Precise Error Estimation for Sketch-based Flow Measurement

Peiqing Chen[★*], Yuhan Wu[†*], Tong Yang[†], Junchen Jiang[‡], Zaoxing Liu[★]
[★]*Boston University,* [†]*Peking University,* [‡]*University of Chicago*

Abstract

As a class of approximate measurement approaches, sketching algorithms have significantly improved the estimation of network flow information using limited resources. While these algorithms enjoy sound error-bound analysis under worst-case scenarios, their actual errors can vary significantly with the incoming flow distribution, making their traditional error bounds too "loose" to be useful in practice. In this paper, we propose a simple yet rigorous error estimation method to more precisely analyze the errors for *posterior* sketch queries by leveraging the knowledge from the sketch counters. This approach will enable network operators to understand how accurate the current measurements are and make appropriate decisions accordingly (*e.g.*, identify potential heavy users or answer "what-if" questions to better provision resources). Theoretical analysis and trace-driven experiments show that our estimated bounds on sketch errors are much tighter than previous ones and match the actual error bounds in most cases.

CCS Concepts

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

Keywords

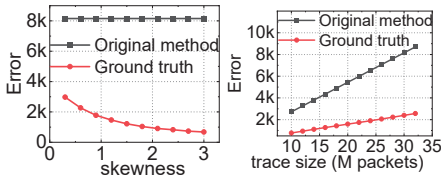
Sketch, Error Estimation, Network Algorithm

ACM Reference Format:
Peiqing Chen, Yuhan Wu, Tong Yang, Junchen Jiang, Zaoxing Liu. 2021. Precise Error Estimation for Sketch-based Flow Measurement. In *ACM Internet Measurement Conference (IMC '21), November 2–4, 2021, Virtual Event, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3487552.3487856

1 Introduction

Recent advances in sketching algorithms (sketches) enable approximate flow measurement using small memory footprints. At a high level, these algorithms are designed to collect approximate flow information (*e.g.*, flow sizes and byte counts) as a small "sketch" summary from resource-constrained network devices (*e.g.*, switches and NICs). These measurement results are crucial to various decision-making processes and system performance predictions, such as traffic engineering [2, 16] and load balancing [1, 14, 17, 22].

While tremendous efforts have been made towards optimizing the sketch accuracy for supported tasks [12, 20, 28] or designing novel sketches for new tasks [19, 24, 31], a fundamental issue in sketching is *how to estimate the errors of the sketch results*. Once a sketch is configured with certain memory, the actual errors of its results can vary significantly depending on the workload distributions. For instance, when the workload is "skewed" with large flows accounting for a large portion of the traffic, the actual errors of these large flows can be significantly lower than the error bounds derived from the traditional worst-case-based analysis. Figure 1 shows an example of Count-Min sketch, and error bounds are similarly overestimated in other sketches too (more details in §2). The question we then ask in this paper is ***can we precisely estimate the errors of sketch results at the query time?***



(a) Synthetic trace **(b) Real-world trace**

Figure 1: Traditional error bounds are not precise.

We tackle this problem by introducing a simple yet rigorous mechanism to precisely estimate error bounds without prior knowledge about the workload distribution. The key insight is that the "after-the-fact" counter values in the sketches can be highly indicative of the actual errors of the current workload. Specifically, when querying a Count-Min Sketch of r counter arrays, our technique traverses all counters in an array and uses the $(\delta^{\frac{1}{r}})$-*fractile* largest value among all counters as the estimated error bound with confidence 1-δ (*i.e.*, actual errors fall in the bound with 1-δ probability). In contrast, traditional error bounds of sketches [6, 8] are derived *before* the measurement starts, which makes them dependent on the worst-case input and oblivious to the actual flow distribution, causing the resulting error bound to be higher than the actual errors in most cases. We formally prove that our method yields tighter (near-optimal) error bounds than traditional ones (§3) and empirically show that the our error bounds match the real errors well in practice (§4).

By further extending this technique to other sketches (e.g., Count-Sketch [6]), we demonstrate that it is a general insight for a variety of sketches. Our trace-driven evaluation with three representative sketches (Count-Min, Count-Sketch, CU-Sketch) in four types of network workloads (*e.g.*, WAN [5], data center [4], DDoS attack [21], Zipf-synthetic [25]) show that our approach is 20 to 700 times more accurate than the traditional worst-case error bounds under various workload distributions.

While our contribution is on the theoretical front, it bears significant practical implications, as we envision a wide spectrum of potential sketch-based applications (decision making and performance prediction) enabled by tighter error bounds on sketch results at the query time. Consider two examples. (1) In-network caching and load balancing in CDN use sketches to identify hot

*Equal contribution.

IMC '21, November 2–4, 2021, Virtual Event, USA

ACM ISBN 978-1-4503-9129-0/21/11…$15.00
https://doi.org/10.1145/3487552.3487856

objects [14, 17]. A loose error bound gives network operators less confidence in the sketch results, forcing them to reserve more memory for worst-case workloads, rather than the actual workload. In contrast, by offering more precise error bounds, our method can help operators save cache space and balance the loads more efficiently. (2) When deploying telemetry capabilities, operators often need to meet specific measurement accuracy. With loose error bounds, operators will have to commit more hardware resources to run the sketches to ensure worst-case sketch accuracy. In contrast, by enabling precise runtime error analysis based on sketch counters, our method can help operators identify the best resource configurations for the actual traffic workload. We demonstrate the real-world benefits of a more precise error estimator in the first scenario using the simulator of [17]: our precise error estimation increases the confidence of sketch-based performance prediction and reduces resource overprovisioning.

In summary, the main contributions of this paper are:

- We introduce a new precise online error estimator for sketches using the insight from sketch counters.
- We rigorously prove that our approach yields tighter bounds than the prior worst-case error bounds.
- We evaluate our approach in a number of real-world traces representing different types of network traffic and conduct a case study to demonstrate the potential practical benefits of tighter error bounds.

2 Background

We begin with the background of *sketches*, followed by the limitations of existing approaches to sketch error estimation.

Sketches for flow measurement: Sketches are known to be useful in providing various flow measurement statistics, such as heavy hitters [6, 8, 10, 18, 30], change detection [15, 18], and hierarchical heavy hitters [3, 7]. In this paper, we consider popular sketches for measuring flow sizes, including Count-Min Sketch (CM) [8], Conservative-Update Sketch (CU) [10], and Count Sketch (CS) [6].

To make the discussion more concrete, we use CM as an example. At a high level, CM maintains r arrays (rows) of w counters (columns). When receiving a packet with flow identity k (*e.g.*, 5-tuple), CM computes r pairwise independent hash values based on k. Each hash value provides an offset within one of the r arrays in CM, and then we increment the corresponding counters. Since w is usually much smaller than the number of flows (to reduce memory footprints), different flows may update the same counters (*i.e.*, collisions), resulting in over-estimation errors. Thus, when querying the size of flow k, we locate r counters using the same r hash functions that are used to update the sketch counters and return the minimum among the r counters. This minimum counter represents an estimated flow size with minimum over-estimation.

Limitation of worst-case error estimation: Let us consider a CM sketch configured with $w = O(\frac{1}{\epsilon})$, $r = O(\log(\frac{1}{\delta}))$. It has been shown [8] that an estimated size of each flow k falls in $[f(k), f(k) + \epsilon|F|_1)$ with probability $1 - \delta$, where $f(k)$ and $|F|_1$ are the actual size of flow k and the total size of all flows, respectively. This error bound $\epsilon|F|_1$ was derived by constructing the "worst case": δ portion of flows have $\epsilon|F|_1$ estimation errors and the other flows have zero errors. This worst case maximizes the number of flows that have error of $\epsilon|F|_1$. Since the total size of all counters in one array is $|F|_1$, to reach the aforementioned worst case, $\frac{1}{\epsilon}$ counters should have the same flow counts, while the remaining counters are zero.

While the worst-case error bound of $\epsilon|F|_1$ offers an *a priori* error estimation (*i.e.*, before measurements start), it is more useful to estimate the *actual* errors in practice, because they can be much lower than the worst-case error bound. To confirm it, we evaluate the worst-case error bound in CM using synthetic traces of 30M packets with skewnesses ranging from 0.3 to 3.0 and real-world traces [5] of 10M to 30M packets. CM is configured with 4 rows each containing 10,000 counters. We use $\epsilon = 0.0027$ and $\delta = 0.018$ in calculating the error bounds. As depicted in Figure 1, there is a large gap between the worst-case bound provided in the original paper [8] and the actual (ground-truth) error bound[1].

Limitation of related work: The closest efforts to us are SCREAM [23] and D. Ting's error estimator [29]. SCREAM is a dynamic resource management system for sketches. To meet a given accuracy requirement, SCREAM provides an accuracy estimator for the recall and precision of heavy hitter detection by taking into account the skewness in the sketch output. However, SCREAM only separates the estimates of large flows and those of small flows, and the flow size estimation for large (or small) flows still has the same worst-case error bound as in [8]. Thus, they cannot precisely estimate the flow size errors. Though D. Ting's work [29] also derives an improved error bound that is proven to be tighter than the original one in [8], they do not achieve a near-optimal bound, since they only utilize unused counters to estimate the error distribution, whereas our method leverages all counter information in the sketch. Moreover, D. Ting's method is more compute-intensive than ours due to its use of Likelihood Estimation.

3 Precise Error Estimator

In this section, we start by describing our error estimation algorithms and then provide rigorous proofs that they always yield tighter bounds. The symbols used are summarized in Table 1.

3.1 Error Estimation Algorithms

We give our error estimation algorithm for Count-Min (CM) in Algorithm 1 and defer the algorithm and its analysis for CU-Sketch and Count-Sketch (CS) to Appendix A.1. Given a CM-Sketch with r rows of w counters (*i.e.*, a counter matrix $A[row][column]$), our goal is to provide a precise error bound by only accessing matrix A.

Algorithm 1: Error Estimator for CM.

Input: $A[1 \dots r][1 \dots w]$ with confidence $1 - \delta$.

1 $p \leftarrow \delta^{\frac{1}{r}}$;

2 SortToDescendingOrder($A[1][1] \dots A[1][w]$);

Output: $A[1][\lceil wp \rceil]$

Let $g(\delta)$ be the ground-truth error bound that we want to estimate (*i.e.*, the actual error is below $g(\delta)$ with $1 - \delta$ probability). In Algorithm 1, let p be the probability that the error of the corresponding counter in each row is no less than $g(\delta)$, *i.e.*, not bounded

[1] Given an arbitrary set of flows $\mathcal{F} = \{e_1, e_2, \dots, e_n\}$, we calculate the actual flow-size estimation error x_i on flow e_i and return the $1 - \delta$ largest fractile of $x_1, x_2, \dots, x_n$ as the actual (ground-truth) estimation error.

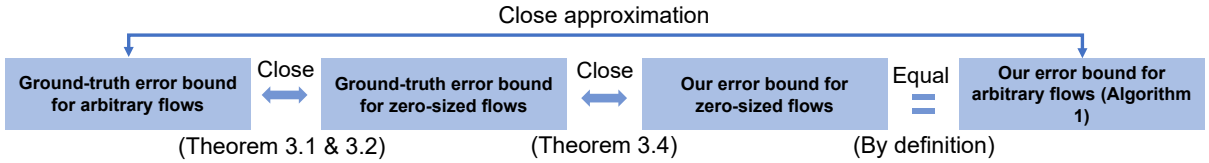


Figure 2: The key steps in the proof that our error bound (Algorithm 1) is close to the ground-truth error bound.

Symbol	Description
$\mathcal{F}$	traffic stream $\mathcal{F} = \{(e_i, f_i)\}$
e_i	flow identifier of e_i
f_i	size of flow e_i
X_{e_i}	the additive error of flow e_i in one sketch row [2]
Y	the additive error of a non-existent flow (size 0)
$I_{i,j}$	a variable indicating flow e_i and e_j collides
$F_Y(.)$	the cumulative distribution function of Y
$g_{e_i}(\delta)$	ground-truth error bound for flow e_i with confidence δ
r	number of rows in Count-Min Sketch
w	number of counters in each row
ξ_p	largest p-fractile of $F_Y(.)$
$\hat{\xi}_x$	x-th largest of counter values
$\|F\|_1$	the total size of flows in $\mathcal{F}$

Table 1: List of symbols and notations.

by $g(\delta)$. Then the probability δ that all r corresponding counters are not bounded by $g(\delta)$ is p^r. Therefore, $p = \delta^{\frac{1}{r}}$. To estimate $g(\delta)$, we pick an arbitrary row (e.g., row 1: $A[1][1] \ldots A[1][w]$), sort all its counters by non-ascending order, and report the $\lfloor wp \rfloor$-th largest counter as our estimation for $g(\delta)$.

3.2 Analysis

Next, we will prove that this error estimation algorithm provides an error bound that is close to the ground-truth error bound under all circumstances.

Proof outline: For ease of understanding, we begin by sketching the key steps (illustrated in Figure 2) in our formal proof:

- **Step 1:** We first show in Theorem 3.1 that the ground-truth error bound for *non-zero flows* (those that exist in the traffic) is smaller than that of *zero-sized flows* (those that do not appear in the traffic). An intuitive explanation is that the error of flow e_a grows with the total size of other flows except e_a, so the error is at the highest when e_a itself is zero-sized. Further, we will show in Theorem 3.2 that the ground-truth error bounds for any flows are similar. Together, they imply that the ground-truth error bound of an arbitrary flow (the first block in Figure 2) is tightly upper bounded by the ground-truth error bound of a zero-sized flow (the second block in Figure 2).
- **Step 2:** We then show in Theorem 3.4 that for zero-sized flows, the difference between ground-truth error bound (the second block in Figure 2) and our error bound produced by Algorithm 1 (the third block in Figure 2) can be tightly bounded. Finally, since Algorithm 1 by definition returns the same error bound for all flows, our error bound for arbitrary flows (the last block in Figure 2) closely approximates the real ground-truth error bound (the first block in Figure 2). In particular, Equation (5) gives a formal bound of this approximation.

[2]For a flow e_i and one row in the CM-Sketch, the additive error is the difference between the hashed counter value and f_i.

Step 1: Ground-truth error bounds for any flows are similar. In this step, we first show that the ground-truth error bound for any non-zero flow is smaller than the ground-truth error bound for a zero-sized flow (Theorem 3.1). We then prove that the difference between both ground-truth bounds is small (Theorem 3.2).

For arbitrary flow e_a, its additive error is $\mathbb{X}_{e_a}$. For the j^{th} row of a sketch, let $X_{e_a}^{(j)}$ be the additive error of flow e_a in the j^{th} row. For CM-Sketch, $X_{e_a}^{(j)} = \sum_{e_i \in S \setminus \{e_a\}} f_i \cdot I_{a,i}$, where S is the set of all flows and $I_{a,i}$ indicates whether flow e_a and flow e_i are hashed to the same counter by $h_j(.)$. The value of $I_{a,i}$ is either 0 or 1. The additive estimation error of e_a is the minimum error among all rows of the sketch, *i.e.*, $\mathbb{X}_{e_a} = min_{j=1}^{r}\{X_{e_a}^{(j)}\}$.

The ground-truth error bound of arbitrary flow e_a is denoted as $g_{e_a}(\delta)$, which satisfies:

$$\Pr\left[\mathbb{X}_{e_a} > g_{e_i}(\delta)\right] = \delta, \forall \delta \tag{1}$$

For zero-sized flow e_0 (*i.e.*, $f_0 = 0$), the additive error is $\mathbb{X}_{e_0}$ and its ground-truth error bound is $g_{e_0}(\delta)$. Next, we prove that the ground-truth error bound for zero-sized flows is no less than the ground-truth error bound for arbitrary flows.

Theorem 3.1. *Given the traffic stream $\mathcal{F}$ and the sketch configurations (i.e., w counters in each row and r rows). We have*

$$g_{e_a}(\delta) \leqslant g_{e_0}(\delta), \forall e_a \tag{2}$$

Proof. Since $X_{e_a}^{(j)} = \sum_{e_i \in S \setminus \{e_a\}} f_{e_i} I_{a,i}$ and $X_{e_0}^{(j)} = \sum_{e_i \in S} f_{e_i} I_{0,i}$, we have $\forall t, \Pr\left[X_{e_a}^{(j)} > t\right] \leqslant \Pr\left[X_{e_0}^{(j)} > t\right]$. Therefore,

$$\Pr\left[\mathbb{X}_{e_a} > t\right] \leqslant \Pr\left[\mathbb{X}_{e_0} > t\right], \forall t$$

From the definition of g_{e_a}, we have

$$g_{e_a}(\delta) \leqslant g_{e_0}(\delta), \forall e_a$$

□

As Theorem 3.1 only shows non-zero flows having a larger ground-truth error bound than that of zero-sized flows, we now argue that any non-zero flow's ground-truth error bound can be closely approximated by zero-sized flow's ground-truth error bound.

Theorem 3.2. *$\forall t$ as a bound, an arbitrary flow e_a (can be a zero-sized flow) and e_0 , $0 \leqslant F_{\mathbb{X}_{e_a}}(t) - F_{\mathbb{X}_{e_0}}(t) \leqslant \frac{r}{w}$, where $\mathbb{X}_{e_a}$ is the error of flow e_a and $F(.)$ is the Cumulative Distribution Function (CDF) of errors.*

Proof. Recall that $X_{e_a}^{(j)} = \sum_{e_i \in S \setminus \{e_a\}} f_{e_i} I_{a,i}$.

We have

$$F_{X_{e_a}^{(j)}}(t) = \Pr\left[X_{e_a}^{(j)} \leqslant t\right]$$

$$F_{X_{e_0}^{(j)}}(t) = \Pr\left[X_{e_a}^{(j)} \leqslant t\right]\left(1-\frac{1}{w}\right) + \Pr\left[X_{e_a}^{(j)} \leqslant t - f_{e_a}\right]\frac{1}{w}$$

And then,

$$0 \leqslant F_{X_{e_a}^{(j)}}(t) - F_{X_{e_0}^{(j)}}(t) \leqslant \frac{1}{w}$$

Since $\mathbb{X}_{e_a} = min_{j=1}^{r}\{X_{e_a}^{(j)}\}$, we have $0 \leqslant F_{\mathbb{X}_{e_a}}(t) - F_{\mathbb{X}_{e_0}}(t) \leqslant \frac{r}{w}$. □

Theorem 3.2 enables us to use the ground-truth error bound for zero-sized flows as the ground-truth error bound for any specific flows. For simplicity, we denote $g_{e_0}(\delta)$ as $g(\delta)$. Our goal is to estimate $g(\delta)$.

Step 2: Our error bound for arbitrary flows is close to zero-sized flow's ground truth error bound. For one row of a sketch, the value of each counter can be viewed as a sample of the same random variable, denoted as Y. $Y = \sum_{e_i \in S} f_{e_i} X_i$, where $\Pr[X_i = 1] = \frac{1}{w}$ and $X_i = 0$ otherwise. From the definition of Y and $X_{e_0}^{(j)}$ (the error of zero-sized flow in one row), we find that they are exactly the same. Therefore, we can estimate the $g(\delta)$ from the distribution of the counter values. In one row of CM-Sketch, the w counter values can be viewed as w sample results. We then sort the w counter values, and pick the k-th largest counter $\hat{\xi}_k$, where $k = \lceil wp \rceil$ and $p = \delta^{\frac{1}{r}}$. Next, we try to prove that our estimation of $g(\delta)$ ($\hat{g}(\delta) = \hat{\xi}_{wp}$) is accurate.

In Lemma 3.3, we prove that, for any p-fractile counter value picked (*i.e.*, $\hat{\xi}_{wp}$), it is a close approximation to the ground-truth error bound of a zero-sized flow with confidence p.

Lemma 3.3. *For any probability p, the sampled sketch counter value $\hat{g}(\delta) = \hat{\xi}_{wp}$ falls in a close range of the CDF F_Y at value p.*

$$\Pr\left[p - \gamma < F_Y\left(\hat{g}(\delta)\right) < p + \gamma\right] \geqslant 1 - 2e^{-2w\cdot\gamma^2} \tag{3}$$

Proof. Let ξ_p be the (p)-th largest fractile or quantile of F_Y, $\xi_p = F_Y^{-1}(p) = \inf\{x | F(x) \geq p\}$.

$$\begin{aligned}
&\Pr\left[F_Y\left(\hat{\xi}_{wp}\right) \geqslant p + \gamma\right] \\
&= \Pr\left[\left(\sum_j I\left(X_j > \xi_{p+\gamma}\right)\right) > w(1-p)\right] \\
&= \Pr\left[\left(\sum_j I\left(X_j > \xi_{p+\gamma}\right)\right) - w\left(1-(p+\gamma)\right) > w\gamma\right]
\end{aligned}$$

Due to Hoeffding's inequality, variables X_j is strictly bounded by $[0, 1]$. Also $E\left[I\left(X_j > \xi_{p+\gamma}\right)\right] = 1 - (p + \gamma)$. Therefore we have

$$Pr\left[I\left(X_j > \xi_{p+\gamma}\right) - \left(1-(p+\gamma)\right) < \gamma\right] < e^{-\frac{2w^2\gamma^2}{w}} = e^{-2w\cdot\gamma^2}$$

Similarly, we have

$$Pr\left[I\left(X_j > \xi_{p-\gamma}\right) - \left(1-(p-\gamma)\right) > \gamma\right] < e^{-\frac{2w^2\gamma^2}{w}} = e^{-2w\gamma^2}$$

By Hoeffding bound, we prove the above theorem. □

From Lemma 3.3, we can estimate the bias of our bound:

Theorem 3.4. *Given the traffic stream $\mathcal{F}$ and the sketch configurations (i.e., w, and r). The bias of our estimated bound $\hat{g}(\delta)$ from $g(\delta)$ is small.*

$$\Pr\left[|\hat{g}(\delta) - g(\delta)| > g(\delta_l) - g(\delta_r)\right] \leqslant 2e^{-2w\gamma^2}, \forall\gamma > 0 \tag{4}$$

where $\delta_l = \left(\delta^{\frac{1}{r}} - \gamma\right)^r$, $\delta_r = \left(\delta^{\frac{1}{r}} + \gamma\right)^r$.

Let $\gamma = t(2w)^{-0.5}$. We have $1 - 2e^{-t^2}$ confidence that the bias of our estimation is no greater than $g(\delta_l) - g(\delta_r)$. δ_l and δ_r are both close to δ, and therefore $g(\delta_l) - g(\delta_r)$ is often a small value compared with the truth $g(\delta)$. Thus we derive our bound:

$$\hat{g}(\delta) = \hat{\xi}_{wp} \in \left[g\left(\delta\cdot\left(1-O\left(\frac{r}{\sqrt{w}}\right)\right)\right), g\left(\delta\cdot\left(1+O\left(\frac{r}{\sqrt{w}}\right)\right)\right)\right] \tag{5}$$

with high probability.

Summary: In Step 1, we prove in Theorem 3.1 that the ground-truth error bound of zero-sized flows is the larger than that of any arbitrary flow, making the error bound for zero-sized flows a feasible error bound for all flows. Then, Theorem 3.2 shows that the ground-truth error bound of zero-sized flows is a close approximation to any non-zero flow's error bound. Both theorems imply that, if we can derive an error bound that is close to the ground-truth error bound for zero-sized flows, that bound will be a close approximation error bound for arbitrary flows. Thus, in Step 2, we prove in Theorem 3.4 that the error bound we drive is a close approximation to the ground-truth error bound for zero-sized flows. With both steps, our error bound from Algorithm 1 shows its near-optimality.

3.3 Additional Analysis for the Original Bound

Our error bound is tighter than the original bound: Given a confidence of success $1 - \delta$, the original CM bound (which requires $r = \lceil ln(\frac{1}{\delta})\rceil$ and $w = \lceil\frac{e}{\epsilon}\rceil$ as in [8]) guarantees $\Pr\left[\mathbb{X}_{e_i} > \epsilon|F|_1\right] < \delta$, *where* $\epsilon = \frac{e}{w}, \delta = e^{-r}$. Our approach gives an error bound $\hat{g}(\delta) = \hat{\xi}_{wp}$ (recall that $p = \delta^{\frac{1}{r}}$) which satisfies $Pr\left[X_{e_i} > \hat{\xi}_{wp}\right] < \delta$. Now we prove that $\hat{\xi}_{wp} \leqslant \epsilon|F|_1$. Due to the properties of quantile, we have $\Sigma_{j=1}^{w}\hat{\xi}_j = |F|_1$. Also, $\hat{\xi}_1 \leqslant \hat{\xi}_2 \leqslant ... \leqslant \hat{\xi}w$. Thus, we have

$$\hat{\xi}_{wp} = \hat{\xi}_{w\left(1-\delta^{\frac{1}{r}}\right)} = \hat{\xi}_{w\left(1-(e^{-r})^{\frac{1}{r}}\right)} = \hat{\xi}_{1-\frac{1}{e}} \leqslant \frac{|F|_1}{w\times\frac{1}{e}} = \epsilon|F|_1 \tag{6}$$

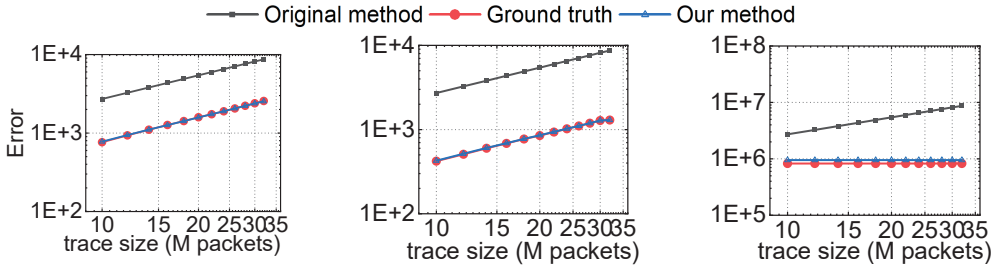
Therefore, the bound given by our method is always tighter than the original bound.

Analyzing the worst-case scenarios in the original bound: The original error bound of CM-Sketch is met if and only if the Markov inequality is reached. If we see the Markov inequality:

$$\begin{aligned}
E(X) &= \Sigma_0^w xP(X=x) \\
&= \Sigma_0^a xP(X=x) + \Sigma_a^w xP(X=x) \\
&\geqslant \Sigma_a^w xP(X=x) \geqslant \Sigma_a^w aP(X=x) \\
&= a\Sigma_a^w P(X=x) = aP\left(X \geqslant a\right)
\end{aligned}$$

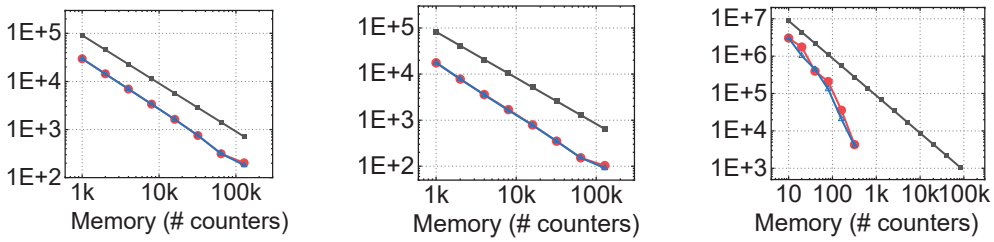
where a stands for $eE(X_{i,j})$ and X stands for $X_{i,j}$, which is the error of flow e_j in the i^{th} row. Reaching the equality requires the following two equations:

$$\Sigma_0^a xP(X=x) = 0$$



(a) CAIDA Trace (b) DC Trace (c) MACCDC Trace

Figure 3: CM-Sketch evaluation on different real-world trace sizes.



(a) CAIDA Trace (b) DC Trace (c) MACCDC Trace

Figure 4: CM-Sketch evaluation with different memory constraints.

$$\Sigma_a^w x P(X = x) = \Sigma_a^w a P(X = x)$$

This requires on the distribution of variable X: X shall be either 0 or a. For zero-sized flows, their error distribution X is the same as the counter value distribution, where each counter value is either 0 or a. Given that the sum of all counters is $|F|_1$ in any array of CM, there shall be $\left(1 - \delta^{\frac{1}{r}}\right) w$ counters with value 0 and $\delta^{\frac{1}{r}} w$ counters with value a in each array. The CM reaches the worst case when every its array reaches the worst case. This is the worst-case counter distribution constructed in the original error bound and is nearly impossible to achieve in practice due to hashing.

4 Evaluation

We evaluate our error estimator for Count-Min Sketch (with additional results for Count-Sketch and CU-Sketch deferred to Appendix B) on real-world traces (CAIDA [5], a data-center trace set [4], and MACCDC [21]) and synthetic traces [25]. Our experiments demonstrate that our estimator (1) provides significantly more accurate error bounds than the prior approach [8] on all three sketches, (2) matches the ground-truth error bound under all traffic patterns, and (3) shows potential benefits to real-world use cases.

4.1 Methodology

Traces: We use the five-tuple as the flow identity in all experiments. Evaluated traces include: (a) an anonymized packet trace from CAIDA [5] containing 33.6M packets and 7.7M flows; (b) a campus data center trace (DC) [4] containing 30.3M packets and 4.2M flows; (c) a DDoS attack trace from MACCDC [21] containing 32.28M packets and 195 flows; (d) 10 synthetic traces generated from Web-Polygraph [26], following the Zipf [25] distribution with skewnesses ranging from 0.3 to 3.0 and with a gap of 0.3. Each trace has 30M packets. The default sketch size is 4×1000 counters for CAIDA, DC, and the synthetic traces. The default sketch size for the DDoS attack trace is 4×10 counters, due to a smaller number

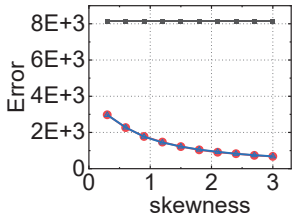


Figure 5: Error bounds under different skewnesses.

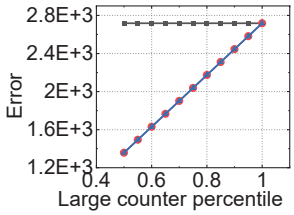


Figure 6: Finding "worst-case" workloads.

of flows. We use $\delta = e^{-r}$ as defined in [8], which is 0.018. We use $\epsilon = \frac{e}{w}$ as defined in [8], which is 0.0027.

Estimators under evaluation: *Original bound* refers to the error bound given by the original papers of CM-Sketch [8], Count-Sketch [6], and CU-Sketch [10]. *Ground truth* refers to the ground-true error bound. *Our method* refers to our error bound.

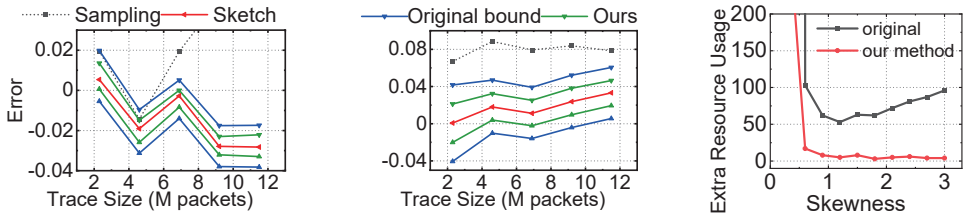
4.2 Evaluation of Error Estimator

We evaluate the aforementioned error bounds under 1) different trace sizes, 2) different memory constraints, and 3) datasets with different skewnesses.

Impact of trace sizes: Our method outperforms the original error bounds under varying trace sizes. Figure 3a, 3b, 3c show that the original error bounds have a 2000% to 4400% deviation from the ground truth for CM-Sketch, whereas our method has only a 0.8% to 14% deviation.

Impact of memory constraints: Figure 4a, 4b, 4c show that, under different memory constraints, the original method has a 9500% to 13000% deviation from the ground truth. In contrast, our method reduces the deviation to less than 40.4%.

Impact of trace skewness: Figure 5 demonstrates that our method outperforms the original error bound under different values of flow-size skewness. Our method has an error bound very close to the ground truth, with only 0.16% to 2.6% deviation for Count-Min



(a) Spine Cache Hit **(b) Leaf Cache Hit** **(c) Resource Usage**

Figure 7: Benefits of more precise error estimation.

Sketch. As the flow-size skewness raises from 0.3 to 3.0, the error reduction of our method, compared to the original bound, increases from 80× to 700×.

Figure 6 explores the worst-case scenarios in which the original error bound might work. We create these worst-case scenarios as follows: we use 1000 sketch counters, and we set the same value to $\delta^{\frac{1}{r}} w = 368$ counters (*i.e.*, "large" counters) and another value to all remaining 632 counters (*i.e.*, "small" counters). We set the counters such that the sum of the large counters over the total sum of counters varies from 100% to 50%. Figure 6 shows that the original bound equals the ground truth only when the sum of large counters equals the total counters (*i.e.*, small counters are zero-sized), which is arguably rare in real-world traffic.

4.3 Case Study: Distributed Caching

Count-Sketch can be used in caching to detect hot objects. Here we show the benefits of our estimator to improve cache performance in a particular distributed cache setting [17], where two layers of programmable switches (*e.g.*, spine and leaf switches in a Clos topology) serve as the cache nodes to store hot objects and balance the server loads.

Accurate cache hit-rate prediction: We feed the CAIDA traces (with different sizes) to Count-Sketch and NetFlow (with 1% sampling), and then reconstruct each trace using three methods (Count-Sketch with the original error bound or with our error bound, and 100× replication of the NetFlow-sampled packets). We then use the reconstructed traces to predict the range of hit rate *if* the CAIDA traces are fed to the aforementioned two-layer caches. The reconstruction and prediction methods are detailed in §B.1. Figure 7a and 7b show that our method (Count-Sketch-reconstructed traces with our proposed error bound) produces a much narrower estimation of the hit rate than Count-Sketch with the original (worst-case) error bound (*i.e.*, tighter gap between the green lines than between the blue lines). It is also more accurate than NetFlow-based sampling. Our gains over the baselines persist under varying trace sizes.

Better cache space provisioning: Given that the predicted cache hit rate will not be 100% accurate, operators must overprovision the cache space in order to ensure that a given number (by default, 100) of hottest objects are cached [11]. Here, we show that our method (Count-Sketch with the proposed error bound) can help operators overprovision less cache space. To this end, we assume that our goal is to cache the top-100 hottest objects, and that the flow size of the 100^{th} hottest object is f_{100}. We use Count-Sketch to estimate the flow sizes with an error bound E_r, and put all objects with estimated size greater than $f_{100} - E_r$ in the cache memory. Figure 7c shows that since our estimator of E_r is much smaller than the original error bound, it can use less overprovisioned cache space, while still caching the top-100 hottest objects.

5 A Future Roadmap

We have shown that sketch errors can be estimated much more accurately *a posteriori* (after the sketch output is known) than *a priori* (before the flows arrive). This result, together with the rigorousness of sketches, suggests an exciting direction to systematize a new approach to sketch-based analytics, which can benefit many network management tasks. Here, we outline a possible research agenda towards a *vision of highly accurate sketch-based analytics*.

Distributed sketch analytics with precise error estimation: Sketches show a viable path towards a resource-efficient, scalable analytics platform. The increasing demands on performance and reliability raise the bars for identifying and preventing failures in the network. The ability to precisely measure the errors of sketch results would facilitate adoption of sketches as a reliable data source; *e.g.*, L4 load balancers [13, 22] require precise estimation of large flows to rebalance the load, and traffic engineering [16, 27] requires precise estimation of dynamic link utilization. The opportunities and challenges lie in how to bound the errors of network-wide analytics results.

Enabling accurate "what-if" analysis: Sketch-based analytics could enable scalable, accurate "what-if" analysis: if we leverage sketches to process real-world workloads and reconstruct the original workloads with precise error estimates. This is because fully capturing the traffic (*e.g.*, at backbone WAN and an ISP ingress) is not infeasible but may be very expensive, and traditional packet sampling approaches (*e.g.*, NetFlow) are known to be inaccurate [9]. When designing sketch-based "what-if" analyzers, care must be taken to augment the sketch features as needed (*e.g.*, additional information about flow time-span and loss rate of the flows) and analyze the impact of errors.

Enabling reliable self-driving network control: Sketch-based analytics can facilitate autonomous control in diverse networked applications. When a decision-making process needs to query the analytics platform, it is critical to assess the confidence of the result and whether an refinement of the result is needed. Our precise error estimator is a promising first step towards this goal, with several open questions to be addressed in future work, such as dynamic sketch adjustment and on-demand error correction.

6 Acknowledgements

We would like to thank our shepherd Walter Willinger and the anonymous reviewers for their constructive feedback. This work was supported in part by NSF grants CNS-2107086 and CNS-2106946.

References

[1] Mohammad Alizadeh, Tom Edsall, Sarang Dharmapurikar, Ramanan Vaidyanathan, Kevin Chu, Andy Fingerhut, Vinh The Lam, Francis Matus, Rong Pan, Navindra Yadav, et al. 2014. CONGA: Distributed congestion-aware load balancing for datacenters. In *Proc. of ACM SIGCOMM*.

[2] Mohammad Alizadeh, Shuang Yang, Milad Sharif, Sachin Katti, Nick McKeown, Balaji Prabhakar, and Scott Shenker. 2013. PFabric: Minimal near-Optimal Datacenter Transport. In *Proc. of ACM SIGCOMM*.

[3] Ran Ben Basat, Gil Einziger, Roy Friedman, Marcelo Caggiani Luizelli, and Erez Waisbard. 2017. Constant Time Updates in Hierarchical Heavy Hitters. In *Proc. of ACM SIGCOMM and CoRR/1707.06778*.

[4] Theophilus Benson, Aditya Akella, and David A Maltz. 2010. Network traffic characteristics of data centers in the wild. In *Proc. of SIGCOMM IMC*.

[5] CAIDA. 2018. The CAIDA UCSD Anonymized Internet Traces equinix-chicago. http://www.caida.org/data/passive/passive_dataset.xml

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

[7] Graham Cormode, Flip Korn, S. Muthukrishnan, and Divesh Srivastava. 2008. Finding Hierarchical Heavy Hitters in Streaming Data. *ACM Trans. Knowl. Discov. Data* (2008).

[8] Graham Cormode and S. Muthukrishnan. 2005. An Improved Data Stream Summary: The Count-Min Sketch and Its Applications. *J. Algorithms* (2005).

[9] Cristian Estan and George Varghese. 2002. New directions in traffic measurement and accounting. In *Proc. of ACM SIGCOMM*.

[10] Cristian Estan and George Varghese. 2003. New directions in traffic measurement and accounting: Focusing on the elephants, ignoring the mice. *ACM Transactions on Computer Systems* (2003).

[11] Bin Fan, Hyeontaek Lim, David G Andersen, and Michael Kaminsky. 2011. Small cache, big effect: Provable load balancing for randomly partitioned cluster services. In *Proc. of SoCC*.

[12] Qun Huang, Patrick PC Lee, and Yungang Bao. 2018. SketchLearn: Relieving User Burdens in ApproximateMeasurement with Automated Statistical Inference. In *Proc. of ACM SIGCOMM*.

[13] Intel. [n. d.]. High Performance Layer-4 Load Balancer based on DPDK. https://github.com/iqiyi/dpvs.

[14] Xin Jin, Xiaozhou Li, Haoyu Zhang, Robert Soulé, Jeongkeun Lee, Nate Foster, Changhoon Kim, and Ion Stoica. 2017. NetCache: Balancing Key-Value Stores with Fast In-Network Caching. In *Proc. of ACM SOSP*.

[15] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. 2003. Sketch-based Change Detection: Methods, Evaluation, and Applications. In *Proc. of ACM IMC*.

[16] Hongqiang Harry Liu, Srikanth Kandula, Ratul Mahajan, Ming Zhang, and David Gelernter. 2014. Traffic engineering with forward fault correction. In *Proc. of ACM SIGCOMM*.

[17] Zaoxing Liu, Zhihao Bai, Zhenming Liu, Xiaozhou Li, Changhoon Kim, Vladimir Braverman, Xin Jin, and Ion Stoica. 2019. DistCache: Provable Load Balancing for Large-Scale Storage Systems with Distributed Caching. In *Proc. of USENIX FAST*.

[18] 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 *Proc. of ACM SIGCOMM*.

[19] Zaoxing Liu, Samson Zhou, Ori Rottenstreich, Vladimir Braverman, and Jennifer Rexford. 2019. Memory-Efficient Performance Monitoring on Programmable Switches with Lean Algorithms. *Proc. of SIAM/ACM APoCS* (2019).

[20] Yi Lu, Andrea Montanari, Balaji Prabhakar, Sarang Dharmapurikar, and Abdul Kabbani. 2008. Counter Braids: A Novel Counter Architecture for Per-Flow Measurement. In *Proc. of ACM SIGMETRICS*.

[21] MACCDC. 2012. Capture Traces from Mid-Atlantic CCDC. http://www.netresec.com/?page=MACCDC

[22] Rui Miao, Hongyi Zeng, Changhoon Kim, Jeongkeun Lee, and Minlan Yu. 2017. SilkRoad: Making Stateful Layer-4 Load Balancing Fast and Cheap Using Switching ASICs. In *Proc. of ACM SIGCOMM*.

[23] Masoud Moshref, Minlan Yu, Ramesh Govindan, and Amin Vahdat. 2015. Scream: Sketch resource allocation for software-defined measurement. In *Proc. of ACM CoNEXT*.

[24] Chen Peiqing, Chen Dong, Zheng Lingxiao, Li Jizhou, and Yang Tong. 2021. Out of Many We are One: Measuring Item Batch with Clock-Sketch. In *Proceedings of the 2021 International Conference on Management of Data* (Virtual Event, China) *(SIGMOD '21)*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3448016.3452784

[25] David MW Powers. 1998. Applications and explanations of Zipf's law. In *New methods in language processing and computational natural language learning*.

[26] Alex Rousskov and Duane Wessels. 2004. High-performance benchmarking with Web Polygraph. *Software: Practice and Experience* (2004).

[27] Rachee Singh, Manya Ghobadi, Klaus-Tycho Foerster, Mark Filer, and Phillipa Gill. 2018. RADWAN: rate adaptive wide area network. In *Proc. of ACM SIGCOMM*.

[28] Cha Hwan Song, Pravein Govindan Kannan, Bryan Kian Hsiang Low, and Mun Choon Chan. 2020. FCM-sketch: generic network measurements with data plane support. In *Proc. of CoNEXT*.

[29] Daniel Ting. 2018. Count-min: Optimal estimation and tight error bounds using empirical error distributions. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2319–2328.

[30] 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 ACM SIGCOMM*.

[31] Yinda Zhang, Zaoxing Liu, Ruixin Wang, Tong Yang, Jizhou Li, Ruijie Miao, Peng Liu, Ruwen Zhang, and Junchen Jiang. 2021. CocoSketch: High-Performance Sketch-based Measurement over Arbitrary Partial Key Query. In *Proc. of ACM SIGCOMM*.

A Detailed Proofs for Section 3.2

Background on CU-Sketch and Count-Sketch: In addition to CM-Sketch, CU and CS are two other widely-used sketches. Our error estimator can also perform well on them after minor updates. Specifically, CU and CS share the same $r \times w$ counter structures with CM but have different insertion and query strategies. To insert flow e, CU only increments the smallest counter(s) among the r rows while CM increments all r corresponding counters. CS needs to update all r counters, the same as CM. But instead of always increment the counters, CS updates the counters by $S(e)$, where $S(\cdot)$ is a hash function that outputs $\{+1, -1\}$ with equal probability. When queried with the r hash keys of a flow identity, CS reports the *median* value of the r corresponding counters (whereas CM and CU reports the *minimum* value).

A.1 Algorithm details for CU-Sketch and Count-Sketch

We first show the pseudo code of error estimator for CU-Sketch in Algorithm 2 and for Count-Sketch in Algorithm 3. Notice that the algorithm for CU-Sketch is exactly the same as that for CM-Sketch due to their similar properties in generating sketch: they share a common query paradigm that returns the minimum counter value among the r rows as the estimated flow size.

For Count Sketch: In CS, the flow size is estimated by the median value of among the corresponding counters. Let $g(\delta)$ be the ground truth error bound. In Algorithm 2, let p_0 be the probability that, for one row, the absolute error of the corresponding counter is no less than $g(\delta)$. For CS, the relationship between p_0 and δ is more complex than that of CM/CU. The probability that the median value is not bounded by $g(\delta)$ is $2 \times \sum_{j=\frac{r+1}{2}}^{r} \binom{r}{j} \frac{p_0}{2}^j (1 - \frac{p_0}{2})^{r-j} = \delta$ when r is an odd number[3] (Proof in Section 3.2). We find the value of p_0 by enumeration for easy understanding, which can also be solved by faster bisection approach. Then, we sort $A[1][1] \ldots A[1][w]$ by non-ascending order based on their absolute values and report the $\lfloor wp_0 \rfloor$-th largest counter as our estimation result.

Algorithm 2: Error Estimator for CU-Sketch.

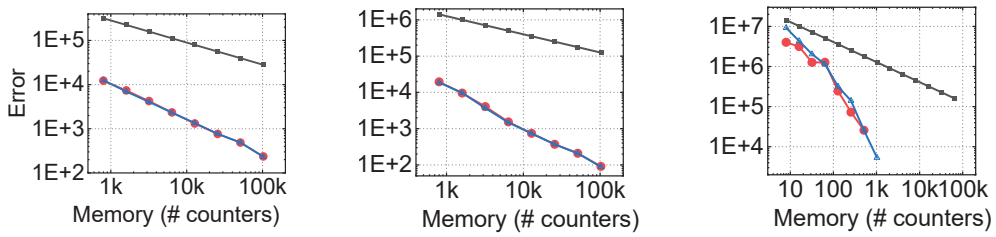
Input: $A[1 \ldots r][1 \ldots w]$ with confidence $1 - \delta$.

1 $p_0 \leftarrow \delta^{\frac{1}{r}}$;

2 SortToDescendingOrder($A[1][1] \ldots A[1][w]$);

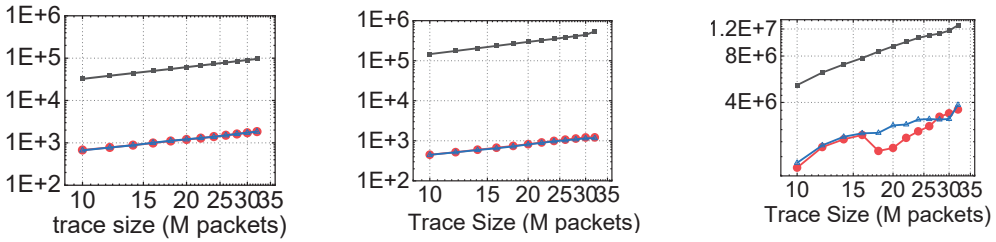
Output: $A[1][\lceil wp_0 \rceil]$

[3] Similar equation for an even r.



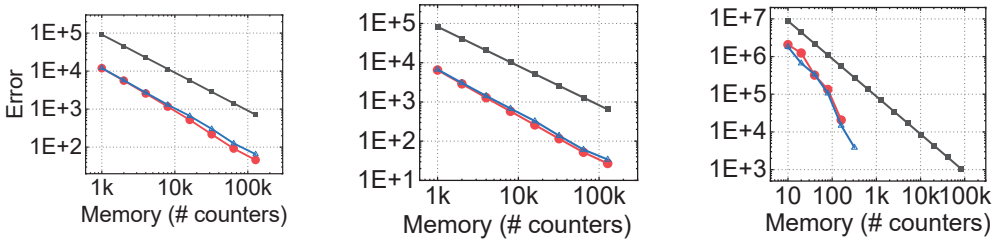
(a) CAIDA Trace (b) DC Trace (c) MACCDC Trace

Figure 8: Count-Sketch evaluation on different real-world trace sizes.



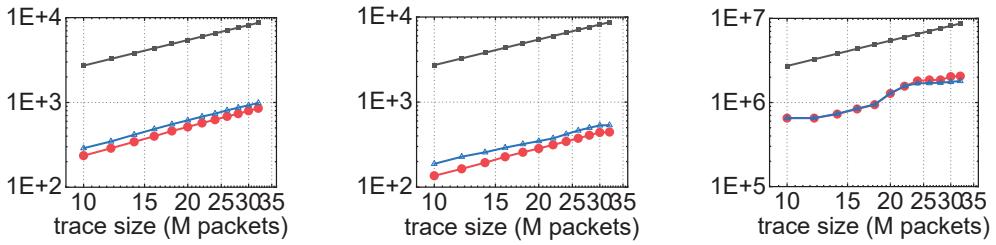
(a) CAIDA Trace (b) DC Trace (c) MACCDC Trace

Figure 9: Count-Sketch evaluation with different memory constraints.



(a) CAIDA Trace (b) DC Trace (c) MACCDC Trace

Figure 10: CU-Sketch evaluation on different real-world trace sizes.



(a) CAIDA Trace (b) DC Trace (c) MACCDC Trace

Figure 11: CU-Sketch evaluation with different memory constraints.

A.2 Our bound for Count-Sketch is tighter

CS has the similar property when giving a better error bound as CM.

Theorem A.1. *For Count-Sketch with $r = 2k + 1, k \in \mathbb{Z}$, the optimal error bound $g(\delta)$ is ξ_p, where p satisfies equation $\sum_{j=k+1}^{r} \binom{r}{j} p^j (1-p)^{r-j} = \frac{\delta}{2}$.*

Proof. Because our algorithm sorts all the absolute counter values, it is slightly different from the formula used here. Suppose the error of the r-th row is $X_r := A[r][h_r(e_i)] - f_i$. When $f_i = 0$, we calculate error bound $g(\delta)$:

$$g(\delta) := \Pr[|median_r(X_r)| > g(\delta)] = \delta, \forall \delta$$

As X_r are distributed symmetrically about 0, we have

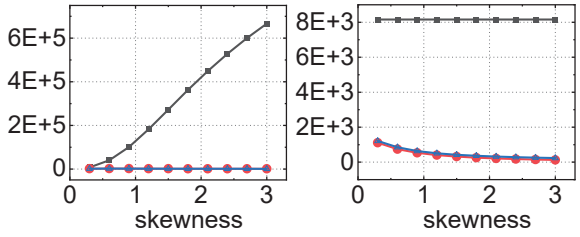
$$\Pr[median_r(X_r) < g(\delta)] = \frac{\delta}{2}$$

Algorithm 3: Error Estimation for Count Sketch.

Input: $A[1\dots r][1\dots w]$ with confidence $1-\delta$.

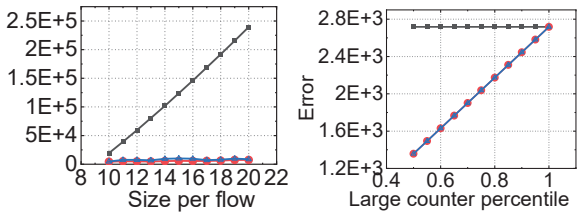
```
for i = 1 → w do
    p_0 ← (i+1)/w;
    if 2 · Σ_{j=(r+1)/2}^{r} C(r,j) (p_0/2)^j (1 − p_0/2)^{r−j} ≥ δ then
        Break;
SortToDescendingOrder(|A[1][1]|, ..., |A[1][w]|);
```

Output: $|A[1][\lceil wp_0 \rceil]|$



(a) Count-Sketch **(b) CU-Sketch**

Figure 12: Error bounds under different skewnesses for Count-Sketch and CU-Sketch.



(a) Count-Sketch **(b) CU-Sketch**

Figure 13: Finding "worst-case" workloads on Count-Sketch and CU-Sketch.

Then

$$\Pr[\sum_{r=1}^{r} I(X_r < g(\delta)) \geqslant k+1] = \frac{\delta}{2}$$

$$\text{let } p = I(X_r < g(\delta))$$

$$\sum_{j=k+1}^{r} \binom{r}{j} p^j (1-p)^{r-j} = \frac{\delta}{2}$$

□

B Evaluation on Count-Sketch and CU-Sketch

Impact of trace sizes: Our method outperforms the original error bounds under varying trace sizes on all three sketches. Figure 8a, 8b, 8c (Count-Sketch), and Figure 10a, 10b, and 10c (CU-Sketch) show that our method achieves an error bound with less than 0.7% deviation for Count-Sketch, and less than 41% deviation for CU-Sketch.

Impact of memory constraints: Count-Sketch and CU-Sketch using our approach have tighter bounds than the original error bounds under different memory constraints too. Figure 9a, 9b, 9c (Count-Sketch), and Figure 11a, 11b, 11c (CU-Sketch) show that our method improves the error bounds with a less than 40.4% deviation for Count-Sketch, and a less than 20.7% deviation for CU-Sketch.

Impact of trace skewness: Figure 12 demonstrates that our method outperforms the original error bound for Count-Sketch and CU-Sketch when skewness varies. Our method gets an error bound extremely close to the ground truth (0.7% to 5.6% deviation for Count-Sketch and < 9.3% deviation for CU-Sketch). Besides, as skewness grows, our method outperforms the original method more. Count-Min Sketch and CU-Sketch has a constant error bound because it only depends on the sum of all flow sizes. Count-Sketch has an error bound that grows with trace skewness. When using trace skewness from 0.3 to 3.0, our method's error bounds are 80 to 700 times tighter on tested sketches.

Finding worst-case scenarios: In Figure 13, we explore the scenarios in which the original error bounds may work. In CU-Sketch (Figure 13b), the setting is exactly the same as CM-Sketch because these two sketches have the same error bound. The major difference is that, it is even more difficult for CU-Sketch to reach such a counter value distribution. Because it only adds to the smallest counter when inserting a packet, the sum of counter values in one row may be smaller than $|F|_1$. Therefore, the largest possible value of the $\delta^{\frac{1}{r}}$-fractile shall be even smaller. For Count-Sketch, we approach the worst case by setting the total size of flows as 1M and the size of each flow equally (from 10 to 20). Results in Figure 13a show that a "better" scenario for the original bound is when all flow sizes are 10, which is difficult to achieve in real-world traffic.

B.1 Trace Recovery Methods in Case Study

Here we demonstrate how we use uniform sampling and sketching to perform statistics on the original trace and later restore the trace.

Uniform Sampling of sample rate $\frac{1}{100}$ records each packet in the data stream with probability $\frac{1}{100}$. When restoring the trace, it enlarges the sampled trace 100 times (*i.e.*, duplicating each packet by 100 times). There are two major limitations of uniform sampling method: (1) it has large estimation error on the size of large flows; (2) it cannot provide an error bound for flow sizes.

Sketching is a better method which can provide error bounds. We use Count-Sketch(5 rows of 12000 counters each) to measure the flow sizes and store the top-1600 flow ID. For flow size restoration, we first restore the size of Top-K flows and give two error bounds based on the original error bound and our method. After the restoration of Top-Kflow sizes, we calculate the remaining trace size and uniformly allocate them to all small flows.

Prediction method: In the "Accurate cache hit-rate prediction" evaluation, we feed DistCache with the reconstructed traces. In this setting, we suppose that all switches can have the accurate hot objects stored on the switch cache, which is the ideal case. After that, we estimate the number of cache hits generated on two layers of cache nodes and compare them with the ground truth (*i.e.*, cache hits generated by the original trace).