

An example for the compression process of CM-SKTC is illustrated in Figure 4 where a CM-SKTC of size $d = 2, w' = 4$ is computed for a CM of size $d = 2, w = 6$. Notice that the method reduces the number of columns (rather than that of the rows) since typically the number of rows is low beforehand, as there is a need for a distinct hash function per row.

The query method (Algorithm 2) from CM-SKTC is similar to the original CM query. When flow f estimation value is required, for each array $S^c[j]$ (where $j \in [1, d]$) find the appropriate cell $S^c[j][g_j(h_j(f))]$ (one in each array) and return their respective minimal value.

Similar to the original CM and the **MM** [35], the CM-SKTC method also generates only overestimation values.

Yang et al. [35] present error bounds for compressing the CM when reducing w to some divider w' (i.e., $w = z \cdot w'$ for an integer z). The number of counters compressed to the same counter is fixed as w/w'. Our CM-SKTC, however, maps a variable number of counters to each counter using hashing. This number can vary among the counters in an array or among

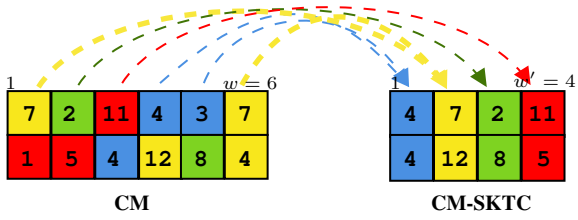

Figure 4. An example for Count-Min sketch (CM) (left side) compression with CM-SKTC (right side). In this example, a CM of size $d = 2, w = 6$ is compressed into a CM-SKTC of size $d = 2, w' = 4$. Values of the hash functions g_1, g_2 are represented by various colors. In row i, multiple counters with the same hash value of g_i are represented by their maximum.

the arrays, thereby adding a taste of randomness. We now analyze the error using this compression method.

Lemma 1 (Single Array compression error)**.** *Given a CM S with d arrays, w counters per array, CM parameters (ϵ, δ), and a CM-SKTC compression ratio $\frac{w'}{w} < 1$. Let N be the size of the stream and denote by $\hat{f}_i$ the estimation of flow with size f_i. Denote by β the term*

$$\left(1 - \left(1 - \frac{1}{\epsilon w}\right)\left(1 - \frac{N}{w(f_i + \epsilon N)}\right)^{\frac{w}{w'} - 1 + \sqrt{-2\ln(1-\delta)\frac{w}{w'}}}\right)^d.$$

The estimation $\hat{f}$ of flow f satisfies

$$\hat{f} \leq f + \epsilon N,$$

with probability $1 - \beta - \delta(1 - \beta)$.

IV. The Traffic-Aware CM in Multiple Nodes

The system has to measure a stream of data that is split in a distributed fashion across several ingestion nodes. Upon stream ingestion completion, the nodes communicate with a centralized server which can then answer queries. This is depicted in Figure 1. Queries are agnostic to the specific architecture, namely, they only refer to the complete stream as a whole and do not refer to its parts observed by each of the nodes. Assume that the channel between the nodes to the centralized server has limited bandwidth. There is a clear tradeoff between the summaries size sent to the server and its ability to answer queries accurately. Accordingly, we would like to optimally use compression to allow high accuracy. **MM** [35] was suggested to reduce summary size through compression of CMs maintained by the nodes but the fact it can be compressed in only particular ratios restricts it from taking advantage of all bandwidth to the server.

In real network traffic, flow size distribution among switches can be imbalanced [23], [32]. For instance, the location of nodes along paths of various lengths or employment of particular network functions in the nodes can result in nodes receiving a small portion of the network stream while others more traffic. We leverage this skew to reduce the summaries size sent to the server. Specifically, we aim to compress the sketches sent by ingestion nodes that received a small portion of the overall stream more than those sketches of nodes with higher portions of traffic. We say that a sketch compression from w columns to w' columns has a *compression ratio* of w'/w. As mentioned, the number of rows is not reduced.

The design of Traffic-Aware Count-Min Sketch (TA-CM) is as follows: Given parameters (ϵ, δ), where ϵ is the desired error, δ is the maximum error probability, we instantiate a CM on every ingestion node with parameters $(\sigma \cdot \epsilon, \delta)$ for some $0 < \sigma \leq 1$. $\frac{1}{\sigma}$ is an enlargement factor and is known in advance to the ingestion nodes and the centralized server. We increase the size of the sketch at the ingestion nodes by $\frac{1}{\sigma}$ and as such decrease their respective error (as the error is tied to the number of columns). By enlarging the sketches at the ingestion nodes, we are able to compress them while maintaining an error within the desired bounds. When the ingestion period ends, the following protocol takes place:

(i) Each ingestion node reports to the central node its local stream size.

(ii) The centralized server computes each ingestion node's compression ratio.

(iii) Every ingestion node compresses its CM with the CM-SKTC method, then sends its summary to the centralized server.

To achieve an overall maximum error of ϵ for an arbitrary flow size estimation, the centralized server can (naively) request a compression ratio of $1/\sigma$. Theorem 1 shows that when there are two ingestion nodes, there are optimal compression ratios that minimize the total summaries size while maintaining the error bounds of the flow size estimation in the centralized server to be at most ϵ.

Theorem 1. *For a given Count-Min sketch parameters ϵ, δ, initial resize factor σ, and two CMs S_1, S_2, with stream sizes of N_1, N_2 respectively, such that w.l.o.g $N_1 \geq N_2$. The TA-CM resize factors $r_1 = \frac{k+1}{\sigma(k+\sqrt{k})}$ and $r_2 = \frac{k+1}{\sigma(\sqrt{k}+1)}$ for $k = N_1/N_2$ generate the minimal amount of network communication such that the error is bounded by ϵ.*

Intuitively, the proof is based on constraints that: (i) the overall error be within the desired bounds, and (ii) the size of sent traffic be smaller than the naive solution (i.e., less than $2 \cdot w \cdot d$ where $w \cdot d$ is the size of a CM with parameters (ϵ, δ)).

As k increases the resize factor of S_2 increases. This is expected, as it has less of an impact on the answer to the query. Note that the summaries size with the optimal resize factors tends to half that of the summaries size sent in the trivial case, therefore utilizing optimal resizing upon uneven stream distribution leads to better usage of network bandwidth.

We now generalize the above result of Theorem 1 for two nodes to the practical case of an arbitrary number of n nodes. We again maintain error bounds within certain limits. The proof is omitted for space constraints.

Theorem 2. *Given are CMs S_i, with stream sizes $N_1 \geq N_2 \ldots \geq N_n$. Denote $k_i = N_i/N_{i+1}$ for $i \in [1, n-1]$ and*

$k_n = 1$. *The optimal TA-CM resize factors are*

$$\forall i \in [1, n-1]: \; r_i = \frac{c_{i+1}}{\sqrt{k_i}} \quad \textit{and} \quad r_n = \frac{\sum_{i=1}^{n} \prod_{j=i}^{n} k_j}{\sigma(\sum_{i=1}^{n} \prod_{j=i}^{n} \sqrt{k_j})}.$$

V. The Traffic-Aware KMV for Cardinality Estimation

We generalize the framework of minimizing sketch summaries, and now present Traffic-Aware KMV (*TA-KMV*), a method to compress the KMV sketch [4], [16]. As discussed, the K-minimum-values sketch is a cardinality estimation sketch. For the case of a single node KMV works as follows: For every flow $f_1, ..., f_m$ it generates hash value $h_1, ..., h_m$ and it saves the k smallest values - $\{h'_1, h'_2, ..., h'_k\}$. The cardinality estimation KMV generates is $\frac{k}{\max_{i \in [1,k]} \{h'_i\}}$. Errors can include over or underestimations of the cardinality and the average error is $\frac{1}{\sqrt{k}}$. We again refer to the scenario from Figure 1 with nodes sending summaries to a centralized server through a channel with bounded bandwidth.

A trivial solution has every node send its k minimal values, resulting in a total number of $n \cdot k$ floating points numbers sent over the network. We aim to better utilize the available bandwidth in this case as well.

Our architecture for this distributed sketch is as follows: Given parameter k, we instantiate *KMV sketch* on the ingestion nodes with parameter $k \ln k$, when the ingestion ends, the following communication takes place:

(i) Ingestion nodes report their cardinality estimation size to the centralized server.

(ii) The central server computes for each ingestion node i its compression ratio n_i.

(iii) Each ingestion nodes sends its n_i minimal values to the centralized server.

We suggest a heuristic of calculating compression ratios n_i. We are unable to provide closed form error bounds for this heuristic. Instead, in Section VI, we show that this method performs well in realistic scenarios.

The goal is for the centralized server to receive some group of k flow hashes such that the group has as large overlap as possible with the group of global k minimal flow hashes. The method works as follows: let e_i be the cardinality estimation of server i, and let $r_i = \frac{e_i}{\sum_{j=1}^{n} e_j}$ be the server estimation ratio. The compression ratios are $n_i = r_i \cdot k \ln k$. In this case the total summaries size is:

$$\sum_{i=1}^{n} n_i = \sum_{i=1}^{n} k \cdot r_i \cdot \ln k = k \ln k \sum_{i=1}^{n} \frac{e_i}{\sum_{j=1}^{n} e_j} = k \ln k$$

The $\ln k$ factor is needed due to collisions – two nodes i, j may receive the same flow with some small hash and both send it. As the hash is only saved once, the centralized node ends with less than k hashes. We consider the coupon collector problem [6], [14] and its solutions, such that the coupons are the k smallest hashes. For the centralized server to have these smallest hashes (with high probability) all ingestion nodes must send a total of at least $k \ln k$ values. The evaluation shows that this method performs well in practical scenarios, and also shows that one can change this ratio in order to achieve a trade-off between precision and bandwidth usage. Note that the coupon collector assumes each coupon is drawn independently from a uniform distribution which is not the case in our scenario. In the analyzed scenarios, the centralized server generally ended with more than the k smallest hashes.

VI. Evaluation

In this section, we evaluate the error bounds of both the CM-SKTC, TA-CM, and the TA-KMV compression methods

A. Comparison of the CM-SKTC vs Maximum Merging Algorithm

First, we compare the CM-SKTC to two different sketches. (1) regular CM with $d = 4$ and varying total memory usage, where each cell is 4 bytes. (2) Maximum Merging (MM) of [35] with an initial 2 MB CM. The evaluation metric we used is *Average Relative Error (ARE)*, defined for a set of flows $\{f_1, ..., f_n\}$ as $\frac{1}{n} \sum_{i=1}^{n} \frac{|\hat{f}_i - f_i|}{f_i} = \frac{1}{n} \sum_{i=1}^{n} \frac{\hat{f}_i - f_i}{f_i}$ where $\hat{f}_i$ is the estimated value of flow f_i. Recall that only overestimations can occur.

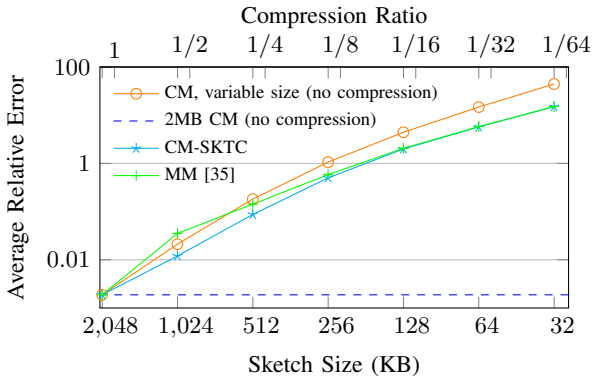

Figure 5. Single node: Compression methods comparison

Figure 5 compares the different compression methods for different sketch sizes. Traces evaluated in this figure have ∼35000 flows, and the blue-dashed baseline represents the *ARE* of the 2 MB CM sketch that both compression methods initiate from. One can observe that CM-SKTC outperforms the two other methods consistently. Furthermore, as the compression ratio decreases and the summary size increases, the *ARE* improves, as expected.

In Figure 6 we depict the same comparison as Figure 5; however the number of flows is ∼90000, and only to the 50 largest flows *ARE* in each trace are considered. In this graph, one can observe that the top flows *ARE* is extremely low for all methods. Moreover, the MM and CM-SKTC curves are similar, and both have the same error as the 2MB sketch.

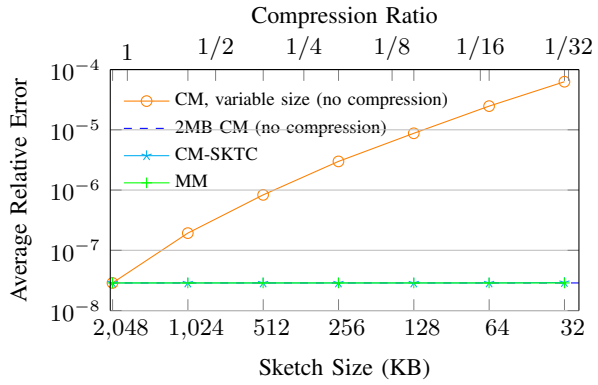

Figure 6. Single node: Compression methods comparison - top 50 flows

In Table I we compare the time of both the compression process and the query of the MM and CM-SKTC compression. The MM is more time-efficient than the CM-SKTC compression. This is due to each hash being calculated twice (for the insertion and for the compression), and therefore we expect the CM-SKTC to be roughly twice as slow as the MM.

Measurement \ Method	**Maximum Merging**	**CM-SKTC**
Average query time	4.75 μs	10.2 μs
Compression time	0.119 s	0.561 s

Table I
TIMING COMPRESSION

From Figures 5 and 6, we deduce that the CM-SKTC has two important traits: (1) CM-SKTC achieves estimations within the required error parameters using smaller summaries, and (2) For large (elephant) flows this error is negligible.

B. The TA-CM with two ingestion nodes ($n = 2$)

We now simulate two local nodes receiving a data stream of size N, where the relation between the size of the data stream N_1 processed at the first node, and the size of the data stream N_2 at the second node is $k = \frac{N_1}{N_2}$. We evaluate the effect of k on the *ARE*.

Figure 7 depicts the *ARE* of merging two sketches when sending different sizes over the network. We compare locally building two sketches with error ϵ, such that they each have size 1MB (meaning 2MB of data is sent over the network), and building larger local sketches with error $\sigma\epsilon$ and compressing them. We compare the trivial compression by factor $1/\sigma$ compared to using our optimal resize factors, for $\sqrt{k} = 3, 7, 10$. Note that our comparison shows that the error of resizing using optimal factors falls in between the error of starting with error ϵ and trivially compressing with error $\sigma\epsilon$. Of great importance is that even in the worst case the error is less than ϵ. Table II compares the summaries size across the network. It follows that there is a trade-off between the accuracy and the summaries size. Figure 8 shows the ratio between summaries size as a function of k, in relation to trivial compression.

In Figure 9 we show the results of compressing the same base CM sketch as in Figure 7. However, in this simulation, we compare the results when the total summaries size is 2MB, i.e., the summaries account for 2MB of network traffic. In this case, we observe that the TA-CM *ARE* is better than the MM *ARE*. TA-CM outperforms MM as it is traffic-aware and considers the distribution across the nodes and calculates the ratios accordingly; it helps to send the larger part of the data from the node that handled the larger chunk of the stream.

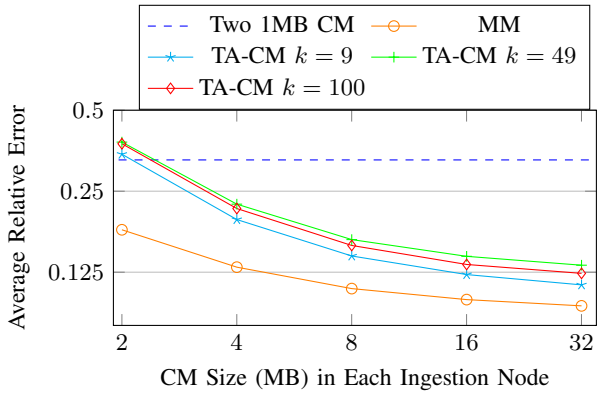

Figure 7. Two nodes: Compression methods comparison

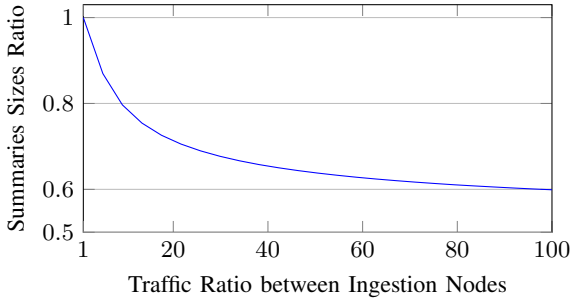

Figure 8. Ratio between summaries size in multiple compression methods

Compression Method	**Summaries Size Sent (% from max)**
Maximum Merging	2 MB (100%)
TA-CM, $k = 1$	2 MB (100%)
TA-CM, $k = 9$	1.6 MB (80%)
TA-CM, $k = 49$	1.28 MB (64.3%)
TA-CM, $k = 100$	1.18 MB (59%)

Table II
COMPRESSION RATIO BY k

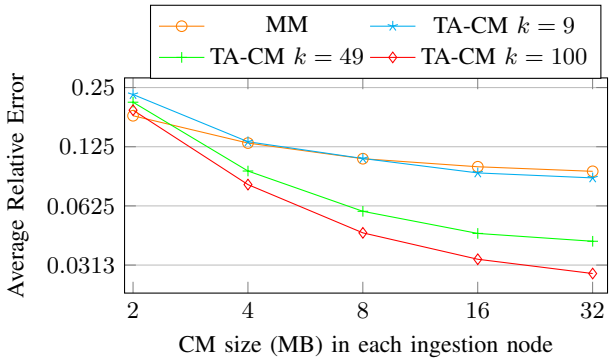

Figure 9. Two nodes: Compression methods comparison. Allowance of 2MB sent from ingestion nodes to centralized server.

C. The TA-CM with n ingestion nodes

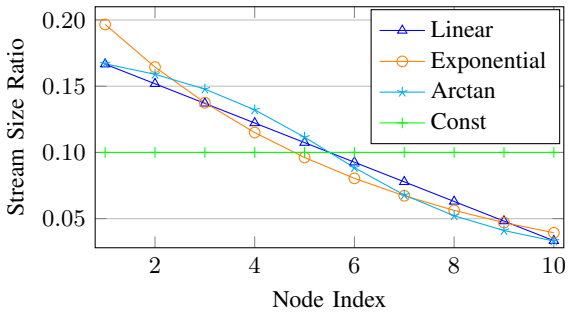

Figure 10. $n = 10$ nodes: Stream size distribution over the nodes

To evaluate TA-CM in multiple-ingestion nodes scenarios, we formulate four different types of distributions for 10 servers (see Figure 10) and compare the TA-CM to MM and the non-compressed CM sketch. The distributions were chosen such that the ratio between the largest server ratio to the smallest server ratio is 5 (i.e., $N_1/N_{10} = 5$).

The CM sketch base size for each of the 10 servers is 32KB. We compare the *ARE* with multiple values of σ (i.e., the ratio by which the ingestion node CM sizes is increased) and compressing with two methods: (1) TA-CM with ratios computed in Section IV (2) MM compression with all ratios are $1/\sigma$. As depicted in Figure 11, TA-CM achieves similar results in terms of *ARE* to the Maximum Merging compression and improves the results of the basic non-compressed CM. However, it does so while decreasing the total summaries size. Table III indicates that the TA-CM saves between 7% to 9% of the total summaries size for chosen distributions. This saving ratio can be increased by choosing other, wider distributions of the stream (for example if the stream distributes across the ingestion nodes by Pareto distribution then TA-CM potentially saves an even higher percentage).

Distribution	Summaries Size (% from max)
Maximum Merging	320KB (100%)
Constant Dist.	320KB (100%)
Exponential Dist.	$\sim$ 292KB (91.3%)
Linear Dist.	$\sim$ 298KB (93.3%)
Arctan Dist.	$\sim$ 293KB (91.6%)

Table III
COMPRESSION RATIO OF VARIOUS DISTRIBUTIONS

D. The TA-KMV for Cardinality Estimation

In this section, we evaluate *TA-KMV*. We use the same method as in the previous section to generate the input stream. However, for this section, we used only the linear distribution. We compare our *TA-KMV* with multiple compression ratios. The compression ratio is measured by *Total Hash-values Sent (THS)*. To the best of our knowledge, no compression scheme is available for this sketch, and therefore we compare our method only to the baseline, i.e., each ingestion node sends k hash values to the centralized server. Our measurement unit is

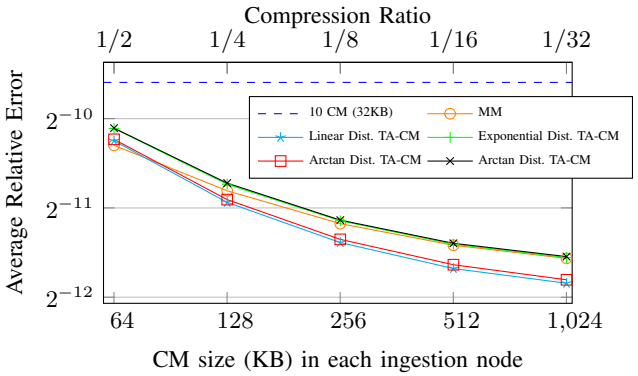

Figure 11. $n = 10$ nodes: Compression method comparison - various distributions

the estimation precision rate to true cardinality. In this case, the number of hash values that are sent to the centralized server is $n \cdot k$, where n is the number of ingestion nodes.

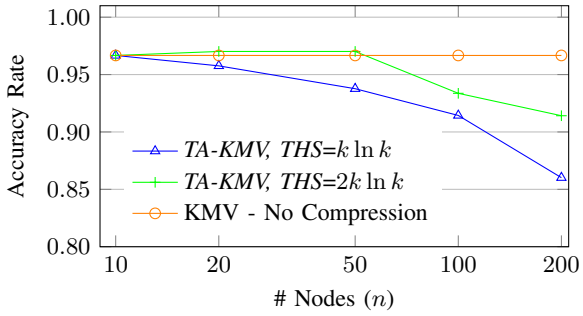

Figure 12. TA-KMV vs baseline - $\sim$15000 flows, $k = 1024$

Plot	THS sent for 200 ingestion nodes (% from max)
KMV - No Compression	204800 (100%)
TA-KMV, THS=$k \ln k$	7098 ($\sim$ 3%)
TA-KMV, THS=$2k \ln k$	14196 ($\sim$ 6%)

Table IV
RECORDS SENT FOR VARIOUS COMPRESSION METHODS

Figure 12 depicts the impact of the number of ingestion nodes on the accuracy rate of TA-KMV. We define the accuracy rate as $\frac{\text{Estimation}}{\text{\# Flows}}$. The streams measured in this figure contain $\sim$15,000 flows. This figure indicates that when the number of ingestion nodes across the network is relatively small, the performance of TA-KMV is similar to that of the baseline, but when the number of ingestion node increases the accuracy rate decreases, as expected. This is due to the increasing difference in total summaries sizes ($O(nk)$ in the no-compression case *vs* $O(k \ln k)$ in TA-KMV). This creates a clear trade-off between the accuracy rate and the total summaries size. In Table IV we show that although TA-KMV accuracy slightly decreased, it saves more than 90% of the total hash values sent over the network. It allows network operators to decide whether they prefer to lose some accuracy and increase the possible bandwidth over the network, or increase the accuracy and pay more in management packets.

VII. Conclusion

In this paper, we presented the problem of merging data from multiple measurement points to one centralized server, described a distributed traffic-aware sketching scheme, and applied it to two unique sketches. We presented the CM-SKTC sketch as a simple, yet efficient method for compressing the CM sketch to any desired size, then used this method to generate *TA-CM*, a new scheme for flow-size measurements that provides high accuracy and decreases the total summaries size sent to the centralized server by considering the traffic of each node. Finally, we analyzed these sketches under multiple network settings and examined the trade-off between the accuracy and the size of summaries.

Several directions can be the focus in future work. A straightforward extension is developing compression algorithms for additional kinds of sketches allowing different measurement tasks (e.g., Quantiles for rank estimation [2]). Likewise, we wish to design further compression of sketches by leveraging existing generic compression techniques such as Huffman codes, LZ77 or gzip [11], [22], [38].

VIII. Acknowledgment

This work was partially supported by the Technion Hiroshi Fujiwara Cyber Security Research Center and the Israel National Cyber Directorate, by the Alon fellowship, by German-Israeli Science Foundation (GIF) Young Scientists Program, by the Taub Family Foundation as well as and by the Polak Fund for Applied Research, at the Technion.

References

[1] Yehuda Afek, Anat Bremler-Barr, Shir Landau Feibish, and Liron Schiff. Detecting heavy flows in the SDN match and action model. *Computer Networks*, 136:1–12, 2018.

[2] Pankaj K Agarwal, Graham Cormode, Zengfeng Huang, Jeff M Phillips, Zhewei Wei, and Ke Yi. Mergeable summaries. *ACM Transactions on Database Systems (TODS)*, 38(4):1–28, 2013.

[3] Daniel Anderson, Pryce Bevan, Kevin Lang, Edo Liberty, Lee Rhodes, and Justin Thaler. A high-performance algorithm for identifying frequent items in data streams. In *ACM Internet Measurement Conference*, 2017.

[4] Ziv Bar-Yossef, TS Jayram, Ravi Kumar, D Sivakumar, and Luca Trevisan. Counting distinct elements in a data stream. In *International Workshop on Randomization and Approximation Techniques in Computer Science*, 2002.

[5] Theophilus Benson, Ashok Anand, Aditya Akella, and Ming Zhang. Understanding data center traffic characteristics. *ACM SIGCOMM Computer Communication Review*, 40(1):92–99, 2010.

[6] Arnon Boneh and Micha Hofri. The coupon-collector problem revisited—a survey of engineering problems and computational methods. *Stochastic Models*, 13(1):39–66, 1997.

[7] Graham Cormode. Sketch techniques for approximate query processing. *Foundations and Trends in Databases. NOW publishers*, 2011.

[8] Graham Cormode, Minos Garofalakis, Peter J Haas, and Chris Jermaine. Synopses for massive data: Samples, histograms, wavelets, sketches. *Foundations and Trends in Databases*, 4(1–3):1–294, 2012.

[9] Graham Cormode and S. Muthukrishnan. An improved data stream summary: The Count-Min sketch and its applications. *J. Algorithms*, 55(1):58–75, 2005.

[10] Mayur Datar and Piotr Indyk. Comparing data streams using hamming norms. In *International Conference on Very Large Databases (VLDB)*, 2002.

[11] Peter Deutsch et al. Gzip file format specification version 4.3. Technical report, RFC 1952, May, 1996.

[12] Druid. Druid. https://druid.apache.org/blog/2014/02/18/hyperloglog-optimizations-for-real-world-systems.html.

[13] Cristian Estan and George Varghese. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems (TOCS)*, 21(3):270–313, 2003.

[14] William Feller. An introduction to probability theory and its applications. *1957*.

[15] Philippe Flajolet and G Nigel Martin. Probabilistic counting. In *IEEE Annual Symposium on Foundations of Computer Science (SFCS)*, 1983.

[16] Frédéric Giroire. Order statistics and estimating cardinalities of massive data sets. *Discrete Applied Mathematics*, 157(2):406–427, 2009.

[17] Rob Harrison, Qizhe Cai, Arpit Gupta, and Jennifer Rexford. Network-wide heavy hitter detection with commodity switches. In *ACM Symposium on SDN Research (SOSR)*, 2018.

[18] Rob Harrison, Shir Landau Feibish, Arpit Gupta, Ross Teixeira, S Muthukrishnan, and Jennifer Rexford. Carpe elephants: Seize the global heavy hitters. In *ACM Workshop on Secure Programmable Network Infrastructure*, 2020.

[19] Stefan Heule, Marc Nunkesser, and Alexander Hall. Hyperloglog in practice: Algorithmic engineering of a state of the art cardinality estimation algorithm. In *International Conference on Extending Database Technology*, 2013.

[20] Hillview. Hillview: A Big Data Spreadsheet. https://research.vmware.com/projects/hillview.

[21] Qun Huang, Xin Jin, Patrick PC Lee, Runhui Li, Lu Tang, Yi-Chao Chen, and Gong Zhang. SketchVisor: Robust network measurement for software packet processing. In *ACM SIGCOMM*, 2017.

[22] David A Huffman. A method for the construction of minimum-redundancy codes. *IEEE Proc. of the IRE*, 40(9):1098–1101, 1952.

[23] Nanxi Kang, Monia Ghobadi, John Reumann, Alexander Shraer, and Jennifer Rexford. Efficient traffic splitting on commodity switches. In *ACM CoNEXT*, 2015.

[24] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. Sketch-based change detection: Methods, evaluation, and applications. In *ACM Internet Measurement Conference*, 2003.

[25] Yuliang Li, Rui Miao, Changhoon Kim, and Minlan Yu. FlowRadar: A better NetFlow for data centers. In *USENIX NSDI*, 2016.

[26] Zaoxing Liu, Ran Ben-Basat, Gil Einziger, Yaron Kassner, Vladimir Braverman, Roy Friedman, and Vyas Sekar. NitroSketch: Robust and general sketch-based monitoring in software switches. In *ACM SIGCOMM*, 2019.

[27] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. One sketch to rule them all: Rethinking network flow monitoring with UnivMon. In *ACM SIGCOMM*, 2016.

[28] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *International Conference on Database Theory*, 2005.

[29] Presto. HyperLogLog in Presto: A significantly faster way to handle cardinality estimation. https://engineering.fb.com/data-infrastructure/hyperloglog/.

[30] Yahoo! Research. Apache DataSketches (Incubating). https://incubator.apache.org/clutch/datasketches.html.

[31] Ori Rottenstreich, Pedro Reviriego, Ely Porat, and S. Muthukrishnan. Avoiding flow size overestimation in the count-min sketch with bloom filter constructions. *IEEE Transactions on Network and Service Management (TNSM)*, 2021.

[32] Yaniv Sadeh, Ori Rottenstreich, Arye Barkan, Yossi Kanizo, and Haim Kaplan. Optimal representations of a traffic distribution in switch memories. In *IEEE INFOCOM*, 2019.

[33] Anshumali Shrivastava, Arnd Christian Konig, and Mikhail Bilenko. Time adaptive sketches (ada-sketches) for summarizing data streams. In *International Conference on Management of Data*, 2016.

[34] Vibhaalakshmi Sivaraman, Srinivas Narayana, Ori Rottenstreich, Shan Muthukrishnan, and Jennifer Rexford. Heavy-hitter detection entirely in the data plane. In *ACM Symposium on SDN Research (SOSR)*, 2017.

[35] Tong Yang, Jie Jiang, Peng Liu, Qun Huang, Junzhi Gong, Yang Zhou, Rui Miao, Xiaoming Li, and Steve Uhlig. Elastic sketch: Adaptive and fast network-wide measurements. In *ACM SIGCOMM*, 2018.

[36] Minlan Yu, Lavanya Jose, and Rui Miao. Software defined traffic measurement with OpenSketch. In *USENIX NSDI*, 2013.

[37] Zheng Zhang, Ming Zhang, Albert G Greenberg, Y Charlie Hu, Ratul Mahajan, and Blaine Christian. Optimizing cost and performance in online service provider networks. In *USENIX NSDI*, 2010.

[38] Jacob Ziv and Abraham Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on information theory*, 23(3):337–343, 1977.