AlignSketch: A Framework for Aligning Theoretical and Practical Estimation Errors

Ce Zheng*, Hanyue Zheng†, Jingwei Shi‡, Xinye Xu†, Wei Zhou§, Tong Yang†, Zhenyu Guan*, and Yong Cui¶
*Beihang University, China, †Peking University, China, ‡Shanghai University of Finance and Economics, China,
§University of Southern California, United States, ¶Tsinghua University, China

***Abstract*—Sketches are widely used in data streams and approximate query processing for their low memory and acceptable errors. They can be categorized into non-classification-based and classification-based sketches. While the latter typically provide higher accuracy, they often involve fragmented stream processing, which increases the complexity of theoretical analysis and leads to a widened gap between theoretical bounds and practical estimation errors. To address the challenge, we propose AlignSketch, a general classification framework for data stream processing with provable theoretical guarantees. It achieves tight and verifiable alignment between theoretical error bounds and practical estimation errors. The framework ensures each stream item is recorded at only one tier at any given time, and that estimation errors do not accumulate across tiers but originate from a single one. An important strength of our design is its ability to leverage theoretical results on LRU and LFU miss rates, which have not been explored in existing sketches, to derive tight error bounds. Extensive experiments show that AlignSketch achieves closer alignment between theoretical and practical estimation errors compared to state-of-the-art methods while reducing practical estimation errors by nearly 1 to 2 orders of magnitude. The source code is available on GitHub[1].**

I. Introduction

A. Background and Motivation

Sketches [1]–[20] are compact and efficient approximate algorithms, widely used in data stream and approximate query processing due to low memory usage and acceptable errors. These properties make them particularly suitable for real-time analytics and system tuning on resource-constrained platforms such as caches and chips [21]–[30]. In practical deployments, it is often necessary to pre-determine the required resources under given error constraints. For example, an operator may need to determine *how much memory is needed to keep the relative error within* 10%. Experimental results alone are insufficient for making such decisions. This calls for sketches whose theoretical guarantees align with their practical behavior, beyond simply providing loose theoretical error bounds.

Existing sketches can be categorized into non-classification-based sketches and classification-based sketches (see Fig. 1). Non-classification-based sketches [11], [18], [31]–[34] use one simple structure for all items, which makes theoretical error bounds easy to derive. However, their theoretical and practical results also diverge. Recent work by Miao *et al.* [35] and Chen *et al.* [36] optimized the practical results for better alignment. Classification-based sketches [3], [10], [37]–[40], in contrast, classify items by frequency using more sophisticated designs. They offer higher accuracy and can reduce errors by one to two orders of magnitude (see § II-A), but deriving tight theoretical bounds is difficult. Achieving close alignment between theory and practice remains a key challenge.

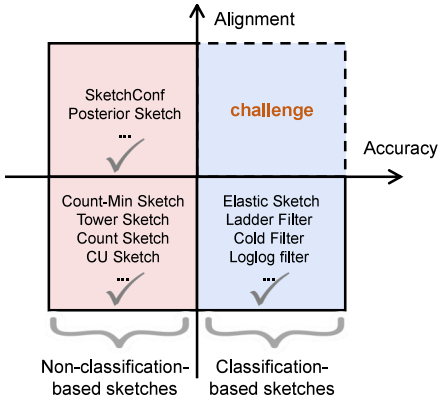


Fig. 1: Classification of Existing Sketch Algorithms.

Our insight is that the root cause of this issue lies in how existing classification strategies handle data streams: a single stream is fragmented and stored across multiple tiers. Such fragmentation introduces cumulative errors across different tiers and, more importantly, uncertainty. This uncertainty stems from the non-negligible and hard-to-predict probability that each tier may encounter its own worst-case scenario at any given time. As a result, theoretical analyses of classification-based sketches often rely on worst-case assumptions across all tiers or numerous idealized conditions, leading to relatively loose error bounds. For example, on the CAIDA dataset, a realistic real-world workload with bursty traffic, Elastic [37] reports a theoretical error bound of about 65% under a given memory budget, whereas the practical error is around 5% (see

Ce Zheng and Zhenyu Guan are with the School of Cyber Science and Technology, Beihang University, Beijing, China.
Hanyue Zheng, Xinye Xu, and Tong Yang are with the School of Computer Science, Peking University, Beijing, China.
Jingwei Shi is with the School of Information Management & Engineering, Shanghai University of Finance and Economics, Shanghai, China.
Wei Zhou is with the Viterbi School of Engineering, University of Southern California, Los Angeles, California, United States.
Yong Cui is with the Department of Computer Science and Technology, Tsinghua University, Beijing, China.
Corresponding author: Tong Yang (yangtong@pku.edu.cn).
[1]https://anonymous.4open.science/r/AlignSketch

Fig. 8(b)). Some other classification-based sketches [38], [39] rely solely on empirical design and have yet to provide the overall error bounds explicitly. This imprecision in theoretical analysis further hinders the validation of alignment between theoretical derivations and practical results. To address this issue, we aim to design a classification-based framework for aligning theoretical and practical estimation errors.

B. Our Solution

We propose AlignSketch, a general classification framework for data stream processing with provable theoretical guarantees. It establishes tight and verifiable alignment between theoretical error bounds and practical estimation errors.

To achieve the goal, the framework is designed to avoid fragmented recording across multiple tiers. We introduce threshold-free Time Buffer Observation and Error Correction Transition mechanisms for stream items (§ IV-B) as the bidirectional transition to ensure that each item resides in exactly one tier at any given time. Meanwhile, frequent items are managed using LRU (Least Recently Used) and LFU (Least Frequently Used) as the frequent tier, while infrequent items are handled by a simple sketch with theoretical error bounds, even if loose (§ IV-A). As a result, since LRU and LFU incur no estimation error, all estimation errors in the entire framework originate solely from the Infrequent tier, *i.e.,* a single, well-controlled error source. Building on the above design, we further incorporate theoretical results on LRU and LFU miss rates from cache management (Eqs. 2-3), which, to the best of our knowledge, have not been explored in prior sketch-based research. These results enable principled error control and facilitate the optimization of sketch error bounds, allowing us to derive tight and verifiable theoretical guarantees through rigorous analysis (§ V-B).

Furthermore, we instantiate the framework with two concrete algorithms, namely AlignSketch(Tower) and AlignSketch(CM), which employ the Tower and the Count-Min sketch as the infrequent tier, respectively. Theoretically, we rigorously prove they achieve optimized error bounds compared to the original sketches, with an improvement factor of $\frac{1}{(\phi P_{miss})^3}$, where $\phi P_{miss} \ll 1$ (Table II). Experimentally, both methods outperform existing classification-based sketches in theoretical-practical alignment on four datasets following different distributions (§ VI-B), with AlignSketch(Tower) demonstrating the best performance. Further comparisons show that, while maintaining the alignment, our method reduces estimation errors by 1 to 2 orders of magnitude, significantly surpassing current state-of-the-art (SOTA) approaches (§ VI-D).

Our main contribution can be summarized as follows.
(1) We propose AlignSketch, a general sketch framework, which lays a theoretical foundation for achieving alignment between theory and practice in classification-based sketches.
(2) Building on this framework, we develop two algorithmic instances and rigorously prove that they achieve optimized theoretical error bounds, further confirming their tight alignment with the practical estimation errors.
(3) Extensive experiments show that our method significantly outperforms existing SOTA methods in accuracy, achieving a 1 to 2 orders of magnitude reduction in estimation error. The source code is available on GitHub [41].

The remainder of this paper is organized as follows: Section II summarizes related work. Section III introduces preliminaries. Section IV presents the proposed framework. Section V provides theoretical analysis. Section VI reports experimental results. Section VII discusses practical considerations, and Section VIII concludes the work.

II. Related Work

This section presents the current state of research in two primary directions of sketch algorithms: enhancing estimation accuracy under memory constraints (§II-A), and reducing the gap between theoretical guarantees and practical estimation error (§II-B). These two directions correspond to the horizontal and vertical axes of Fig. 1, respectively.

A. Enhancing Estimation Accuracy

Common non-classification-based sketches like the Count-Min(CM) sketch [18], [31], the Tower sketch [7], the Space-Saving(SS) [34], Unbiased SpaceSaving(USS) [11], the CU sketch [33], the Count sketch [32], *etc.* treat frequent and infrequent items uniformly, leading to significant errors when conflicts occur between them. As illustrated in Fig. 2, conflicts between frequent and infrequent items in the CM sketch could increase estimation error by one to two orders of magnitude compared to conflict-free scenarios. This suggests that incorporating classification mechanisms into sketches could potentially reduce estimation error by a similar margin compared to non-classification-based counterparts.

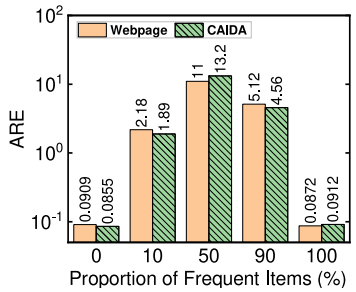


Fig. 2: Estimation error variation in CM sketch under hash conflicts between frequent and infrequent items on two real datasets [42], [43] (200KB memory, 10K tracked items).

Representative examples of classification-based sketches include the Elastic sketch [37], the Ladder Filter [38], the Cold Filter [39], the Loglog Filter [3], and the Heavyguardian [40]. These sketches assign frequent and infrequent items to different tiers for processing. Since the frequency of an item may vary over time, classifying based solely on its current frequency may result in the same item being recorded in different tiers at different times[2]. These sketches must traverse

[2]Discarding partial frequency of one stream item is also considered a form of fragmented recording across different locations.

TABLE I: Key differences: this work *vs.* traditional classification-based sketches

Method	**Ladder Filter** [38]	**Cold Filter** [39]	**Elastic Sketch** [37]	**This work**
High Estimation Accuracy	✓	✓	✓	✓
Comparable Throughput	✓	✓	✓	✓
Comprehensive Theoretical Error-Bound	–	–	✓	✓
Verifiable Theory-Practice Consistency	–	–	✓	✓
Tight Theory-Practice Alignment	–	–	–	✓

multiple tiers when querying, resulting in accumulated errors from each tier. Alternatively, retaining only the records from the most recent tiers and discarding frequency information from the remaining tiers may also increase estimation errors.

B. Reducing the Theory-Practice Gap

Recent research on sketches has made progress in reducing the gap between theoretical and practical estimation errors. Most existing approaches target non-classification-based sketches, such as CM sketches [18], [31], Nitro sketches [27], Count sketches [32], and [44], aiming to align experimental results with established theoretical bounds. Representative works include Miao *et al.* [35] and Chen *et al.* [36]. The former optimizes algorithmic practical performance to approach existing theoretical bounds but does not improve the bounds themselves. The latter employs posterior estimation, which is not applicable in predictive contexts. Meanwhile, these approaches generally treat data streams as homogeneous, without classification. In contrast, classification-based sketches yield estimation errors that may be one to two orders of magnitude lower (details refer to Fig. 2). These observations suggest that both the theoretical guarantees and practical performance of these methods can be further improved.

For classification-based sketches, a representative traditional work is Elastic sketch [37], which is discussed in § I-A. It provides a comprehensive theoretical error bound for the overall framework, but the bound remains relatively loose. Another notable study is LadderFilter [38], which empirically sets a frequency threshold for each LRU to discard items that have not appeared recently, enabling stream classification and achieving good accuracy. Nevertheless, it does not derive explicit error bounds, and its theoretical framework could still benefit from further refinement. Similarly, Cold Filter [39] does not provide strict theoretical guarantees. Table I summarizes the key differences between this work and traditional classification-based sketches. Additionally, sketch optimization approaches based on machine learning have been explored, such as Hsu *et al.* [45], which improves the accuracy of CM and Count sketches using external pre-trained models. However, due to the limited interpretability of many machine learning methods, it remains challenging to rigorously analyze the consistency between their theoretical guarantees and practical behavior.

Overall, none of the above solutions can achieve alignment, which is a key focus of this work.

III. Preliminary

This section first presents an overview of sketch algorithms and the common problems they address (§ III-A), then introduces basic definitions related to this study (§ III-B).

A. Sketch Algorithms

Sketches [1]–[20] are probabilistic data structures that use hash functions to map high-dimensional input into compact vectors or matrices, enabling efficient summarization of large-scale, high-velocity data streams [14], [46]–[48]. This compactness makes them particularly suitable for data stream and approximate query processing. In particular, in resource-constrained environments that impose strict limits on memory and computation, exact methods are often impractical. Sketch-based approaches provide acceptable-error estimates using sublinear space, offering a practical solution for real-time processing under stringent resource constraints [21]–[24].

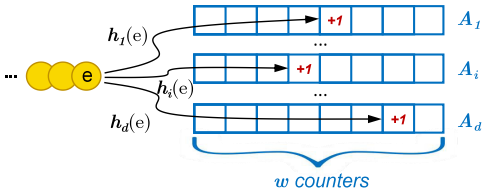


Fig. 3: Example of the CM Sketch.

As shown in Fig. 3, the CM sketch [18], an example of the most basic sketch, consists of d hash arrays (typically d=3), where each hash array A_i, contains w counters and is associated with a hash function $h_i(\cdot)$. Taking frequency query as an example, when a new item e arrives, the counter $A_i[h_i(e)\%w]$ (for $i \in \{1, 2, \ldots, d\}$) is incremented by 1. Due to hash collisions, false positives may occur, introducing estimation errors that result in overestimated outcomes. Therefore, when querying the frequency of e, the CM sketch returns the minimum among all d counters mapped by the hash functions to mitigate errors caused by overestimation. They can also capture other stream statistics, such as inter-arrival times, stream lengths, and byte volumes.

B. Basic Definitions

Data Stream: The data stream $\mathcal{S}$ is defined as a sequence $\{e_1, e_2, \cdots, e_i, \cdots\}$, where each item e_i belongs to an ID set $\{1, 2, \cdots, m\}$. Items within $\mathcal{S}$ may have duplicates; however, each item corresponds exclusively to one ID.

Approximate Query Processing Tasks: In data streaming, approximate query processing commonly involves tasks such as frequency estimation [3], [11], [12], [18], [49], top-K queries [3], [34], [50], [51], heavy hitter detection [16], [27], [40], [52], entropy estimation [53], [54], and change detection [55], [56]. Among them, the most widely used are frequency estimation and top-K queries. Frequency estimation aims to query the number of times a specific item e_i appears in the data streams $\mathcal{S}$, denoted as $\hat{f}_i$. Top-K queries retrieve

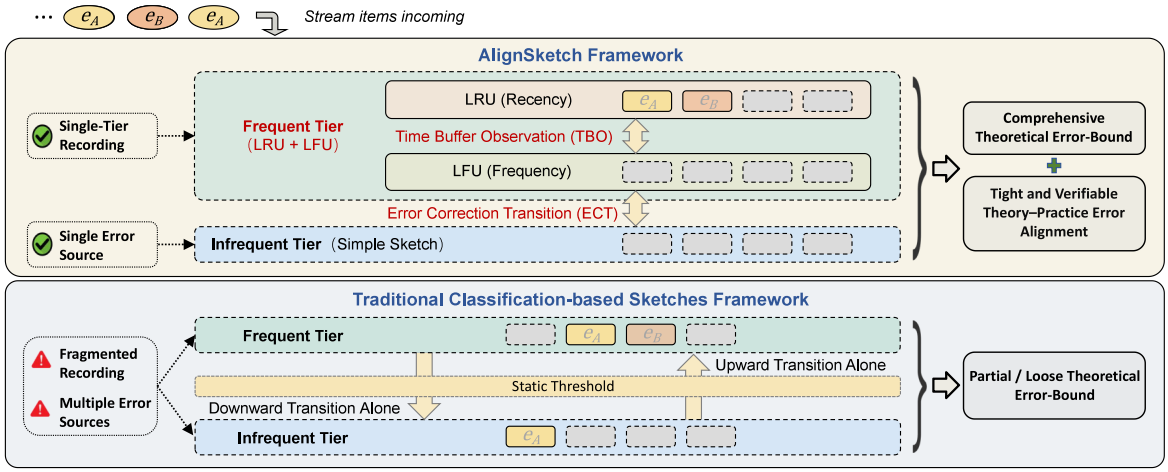


Fig. 4: Comparison between traditional classification-based sketches and AlignSketch.

the K most frequent items in $\mathcal{S}$ and are commonly used in tasks like anomaly detection and trend monitoring.

In the context of approximate query processing for stream item frequency, we introduce the following two key definitions. *Estimation Error:* Let f_i denote the true frequency (*i.e.*, ground truth) of an item e_i, and let $\hat{f}_i$ be the estimated frequency returned upon querying the data stream processing algorithm. The estimation error is defined as the absolute difference $|\hat{f}_i - f_i|$, serving as a direct measure of accuracy for algorithms such as sketches.

Error Bound: Given an arbitrarily small positive number ϵ, the error bound guarantees that the absolute difference between the estimated frequency $\hat{f}_i$ of item e_i and its true frequency f_i does not exceed ϵn with probability at least $1-\delta$, where n is the total number of items and $1-\delta$ is desired to be as close to 1 as possible. Formally, $\Pr\left(|\hat{f}_i - f_i| \leq \epsilon n\right) \geq 1-\delta$.

IV. The AlignSketch Design

In this section, we propose a classification-based multi-tier sketch framework to prevent fragmented recording of stream items across multiple tiers. This design improves the overall estimation accuracy of the algorithm while supporting subsequent theoretical analysis. We first present the overall framework architecture and its design rationale (§ IV-A). We then introduce two transition mechanisms that govern the tier-aware handling of stream items within the framework (§ IV-B). Finally, we discuss the data structures adopted at each tier and their associated time complexities (§ IV-C).

A. Data Stream Classification Framework

We observe that classification-based sketches often achieve higher estimation accuracy than non-classification-based approaches in § II-A. The key reason is that separating stream items by frequency can significantly reduce conflicts between frequent and infrequent items. Motivated by this observation, we refer to the traditional multi-tier structure that classifies stream items according to their access behaviors and records them at different tiers (Fig. 4).

However, in data stream scenarios where items arrive sequentially, classifying items solely based on their instantaneous frequency at arrival introduces a key challenge, namely, fragmented recordings across multiple tiers. As a result, the same item may be recorded at different tiers over time, leading to error accumulation during query processing and limiting further accuracy improvements. More importantly, sketch algorithms with such designs typically admit only relatively loose theoretical error bounds, since their analysis often relies on worst-case assumptions in which multiple tiers simultaneously encounter their worst-case scenarios, whose occurrence probabilities are difficult to characterize.

To address these issues, our core design principle is to enable threshold-free bidirectional transitions across tiers, while ensuring that each item is recorded in exactly one tier at any given time. By adaptively adjusting item tier assignment based on access behaviors, the estimation error is strictly confined to a specific tier, thereby avoiding simultaneous worst-case scenarios across multiple tiers. This property not only improves practical accuracy but also provides a foundation for establishing verifiable and tighter theoretical error bounds.

As illustrated in Fig. 4, we design a classification framework termed AlignSketch based on the above principles. The framework consists of three tiers: an LRU layer, an LFU layer, and a simple sketch tier for handling infrequent items. The LRU and LFU layers jointly form the frequent tier, while infrequent items are handled by the bottom tier using a sketch with theoretical error bounds. In the frequent tier, we adopt a combination of LRU (Least Recently Used) [57], [58] and LFU (Least Frequently Used) [59], [60]. Using LRU alone may cause cache pollution under scan-like or noisy access patterns due to its insensitivity to frequency, while using LFU alone may prematurely evict newly emerging frequent items during workload shifts or cold-start phases because it ignores temporal information. By leveraging LRU to protect recency and LFU to identify frequency, their complementary properties effectively reduce misclassification in the frequent tier, thereby suppressing the propagation of estimation errors caused by misclassification and providing a necessary basis

for subsequent miss-rate-based theoretical analysis (§ V-B).

In AlignSketch, arriving items are inserted into the multi-tier structure according to the transition mechanisms. Under capacity constraints, items may migrate across tiers. Based on the access statistics maintained by these mechanisms (*e.g.,* frequency), items in higher tiers may be evicted to lower tiers, while items in lower tiers may also be promoted to higher tiers. The detailed mechanisms are provided in § IV-B. For the infrequent tier, the framework allows flexible substitution of sketch implementations, provided that they possess provable error bounds. In this paper, we adopt the recently proposed Tower sketch [7] and the classical CM sketch [18], [31] as two representative instances[3], corresponding to AlignSketch(Tower) and AlignSketch(CM), with theoretical and experimental results shown in § V and § VI, respectively.

B. Transition Mechanisms for Stream Items

In this subsection, we present the transition mechanisms that govern how stream items are handled across different tiers in the framework, aiming to improve estimation accuracy in practice. These mechanisms are designed based on a core principle: at any time, each stream item is recorded by exactly one tier, thereby avoiding fragmented recordings and preventing cross-tier error accumulation.

1) Tier-Aware Transition Logic: As stream arrivals are unpredictable, transition decisions are made dynamically based on the observed behavior of each arriving item and the capacity status of different tiers. When a stream item arrives, its transition behavior is determined by two factors: (i) whether the item hits an existing entry in the frequent tier, and (ii) whether the frequent tier still has available capacity.

Algorithm 1: General Insertion Procedure

Input: Incoming item e
Data: frequent tier (LRU LFU), infrequent tier (sketch)
if $e \in$ *frequent tier* ***or*** *frequent tier is not full* **then**
  Alg. 2 (TBO-based Frequent-Tier Insertion)(e);
else
  Alg. 3 (ECT-based Infrequent-Tier Insertion)(e);

AlignSketch follows a general insertion procedure (Alg. 1) that dispatches an incoming item to either the frequent-tier or infrequent-tier handling logic. If the arriving item hits either the LRU or LFU layer, its update and possible transition are handled within the frequent tier by the Time Buffer Observation (TBO) (§ IV-B2). If the item misses the frequent tier but capacity remains available, it is inserted into the LRU layer as a new candidate and continues to be governed by the TBO-based frequent-tier mechanisms. Only when the item misses both layers of the frequent tier, and the frequent tier is already full, is it treated as an infrequent candidate and inserted into the sketch layer, where the Error Correction Transition (ECT) (§ IV-B3) is applied.

[3]Compared to the CM sketch (Fig. 3), the Tower sketch uses counters of different bit widths across layers of the hash array.

2) Frequent-Tier Transition via TBO: To avoid prematurely demoting potentially frequent items[4], we treat the LRU layer as a short-term buffer that provides an observation window for newly arrived or recently accessed items.

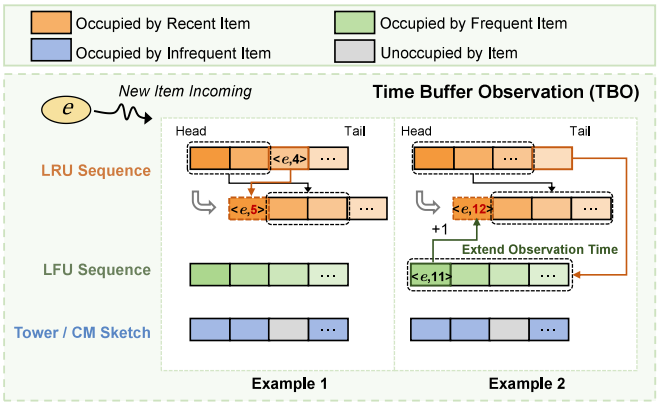


Fig. 5: Time Buffer Observation Example. In the LRU, the darker the color of the item, the more recent its appearance; in the LFU, the darker the color, the greater the frequency of the item.

As illustrated in Fig. 5, the LRU sequence serves as a time buffer that observes incoming items before classification. This observation window enables the transition mechanism to distinguish transient accesses from truly frequent items. Without such protection, potentially frequent items may transiently pass through the sketch layer (*i.e.*, Tower or CM) without their IDs being recorded, absorbing frequency contributions from other items and amplifying estimation errors. By delaying demotion decisions during the cold-start phase, the mechanism prevents such items from being prematurely inserted into the sketch, which is intended for infrequent items. The idea of extending the observation time is conceptually similar to classical buffer management policies such as 2Q and LRU-k [61], [62]. However, unlike these cache-oriented designs, we adopt a simplified transition structure tailored to sketch-based stream processing to control estimation error propagation caused by transient passage through ID-free sketch layers.

Frequent-Tier Insertion (Alg. 2): When an item e arrives,
(1) If item e is recorded in the LRU, its frequency f is incremented by 1. It is moved to the head of the LRU, with others moving toward the tail (shown in Example 1 of Fig. 5).
(2) If the LRU is full and item e is not recorded, but it is recorded in the LFU, e transitions upwards to the head of the LRU, with items moving toward the tail. In this case, the tail item e' in LRU undergoes a downward transition. LFU has at least one empty position, and the item will be inserted according to frequency. This extension of the observation period for frequent items (shown in Example 2 of Fig. 5) helps delay the transition of frequent items into the sketch.
(3) If neither the LRU nor the LFU has recorded item e, and if there is space in the two tiers, it is inserted at the head of the LRU, and other items move toward the tail. If the LRU is full, the oldest item e' is evicted into LFU by frequency.

[4]Potentially frequent items are currently infrequent items that may become frequent as more access evidence is accumulated (*e.g.,* cold-start phase).

Algorithm 2: TBO-based Frequent-Tier Insertion

Input: Incoming item e
Data: LRU (head: newest, tail: oldest), LFU (head: frequent, tail: infrequent)

if $e \in$ *LRU* ***or*** *LFU* **then**
  Move $\langle e, f+1 \rangle$ to the head of LRU;
  Shift other items toward the tail in LRU;
  if $e \in$ *LFU* **then**
    Move $\langle e', f' \rangle$ from LRU to LFU;
    Sort LFU by frequency;
    return;
else
  if *LRU not full* **then**
    Insert $\langle e, f+1 \rangle$ into LRU at head;
    Shift other items toward the tail in LRU;
    return;
  if *LFU not full* **then**
    Move $\langle e', f' \rangle$ from LRU to LFU;
    Sort LFU by frequency;
    Shift other items toward the tail in LRU;
    Insert $\langle e, f+1 \rangle$ into LRU at head;
    return;

As mentioned earlier, frequent items reside in the LRU and LFU layers. To protect the frequent tier from being displaced by newly arriving infrequent items, we introduce a fast transition path that directly inserts infrequent candidates into the sketch when both the LRU and LFU are full and the item does not hit either tier. However, this design may cause some cold-start frequent items to bypass the buffer and be inserted into the sketch temporarily.

3) Infrequent-Tier Transition via ECT: To address this issue, we introduce the second key mechanism, Error Correction Transition (ECT). As shown in Fig. 6, ECT is designed to quickly restore cold-start frequent items to the LFU once sufficient frequency evidence is accumulated, thereby reducing the time frequent items stay in the sketch. In essence, ECT accelerates the upward transition of potentially frequent items from the sketch.

Infrequent-Tier Insertion (Alg. 3): For an incoming item e, if it misses the frequent tier and the frequent tier is full:
(1) The counter value at the position corresponding to the hashed item e in the sketch is retrieved. This value is then compared against the frequency of the item e' with the minimum frequency in the LFU.
(2) If the counter value plus one is smaller than the frequency of e', the counter associated with e in the sketch is updated to the incremented value (shown in Example 1 of Fig. 6).
(3) If the incremented counter value is greater than or equal to the frequency of e' (shown in Example 2 of Fig. 6), item e transitions upward from the sketch to the LFU, replacing item e'. In this case, the ID of e' is replaced with that of e, and the frequency of e is set to the counter value plus one.

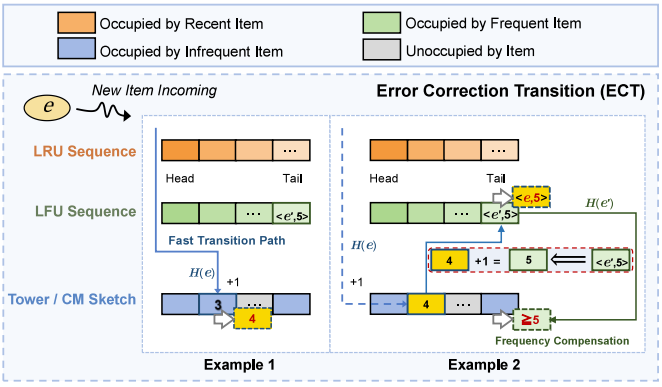


Fig. 6: Error Correction Transition Example. $H(\cdot)$ denotes the simplified form of the three hash functions $h(\cdot)$ used in the Tower/CM sketch, corresponding to the three-tier bucket arrays.

(4) The counter value f'' of the evicted item e', which returns to the sketch, is then adjusted to be no smaller than its previous frequency f' in the LFU.

This frequency compensation ensures consistency because LFU counters do not generate errors, thereby helping to reduce estimation error in the sketch.

Algorithm 3: ECT-based Infrequent-Tier Insertion

Input: Incoming item e
Data: LFU (head: frequent, tail: infrequent), sketch

if $e \notin$ *LRU* $\wedge$ $e \notin$ *LFU* ***and*** *both LRU and LFU are full* **then**
  $f \leftarrow f + 1$;
  if $f \geq f'$ **then**
    Replace e' with e in LFU;
    Insert e' into sketch;
    $f'' \leftarrow \max(f', f'')$;
    return;
  else
    Insert e into the sketch;
    return;

Based on the transition mechanisms described above, AlignSketch ensures that each item is recorded in exactly one tier at any time. This property allows query operations to follow a once-lookup-and-return strategy: once the queried item is found in a tier, the search terminates immediately, without the need to traverse all tiers. For top-K queries, only the LRU and LFU layers are involved, since by design, all frequent items are maintained within the frequent tier. For frequency estimation queries, the item is searched sequentially in the LRU layer, the LFU layer, and the sketch layer. The search stops as soon as the item is found, and the corresponding frequency estimate is returned.

C. Selection of Data Structures

For the frequent tier, to select a more efficient data structure implementation, we analyze the time complexity of LRU

and LFU under two implementations, queue-based and hash-bucket-based, assuming that each can hold up to N items.

Queue-based structure: For LRU, items are maintained in temporal order, enabling $O(1)$ insertion at the head and eviction from the tail, while queries require a full traversal and incur $O(N)$ time. For LFU, items are organized in frequency order, resulting in $O(N)$ time for both insertion and query.

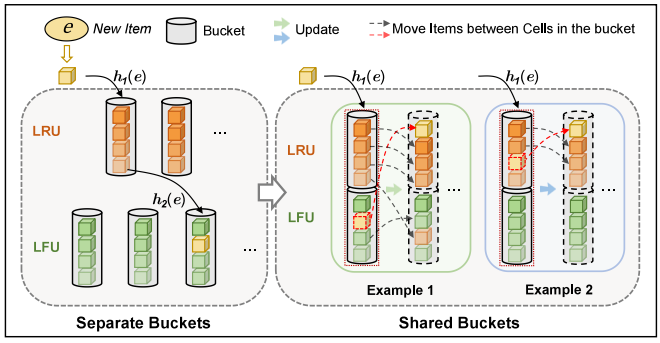


Fig. 7: Hash-bucket-based LRU and LFU with shared buckets.

Hash-bucket-based structure: We replace the queue structures of LRU and LFU with hash buckets, where each bucket contains a small number of cells storing different items. By allowing LRU and LFU to share the same hash function and merging their buckets, transitions between the two tiers are confined to bucket-local operations. As illustrated in Fig. 7, when a new item arrives, it only needs to swap positions within a constant number of cells in the same bucket, avoiding repeated hashing and improving execution efficiency.

Compared to the queue-based implementation with $O(N)$ insertion time, the hash-bucket-based structure achieves $O(1)$ insertion. For queries, although the worst case may traverse all tiers, each bucket contains only a constant number of cells, so the query complexity remains $O(1)$ in practice. All theoretical analysis (§ V) and experimental evaluation (§ VI) in this paper are conducted based on the hash-bucket-based structure. All memory-related comparisons are conducted under the same total memory, which for AlignSketch includes shared LRU and LFU hash buckets in the frequent tier. Each bucket contains 8 cells, and each cell stores an item ID and its counter, both using 4 bytes as commonly adopted in prior work [37], [38].

In our framework, estimation errors arise solely from the infrequent tier, regardless of whether queue-based or hash-bucket-based structures are used for LRU and LFU. The frequent tiers introduce no error, thereby confining the error source. Even if an item passes through the infrequent tier before returning to LRU or LFU, any error is incurred only during that period. Accordingly, the sketch for the infrequent tier is selected to provide theoretical error bounds, even if they are loose; our framework can further optimize them.

V. Mathematical Analysis

In this section, we provide a rigorous theoretical analysis to demonstrate that our framework AlignSketch can achieve optimized error bounds.

A. Prerequisites for Theoretical Analysis

Given that sketches are primarily deployed in distributed systems, we conduct our analysis under a typical scenario of network traffic processing. The analyzed data stream $\mathcal{S}$ contains n items distributed among m distinct IDs, where n denotes the total frequency of all items in the stream.

In the frequent tier, we utilize a hash function $h(\cdot)$ with w buckets. Each bucket employs two recording strategies, LRU and LFU, and consists of d cells (see Fig. 7). Each cell stores exactly one item. When an incoming item with ID e_i ($i \in 1, 2, \cdots, m$) arrives, it is initially mapped to the bucket indicated by $h(e_i)$. If the frequent tier is full, e_i is inserted into the infrequent tier. When the infrequent tier adopts the Tower sketch, it consists of three counter layers, each with a different size. For example, if the counter sizes of each layer are 8-bit, 4-bit, and 2-bit, respectively, then the corresponding numbers of counters are l, $2l$, and $4l$. Similarly, when the infrequent tier adopts the CM sketch, each of its three layers contains l counters, which are 8-bit in size. The l can be determined when the available memory of our algorithm is given.

B. Theoretical Proof for Error Bound

Let f_i denote the true frequency of e_i, $\hat{f}_i$ the estimated frequency from the AlignSketch. Based on these definitions, we derive the following theorems.

Theorem 1. *In the frequent tier, for each bucket j ($j \in 1, 2, \cdots, w$), consider an e_i such that $h_1(e_i) = j$. Let $s_j = \sum_{h_1(e_i)=j} f_i$ denote the total frequency mapped to bucket j, and let $\delta_j = \max_{h_1(e_i)=j}(\hat{f}_i - f_i)$ denote the maximum estimation error in that bucket. If $f_i > \frac{s_j}{d} + \delta_j$, then e_i is guaranteed to be recorded in bucket j.*

Proof. Estimation errors originate solely from the infrequent tier. If an item always remains in the frequent tier, its estimate is exact; otherwise, any overestimation incurred when entering the infrequent tier is bounded by δ_j. Thus, for any cell in bucket j of the frequent tier, the error is at most δ_j.

Let s_j denote the total true frequency of the d items stored in bucket j. The total estimated frequency in the frequent tier of bucket j is therefore at most $s_j + d\delta_j$, implying that the minimum estimated frequency among these d items is at most $\frac{s_j}{d} + \delta_j$. Now consider an item e_i with $f_i > \frac{s_j}{d} + \delta_j$. At the last update, when e_i enters the framework, if it is already in the frequent tier, its estimated frequency exceeds the minimum, and it cannot be evicted thereafter. Otherwise, its estimated frequency, being no smaller than f_i, also exceeds the minimum, and e_i is promoted to the frequent tier. In either case, once in the frequent tier, e_i will not be evicted in subsequent updates. □

Next, we derive the miss rate by introducing the d-tail distribution, which denotes the fraction of total frequency not captured by the frequent tier when each bucket stores up to d distinct IDs. *i.e.,* it represents the contribution of items outside the top-d to the total stream frequency.

Definition 1. Let Ω_j denote the total frequency of the top-d most frequent IDs mapped to the j-th bucket, where $j \in 1, 2, \cdots, w$. The d-tail distribution, denoted by η_d

$$\eta_d = 1 - \frac{\sum_{j=1}^{w} \Omega_j}{n} \tag{1}$$

Lemma 1. *Consider the data streams of incoming items recorded using the LFU. Let u_k denote the frequency of the k-th most frequent ID, and define $u = \sum u_k$ ($k \to \infty$) as the total frequency. When u is sufficiently large and $\mathcal{S}$ has been observed for a long enough period, the probability P_d that the next incoming item belongs to one of the top-d entries satisfies*

$$P_d = \frac{\sum_{k=1}^{d} u_k}{u} \tag{2}$$

Proof. The property of LFU ensures that the first d IDs are bound to be $u_1, ..., u_d$ when a sufficient number of items have been entered. Therefore, P_d equals the sum of the relative frequency of the most frequent d IDs, namely Eq. (2). □

Lemma 1 shows that the miss rate of the LFU with d cells in the bucket is equal to the d-tail distribution.

Lemma 2. *Consider a stream of incoming items recorded using the LRU. Let the frequency of the k-th most frequent ID be denoted by u_k, and define $u = \sum u_k$, with $k \to \infty$. When u is sufficiently large and enough time has passed, the probability P_d that the next incoming item belongs to one of the top-d entries satisfies*

$$P_d \geq 1 - 1.78\left(1 - \frac{\sum_{k=1}^{d} u_k}{u}\right) \tag{3}$$

Proof. As reported by Faloutsos *et al.* [63], an α-skewed power-law distribution of item frequencies is commonly observed in network traffic. According to [64], after a sufficiently long period of time, the probability that an incoming item does not belong to the first d IDs is given by $\mathcal{K}(\alpha)(1 - \frac{\sum_{k=1}^{d} u_k}{u})$, where

$$\mathcal{K}(\alpha) = (1 - \frac{1}{\alpha})[\Gamma(1 - \frac{1}{\alpha})]^{\alpha}$$

Since $\mathcal{K}(\alpha)$ is increasing and $\lim_{\alpha\to\infty} \mathcal{K}(\alpha) = e^{\gamma} \approx 1.78$, We have

$$\begin{aligned} P_d &= 1 - \mathcal{K}(\alpha)(1 - \frac{\sum_{i=1}^{d} u_k}{u}) \\ &\geq 1 - 1.78(1 - \frac{\sum_{i=1}^{d} u_k}{u}) \end{aligned}$$

According to [64], the above conclusion (see Eq. 3) still holds for all *light-tailed distributions* (*e.g.*, the normal distribution). Therefore, Lemma 2 also holds beyond the power-law distribution assumption. □

Theorem 2. *After a sufficiently long period, the possibility P_{miss} that an incoming item is not in the frequent tier satisfies the inequality, where $P_{miss} = 1 - P_d$*

$$\eta_d \leq P_{miss} \leq 1.78\eta_d \tag{4}$$

Proof. Suppose each bucket contains a entries for LRU and b entries for LFU, where $a+b = d$. Let $s_{j,k}$ denote the frequency of the k-th most frequent ID in the j-th bucket, and define $s_j = \sum_{k=1}^{d} s_{j,k}$, where $s_{j,k}$ corresponds to u_k in Eqs. 2–3. If the next item belongs to bucket j, let $P_{j,\text{miss}}$ represent the probability that it is not in the frequent tier. Consider the optimal case where the d entries store exactly the d most frequent IDs. According to Lemma 1, we have $P_{j,\text{miss}} \geq \eta_{j,d}$, where $\eta_{j,d} = 1 - \frac{\sum_{k=1}^{d} s_{j,k}}{s_j}$. This represents the theoretical best lower bound for $P_{j,\text{miss}}$. According to Lemma 2, the possibility that it does not belong in LRU is at most $1.78\frac{\sum_{k=a+1}^{\infty} s_{j,k}}{s_j}$. On the other hand, if an item is not contained in the LRU, it can still be in LFU, particularly if it belongs to the set of most frequently accessed items outside the LRU. In this case, the probability of being in LFU is at least $\frac{\sum_{k=a+1}^{a+b} s_{j,k}}{\sum_{k=a+1}^{\infty} s_{j,k}}$. Thus

$$P_{j,miss} \leq 1.78\frac{\sum_{k=d+1}^{\infty} s_{j,k}}{s_j}$$

Finally, we have

$$\begin{aligned} P_{miss} &= \sum_{j=1}^{w} \frac{s_j}{n} P_{j,miss} \\ &\leq \frac{1.78}{n}(\sum_{j=1}^{w} \sum_{k=d+1}^{\infty} s_{j,k}) \\ &= 1.78(1 - \frac{\sum_{j=1}^{w} \Omega_j}{n}) \\ &= 1.78\eta_d \end{aligned}$$

□

Theorem 3. *After all the data streams have been inserted, consider bucket j. Suppose that, among the d finalists in its frequent tier, c of them have never been in the frequent tier before. Let ϕ_j represent the average number of items that enter the infrequent tier due to each miss. We observe the following*

$$\phi_j \leq \frac{\sum_{k=d-c+1}^{\infty} s_{j,k}}{\sum_{k=d+1}^{\infty} s_{j,k}} \tag{5}$$

Proof. Consider the infrequent tier after the entire insertion. The number of items that enter the infrequent tier is no more than $\sum_{k=d-c+1}^{\infty} s_{j,k}$, while the number of missing cases is at least $\sum_{k=d+1}^{\infty} s_{j,k}$. Consequently, Eq. 5 is obtained. □

Theorem 2 provides the upper and lower bounds of the miss rate of the frequent tier. Theorem 3 presents the average cost per miss, which approaches 1 when n is sufficiently large, under common distributions such as power-law and all light-tailed distributions (*e.g.*, the normal distribution). These two properties restrict the number of items entering the infrequent tier, effectively optimizing the estimation error.

We now derive the specific error bound based on Theorems 2–3 and the algorithm for the infrequent tier.

Theorem 4. *Let ϕ be the weighted average of ϕ_i based on the number of misses. If the infrequent tier is instantiated*

with the Tower sketch (i.e., our algorithm instance AlignSketch(Tower)), then by Theorem 3, for any arbitrarily small positive ϵ that characterizes the admissible error tolerance, the estimation error of item e_i is bounded by

$$\Pr(|\hat{f}_i - f_i| \leq \epsilon n) \geq 1 - \frac{(\phi P_{miss})^3}{8\epsilon^3 l^3} \tag{6}$$

Proof. Under the same configuration, the error bound of the original Tower sketch from [7] is given by

$$\Pr(|\hat{f}_i - f_i| \leq \epsilon n) \geq 1 - \frac{1}{8\epsilon^3 l^3} \tag{7}$$

Considering that the number of items entering the Tower sketch is no more than $\phi P_{miss} n$, the error bound of the Tower sketch from Eq. 7 immediately applies, yielding Eq. 6. □

TABLE II: Comparison of the error bound.

Algorithm	Error Bound
Tower Sketch	$\Pr(\lvert\hat{f}_i - f_i\rvert \leq \epsilon n) \geq 1 - \frac{1}{8\epsilon^3 l^3}$
AlignSketch(Tower)	$\Pr(\lvert\hat{f}_i - f_i\rvert \leq \epsilon n) \geq 1 - \frac{(\phi P_{\text{miss}})^3}{8\epsilon^3 l^3}$
CM Sketch	$\Pr(\lvert\hat{f}_i - f_i\rvert \leq \epsilon n) \geq 1 - \frac{1}{\epsilon^3 l^3}$
AlignSketch(CM)	$\Pr(\lvert\hat{f}_i - f_i\rvert \leq \epsilon n) \geq 1 - \frac{(\phi P_{\text{miss}})^3}{\epsilon^3 l^3}$

In Table II, we present the error bounds of the original Tower sketch, AlignSketch(Tower), the original CM sketch, and AlignSketch(CM), respectively. *To ensure fairness*, we compare the error bounds of each algorithm under the same total memory budget. Specifically, we fix the number of counters l per layer and adjust the size of each counter to ensure that all algorithms use the same total memory. For example, in the comparative analysis between the original Tower sketch and AlignSketch(Tower), we assume both are allocated the same total memory, denoted as M. If, in AlignSketch(Tower), the frequent tier and the infrequent tier each occupy half of M, then the counter size in each layer of the Tower sketch component in AlignSketch(Tower) is set to be half of that in the original Tower sketch, ensuring equal total memory usage between the two approaches.

We define the Optimization Factor as the ratio of the degree of improvement to the original degree. The focus of optimization is the difference between two probabilities. As an example, we calculate the optimization factor by comparing the error bound of AlignSketch(Tower) with the original Tower sketch. Thus, the difference terms are $\frac{(\phi P_{miss})^3}{8\epsilon^3 l^3}$ and $\frac{1}{8\epsilon^3 l^3}$, respectively. The optimization factor can be calculated as the ratio of the difference terms, *i.e.*, the optimization factor equals $\frac{1}{(\phi P_{miss})^3}$. Importantly, ϕP_{miss} is a small coefficient related to the tail distribution, often much smaller than 1 (*i.e.*, $\phi P_{miss} \ll 1$). Table II shows that AlignSketch(Tower) improves by a factor of $\frac{1}{(\phi P_{miss})^3}$ and $\frac{8}{(\phi P_{miss})^3}$ over the original Tower and CM, respectively. At the same time, AlignSketch(CM) improves by a factor of $\frac{1}{8(\phi P_{miss})^3}$ and $\frac{1}{(\phi P_{miss})^3}$ over the original Tower and CM, respectively.

By comparing the original Tower with AlignSketch(Tower), and the original CM with AlignSketch(CM), we further demonstrate that our framework can optimize the error bounds of other sketches. Meanwhile, we compare the error bounds of our sketch when using the Tower and the CM as the infrequent tier, and find that the former provides a superior error bound. On this basis, we experimentally validate (see § VI) that our theoretical error bounds are closely aligned with the practical estimation errors, demonstrating that our framework achieves tight error bounds.

VI. Experimental Results

TABLE III: Comparison algorithms used in this paper.

Query Tasks	Comparison Algorithms
Top-K Queries	Elastic, SS, USS, Ladder Filter, Cold Filter, Ours
Frequency Estimation	Elastic, Tower, Ladder Filter, Cold Filter, Ours

In this section, we conduct extensive experiments to evaluate our proposed framework across three key aspects: the alignment between theoretical error bounds and practical estimation errors, query accuracy, and insertion efficiency. Specifically, we instantiate the framework with two algorithms, AlignSketch(Tower) and AlignSketch(CM), for practical evaluation. The experiments are carried out on two common query tasks: top-K queries and frequency estimation. A summary of the compared algorithms is provided in Table III. Our evaluation aims to answer the following questions:

Q1: As the classification-based sketch, does our framework achieve alignment between theory and practice? (§ VI-B)

Q2: Within our framework, which configuration achieves higher accuracy ? (§ VI-C)

Q3: How accurate is our framework compared with existing SOTA methods? (§ VI-D)

Q4: How efficient is our framework, and how do multithreaded designs further affect its efficiency? (§ VI-E)

A. Experimental Setup

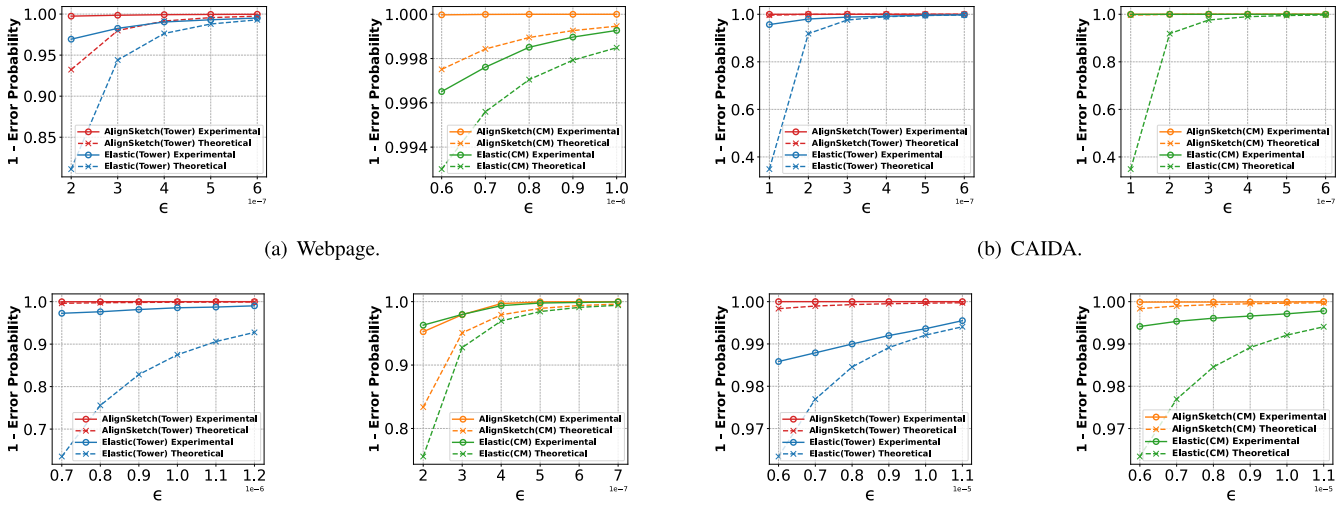
Platform: All tests are conducted on a single-CPU server (Intel(R) Core(TM) i9 − 10980XE CPU @ 3.00GHz) with 36 cores (18 cores per socket, 2 threads per core), equipped with 128GB of memory and 24.8MB of L3 cache.

Default Settings: We implement AlignSketch and its comparative algorithms in C++ and anonymously release all relevant source code [41]. Each hash bucket integrates LRU and LFU strategies containing 8 cells; similarly, the heavy part of Elastic also employs buckets with 8 cells each. The Ladder Filter comprises two LRU tiers, the bucket of each tier equipped with 4 cells, totaling 8 cells across both tiers. Parameters for Cold Filter, Tower, USS, and SS follow the recommendations from prior studies [7], [11], [34], [38].

Datasets: We evaluate on the following four datasets.

(1) *Webpage.* The Webpage dataset consists of a large number of HTML web documents [42], with a total of approximately 3.3×10^7 items, among which there are 9.5×10^5 distinct items. The maximum frequency of any distinct item is 1.6×10^5.

(2) *CAIDA.* We use the CAIDA dataset [43], a public collection of anonymized real-world network traces from high-speed Internet backbone links. The dataset contains 4.2×10^8 items with 1.6×10^6 distinct items, and the maximum frequency



(a) Webpage. (b) CAIDA.

(c) Zipf. (d) Normal.

Fig. 8: Theoretical *vs.* Experimental Comparison of Estimation Error. (1 - Error Probability) closer to 1 indicates a better error bound.

of a single item reaches 3.7×10^6. It is widely used as a representative benchmark in burst detection research [55].

(3) *Zipf.* We generate a synthetic dataset following a Power-law distributed [65], with a total of 1.6×10^7 items. The dataset contains 8.7×10^5 distinct items, and the maximum frequency of distinct items is 1.1×10^6. The dataset with skewness α=1.0.

(4) *Normal.* A synthetic dataset with $X \sim \mathcal{N}(3.2 \times 10^3, 1.5 \times 10^7)$ is generated following a Normal distribution [41], containing a total of 2×10^8 items. It contains 6.3×10^4 distinct items, and the maximum frequency of distinct item is 1.2×10^4.

Evaluation Metrics: We use the following metrics for accuracy and efficiency.

(1) *Average Relative Error (ARE).* $\frac{1}{|\Psi|} \sum_{e_i \in \Psi} \frac{|\hat{f}_i - f_i|}{f_i}$, where f_i is the ground truth frequency of item e_i, $\hat{f}_i$ is its estimated frequency, and Ψ is the query set.

(2) *Zero Error Rate.* The proportion of items selected by our sketch whose estimated frequency is guaranteed to be the same as its ground truth frequency.

(3) *F1 Score.* $\frac{2*CR*PR}{CR+PR}$, where PR (Precision rate) represents the proportion of the correctly selected items among all the selected items, and CR (Recall rate) represents the proportion of the correctly selected items among all the real top-K items.

(4) *Throughput.* The number of proportions (insertions) in million per second (Mops).

B. Evaluation of Theory–Practice Alignment

To answer ***Q1***, we evaluate our method on four datasets with diverse distributions: two real-world datasets (Webpage and CAIDA) and two synthetic datasets (Zipf and Normal).

Experimental results (Fig. 8): We select Elastic sketch [37], a classification-based sketch with established theoretical error bounds, as the comparison algorithm to analyze the alignment between theoretical guarantees and practical estimation errors. In contrast, other classification-based sketches listed in Table III, including the Ladder Filter [38] and the Cold Filter [39], have not yet provided corresponding theoretical error bounds, making such comparisons infeasible. The original Elastic uses the CM sketch to record infrequent items. To demonstrate the benefits of our proposed framework, we set the tier of Elastic that records infrequent items with the CM sketch and the Tower sketch, which correspond to the two algorithm intances of our framework, AlignSketch(Tower) and AlignSketch(CM). In Fig. 8, the theoretical values of *1 - Error Probability* for our two algorithm intances, are derived from Table II[5], representing the proportion of data streams meeting our allowable error-tolerance (ϵ); *this value is preferable when closer to* 1. Accordingly, we focus on the range of ϵ values for which this metric is close to 1, as it better reflects practically meaningful estimation errors. Importantly, by assessing the distance between the theoretical (dashed line) and experimental (solid line) results for each algorithm in Figs. 8(a)–8(d), it is evident that, across the four datasets following different distributions, our methods exhibit a tighter alignment and outperform the SOTA algorithm. As the ϵ decreases, reflecting a stricter accuracy requirement, the alignment between the theoretical and experimental estimation errors of our algorithm becomes increasingly pronounced.

Due to space limitations, subsequent experiments are conducted on a representative dataset (Webpage), as similar trends are observed across all four datasets. Complete results on the remaining datasets are available on GitHub [41].

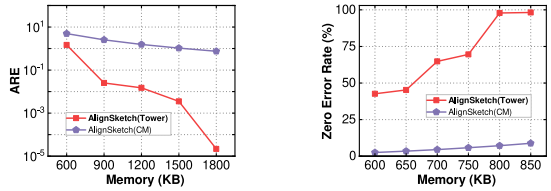
C. Evaluation of Configurations on Accuracy

To answer ***Q2***, we evaluate the impact of different configurations on the accuracy of our framework using the Webpage.

Impact of Infrequent-tier Selection (Fig. 9): We configure the infrequent tier of our framework with the Tower sketch and the CM sketch, which correspond to our algorithm instances AlignSketch(Tower) and AlignSketch(CM), respectively, and

[5]To facilitate comparison, we replace $\Pr(\hat{f}_i - f_i \leq \epsilon n)$ with $1 - \Pr(\hat{f}_i - f_i > \epsilon n)$ in fig. 8, *i.e.*, 1 - Error Probability.

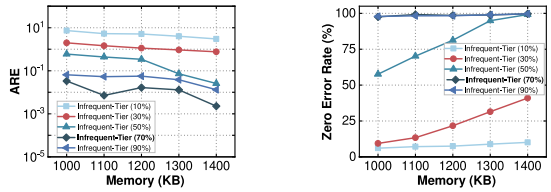
compare their accuracy. As top-K queries are mainly determined by the frequent tier, we evaluate infrequent-tier designs primarily via frequency estimation. As shown in Fig. 9, using the Tower sketch as the infrequent tier consistently results in lower ARE and higher Zero Error Rate compared to the other method, with the advantage becoming more evident as memory increases. When memory exceeds 900 KB, the approach achieves at least an order of magnitude lower ARE than the alternative. For Zero Error Rate, it outperforms the others by up to 3× when memory exceeds 800 KB.



(a) ARE. (b) Zero Error Rate.

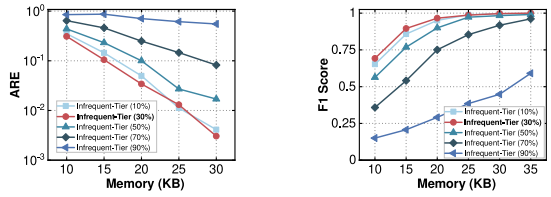
Fig. 9: Experiments for the impact of infrequent-tier selection.

Therefore, we adopt the algorithm instance setting with the Tower sketch (*i.e.*, AlignSketch(Tower)), which achieves higher accuracy, for subsequent experiments in this paper.



(a) ARE. (b) Zero Error Rate.

Fig. 10: Experiments on the proportion of infrequent-tier memory in frequency estimation.



(a) ARE. (b) F1 Score.

Fig. 11: Experiments on the proportion of infrequent-tier memory in top-K queries.

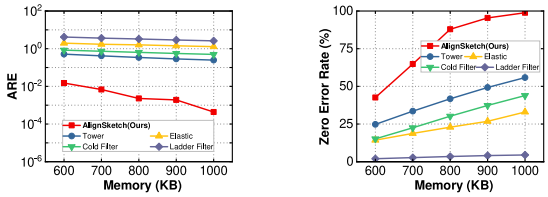
Impact of Infrequent-tier Memory Proportion (Fig. 10 - Fig. 11): Within a given range of total memory budgets, we evaluate the effect of allocating 10% and 90% of the memory to the infrequent tier for each fixed budget, and investigate in these proportions affect the overall accuracy of our algorithm. For frequency estimation, accuracy improves as the Tower proportion increases, reaching the optimal ARE and Zero Error Rate at 70%. However, beyond this point, the estimation error begins to rise. For top-K queries, the best accuracy is achieved when 30% of the memory is allocated to the Tower. The ARE and F1 Score metrics change with the Tower proportion, deteriorating when it deviates from 30%.

We also evaluate this proportion on the remaining three datasets and find that, for the same query task, the preferred value varies slightly across datasets; thus, we use this proportion as the default setting in subsequent experiments. Meanwhile, since the optimal proportion depends on the dataset and the query task, it can be adjusted accordingly; we provide practical selection guidelines in § VII.

D. Evaluation of Accuracy with SOTA

To answer ***Q3***, this subsection compares the accuracy of our most accurate algorithm instance, AlignSketch(Tower), *i.e.*, LRULFU+Tower, against SOTA approaches listed in Table III, across two common query tasks. The accuracy is evaluated under various memory constraints using several metrics (§ VI-A), including ARE, Zero Error Rate, and F1 Score. ARE is the primary evaluation metric, with a particular focus on cases where *the value is below* 1.

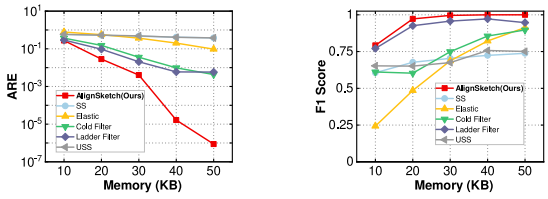
Frequency Estimation (Fig. 12): We compare the estimation error of our algorithm with SOTA methods using ARE and Zero Error Rate. The Fig. 12(b) shows that our algorithm outperforms the SOTA methods by nearly 1 to 2 orders of magnitude for the ARE. As shown in Fig. 12(a), our algorithm requires the least memory size to achieve a Zero Error Rate exceeding 90% compared to others.



(a) ARE. (b) Zero Error Rate.

Fig. 12: Experiments for frequency estimation.

Top-K Queries (Fig. 13): As the available memory increases, our approach consistently demonstrates superior performance over the SOTA methods across all accuracy-related evaluation metrics. In particular, when the memory budget reaches 50 KB, our method achieves up to 3 orders of magnitude reduction in ARE compared to existing SOTA solutions. At the same time, our method consistently achieves higher F1 Scores than all baselines across all memory configurations.



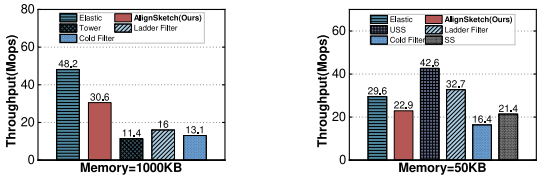
(a) ARE. (b) F1 Score.

Fig. 13: Experiments for top-K queries.

E. Evaluation of Efficiency and Multithreaded Optimization

To answer ***Q4***, we compare insertion throughput with SOTA methods under single-threaded settings, and further evaluate

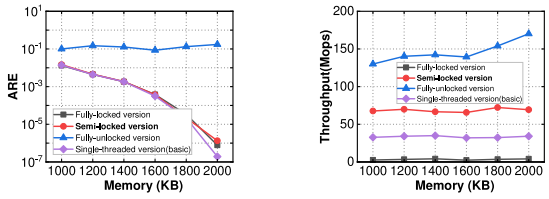
multithreaded variants of AlignSketch to study parallel optimization strategies for improving efficiency, along with their throughput–accuracy trade-offs.



(a) Frequency Estimation. (b) Top-K Queries.

Fig. 14: Experiments for insert throughput.

Insertion Throughput (Fig. 14): We consider two memory settings: common memory, suitable for frequency estimation, and small memory, suitable for top-K queries. As shown in Fig. 14, lightweight sketches such as USS and Ladder Filter achieve higher insertion throughput, particularly under the small memory setting. In contrast, AlignSketch incurs additional bookkeeping overhead from LRU/LFU maintenance and tier transitions, leading to a moderate reduction in throughput. Nevertheless, the throughput gap remains within the same order of magnitude across both memory settings. Under the common memory setting, AlignSketch achieves 30.6 Mops, which is about 63% of Elastic and substantially higher than Tower, Ladder Filter, and Cold Filter. This design trades insertion throughput for tighter theory–practice alignment (§ VI-B), together with improved estimation accuracy by nearly 1 to 2 orders of magnitude (§ VI-D), and thus provides clear benefits when accuracy and theory–practice alignment are priorities.



(a) ARE. (b) Throughput.

Fig. 15: Experiments for multithread variants.

Multithreaded Optimization (Fig. 15): To improve the efficiency of our algorithm, we develop three multithreaded variants: fully-locked, fully-unlocked, and semi-locked. The fully locked variant applies locks to both tiers to synchronize concurrent insert operations and ensure atomicity. In contrast, the fully unlocked variant removes all locks to maximize throughput; however, concurrent updates may lead to lost frequency increments, resulting in reduced accuracy. As shown in Fig. 15, the fully locked variant achieves the lowest throughput, whereas the fully unlocked variant attains the highest throughput at the cost of degraded accuracy. To balance these trade-offs, the semi-locked variant locks only the Frequent tier. Under the same memory constraints, the semi-locked variant achieves approximately a 2× throughput improvement over the single-threaded baseline without a noticeable loss in accuracy. Therefore, we recommend the semi-locked variant as the preferred multithreaded optimization.

VII. Discussion

Hardware deployment feasibility. Given that sketches may be deployed on programmable switches, smart NICs, and FPGAs, we adopt the following adaptations:

Programmable switches. On switches, match–action pipelines and limited on-chip memory restrict transition logic with multiple or dependent accesses; thus, transitions should be simplified into single-pass, pipeline-local updates (*e.g.*, in the spirit of P4LRU [66]).

Smart NICs. Smart NICs offer more programmability but remain constrained by per-packet latency/compute budgets at high rates; transition logic should reduce branching and memory-access overhead, with complex processing selectively offloaded to the control plane.

FPGAs. For FPGAs, clock frequency, timing closure, and on-chip memory constraints necessitate compact state and carefully pipelined transition logic; for specific queries, one can retain only the heavier tier in hardware (*e.g.*, infrequent for frequency estimation, frequent for top-K).

Workload robustness and configuration. Bursty traffic, characterized by short-term frequency surges and strong temporal locality, is common in real-world stream processing. The evaluation in this paper includes representative bursty workloads such as CAIDA, which has been widely used in prior burst-detection studies [55]. The results show that AlignSketch still exhibits close theory–practice agreement in estimation errors, while maintaining high accuracy. This robustness is attributed to the bidirectional transition mechanism and the frequent-tier structure, which keep error propagation within a bounded range and thus make the overall estimation more stable.

The memory proportion further provides a lightweight configuration knob that helps AlignSketch adapt to different tasks and datasets. As shown in § VI-C, frequency estimation generally benefits from allocating more than 50% of memory to the infrequent tier, whereas top-K queries tend to prefer less than 50%. In offline settings, this proportion can be tuned using the available dataset to maximize accuracy. In online real-time deployments, where the workload is unknown a priori, we recommend using the proportion suggested in this paper as the initial configuration, and refining it by mirroring a small fraction of traffic for offline analysis.

VIII. Conclusion

In this paper, we propose AlignSketch, a general classification framework for data stream processing. By leveraging theoretical miss-rate results of LRU and LFU, we derive tighter error bounds for AlignSketch. Experimental results show that AlignSketch outperforms SOTA approaches, achieving alignment with theoretical analyses and reducing estimation errors by nearly 1 to 2 orders of magnitude. The code is publicly available [41] to support future research and applications.

Acknowledgment. This work was supported by the National Key Research and Development Program of China under Grant No. 2024YFB2906602, and in part by the National Natural Science Foundation of China (NSFC) (No. 62372009).

AI-Generated Content Acknowledgement

The authors confirm that no generative AI tools were used in the preparation of this manuscript.

References

[1] Haoyu Li, Liuhui Wang, Qizhi Chen, Jianan Ji, Yuhan Wu, Yikai Zhao, Tong Yang, and Aditya Akella. Chainedfilter: Combining membership filters by chain rule. *Proceedings of the ACM on Management of Data*, 1(4):1–27, 2023.

[2] Zhongxiang Wei, Ye Tian, Wei Chen, Liyuan Gu, and Xinming Zhang. Dune: Improving accuracy for sketch-int network measurement systems. In *IEEE INFOCOM 2023-IEEE Conference on Computer Communications*, pages 1–10. IEEE, 2023.

[3] Peng Jia, Pinghui Wang, Junzhou Zhao, Ye Yuan, Jing Tao, and Xiaohong Guan. Loglog filter: Filtering cold items within a large range over high speed data streams. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, pages 804–815. IEEE, 2021.

[4] Kaicheng Yang, Yuhan Wu, Ruijie Miao, Tong Yang, Zirui Liu, Zicang Xu, Rui Qiu, Yikai Zhao, Hanglong Lv, Zhigang Ji, et al. Chamelemon: Shifting measurement attention as network state changes. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 881–903, 2023.

[5] Qilong Shi, Chengjun Jia, Wenjun Li, Zaoxing Liu, Tong Yang, Jianan Ji, Gaogang Xie, Weizhe Zhang, and Minlan Yu. Bitmatcher: Bit-level counter adjustment for sketches. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 4815–4827, 2024.

[6] Qingjun Xiao, Xuyuan Cai, Yifei Qin, Zhiying Tang, Shigang Chen, and Yu Liu. Universal and accurate sketch for estimating heavy hitters and moments in data streams. *IEEE/ACM Transactions on Networking*, 31(5):1919–1934, 2023.

[7] Kaicheng Yang, Sheng Long, Qilong Shi, Yuanpeng Li, Zirui Liu, Yuhan Wu, Tong Yang, and Zhengyi Jia. Sketchint: Empowering int with towersketch for per-flow per-switch measurement. *IEEE Transactions on Parallel and Distributed Systems*, 2023.

[8] Hengrui Wang, Huiping Lin, Zheng Zhong, Tong Yang, and Muhammad Shahzad. Enhanced machine learning sketches for network measurements. *IEEE Transactions on Computers*, 72(4):957–970, 2022.

[9] Qun Huang, Siyuan Sheng, Xiang Chen, Yungang Bao, Rui Zhang, Yanwei Xu, and Gong Zhang. Toward nearly-zero-error sketching via compressive sensing. In *18th USENIX Symposium on Networked Systems Design and Implementation (NSDI 21)*, pages 1027–1044, 2021.

[10] Yikai Zhao, Wenchen Han, Zheng Zhong, Yinda Zhang, Tong Yang, and Bin Cui. Double-anonymous sketch: Achieving top-k-fairness for finding global top-k frequent items. *Proceedings of the ACM on Management of Data*, 1(1):1–26, 2023.

[11] Daniel Ting. Data sketches for disaggregated subset sum and frequent item estimation. In *Proceedings of the 2018 International Conference on Management of Data*, pages 1129–1140, 2018.

[12] Tong Yang, Yang Zhou, Hao Jin, Shigang Chen, and Xiaoming Li. Pyramid sketch: A sketch framework for frequency estimation of data streams. *Proceedings of the VLDB Endowment*, 10(11):1442–1453, 2017.

[13] Haipeng Dai, Yuankun Zhong, Alex X Liu, Wei Wang, and Meng Li. Noisy bloom filters for multi-set membership testing. In *Proceedings of the 2016 ACM SIGMETRICS International Conference on Measurement and Modeling of Computer Science*, pages 139–151, 2016.

[14] Pratanu Roy, Arijit Khan, and Gustavo Alonso. Augmented sketch: Faster and more accurate stream processing. In *Proceedings of the 2016 International Conference on Management of Data*, pages 1449–1463, 2016.

[15] Haipeng Dai, Muhammad Shahzad, Alex X Liu, and Yuankun Zhong. Finding persistent items in data streams. *Proceedings of the VLDB Endowment*, 10(4):289–300, 2016.

[16] Masoud Moshref, Minlan Yu, Ramesh Govindan, and Amin Vahdat. Scream: Sketch resource allocation for software-defined measurement. In *Proceedings of the 11th ACM Conference on Emerging Networking Experiments and Technologies*, pages 1–13, 2015.

[17] Minos Garofalakis, Daniel Keren, and Vasilis Samoladas. Sketch-based geometric monitoring of distributed stream queries. *Proceedings of the VLDB Endowment*, 6(10):937–948, 2013.

[18] Graham Cormode and Shan Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005.

[19] Kejun Guo, Fuliang Li, Jiaxing Shen, and Xingwei Wang. Advancing sketch-based network measurement: A general, fine-grained, bit-adaptive sliding window framework. In *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pages 1–10. IEEE, 2024.

[20] Fuheng Zhao, Punnal Ismail Khan, Divyakant Agrawal, Amr El Abbadi, Arpit Gupta, and Zaoxing Liu. Panakos: Chasing the tails for multidimensional data streams. *Proceedings of the VLDB Endowment*, 16(6):1291–1304, 2023.

[21] Zijie Zeng, Lin Cui, Mimi Qian, Zhen Zhang, and Kaimin Wei. A survey on sliding window sketch for network measurement. *Computer Networks*, 226:109696, 2023.

[22] Chuanxiong Guo, Lihua Yuan, Dong Xiang, Yingnong Dang, Ray Huang, Dave Maltz, Zhaoyi Liu, Vin Wang, Bin Pang, Hua Chen, et al. Pingmesh: A large-scale system for data center network latency measurement and analysis. In *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication*, pages 139–152, 2015.

[23] 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 *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, pages 113–126, 2017.

[24] Graham Cormode and Marios Hadjieleftheriou. Methods for finding frequent items in data streams. *The VLDB Journal*, 19:3–20, 2010.

[25] Ran Ben Basat, Gil Einziger, Roy Friedman, and Yaron Kassner. Randomized admission policy for efficient top-k and frequency estimation. In *IEEE INFOCOM 2017-IEEE Conference on Computer Communications*, pages 1–9. IEEE, 2017.

[26] Anup Agarwal, Zaoxing Liu, and Srinivasan Seshan. Heterosketch: Coordinating network-wide monitoring in heterogeneous and dynamic networks. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 719–741, 2022.

[27] 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 *Proceedings of the ACM Special Interest Group on Data Communication*, pages 334–350. 2019.

[28] Robert Schweller, Zhichun Li, Yan Chen, Yan Gao, Ashish Gupta, Yin Zhang, Peter A Dinda, Ming-Yang Kao, and Gokhan Memik. Reversible sketches: enabling monitoring and analysis over high-speed data streams. *IEEE/ACM Transactions on Networking*, 15(5):1059–1072, 2007.

[29] Taher M Ghazal, Nidal A Al-Dmour, Raed A Said, Alireza Omidvar, Urooj Yousuf Khan, Tariq Rahim Soomro, Haitham M Alzoubi, Muhammad Alshurideh, Tamer Mohamed Abdellatif, Abdullah Moubayed, et al. Ddos intrusion detection with ensemble stream mining for iot smart sensing devices. In *The Effect of Information Technology on Business and Marketing Intelligence Systems*, pages 1987–2012. Springer, 2023.

[30] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. One sketch to rule them all: Rethinking network flow monitoring with univmon. In *Proceedings of the 2016 ACM SIGCOMM Conference*, pages 101–114, 2016.

[31] Tao Li, Shigang Chen, and Yibei Ling. Per-flow traffic measurement through randomized counter sharing. *IEEE/ACM Transactions on Networking*, 20(5):1622–1634, 2012.

[32] Charikar M Chen K Farach-Colton, M Widmayer P Eidenbenz S Triguero, and F Morales R Conejo R Hennessy. M finding frequent items in data streams automata, languages and programming 2002 heidelberg springer 693 703 10.1007. *Google Scholar Google Scholar Cross Ref Cross Ref*.

[33] 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.

[34] 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*, pages 398–412. Springer, 2005.

[35] Ruijie Miao, Fenghao Dong, Yikai Zhao, Yiming Zhao, Yuhan Wu, Kaicheng Yang, Tong Yang, and Bin Cui. Sketchconf: A framework for automatic sketch configuration. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*, pages 2022–2035. IEEE, 2023.

[36] Peiqing Chen, Yuhan Wu, Tong Yang, Junchen Jiang, and Zaoxing Liu. Precise error estimation for sketch-based flow measurement. In

Proceedings of the 21st ACM Internet Measurement Conference, pages 113–121, 2021.

[37] 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 *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 561–575, 2018.

[38] Yuanpeng Li, Feiyu Wang, Xiang Yu, Yilong Yang, Kaicheng Yang, Tong Yang, Zhuo Ma, Bin Cui, and Steve Uhlig. Ladderfilter: Filtering infrequent items with small memory and time overhead. *Proceedings of the ACM on Management of Data*, 1(1):1–21, 2023.

[39] Yang Zhou, Tong Yang, Jie Jiang, Bin Cui, Minlan Yu, Xiaoming Li, and Steve Uhlig. Cold filter: A meta-framework for faster and more accurate stream processing. In *Proceedings of the 2018 International Conference on Management of Data*, pages 741–756, 2018.

[40] Tong Yang, Junzhi Gong, Haowei Zhang, Lei Zou, Lei Shi, and Xiaoming Li. Heavyguardian: Separate and guard hot items in data streams. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2584–2593, 2018.

[41] Source code related to AlignSketch. https://anonymous.4open.science/r/AlignSketch.

[42] Real-life transactional dataset. http://fimi.ua.ac.be/data/.

[43] The CAIDA Anonymized Internet Traces. http://www.caida.org/data/overview/.

[44] Fan Deng and Davood Rafiei. New estimation algorithms for streaming data: Count-min can do more. *Webdocs. Cs. Ualberta. Ca*, 2007.

[45] Chen-Yu Hsu, Piotr Indyk, Dina Katabi, and Ali Vakilian. Learning-based frequency estimation algorithms. In *International Conference on Learning Representations*, 2019.

[46] Kaiyu Li and Guoliang Li. Approximate query processing: What is new and where to go? a survey on approximate query processing. *Data Science and Engineering*, 3(4):379–397, 2018.

[47] Graham Cormode. Sketch techniques for approximate query processing. *Foundations and Trends in Databases. NOW publishers*, 15, 2011.

[48] Amit Goyal, Hal Daumé III, and Graham Cormode. Sketch algorithms for estimating point queries in nlp. In *Proceedings of the 2012 joint conference on empirical methods in natural language processing and computational natural language learning*, pages 1093–1103, 2012.

[49] Qilong Shi, Yuchen Xu, Jiuhua Qi, Wenjun Li, Tong Yang, Yang Xu, and Yi Wang. Cuckoo counter: Adaptive structure of counters for accurate frequency and top-k estimation. *IEEE/ACM Transactions on Networking*, 31(4):1854–1869, 2023.

[50] Chao Wang, Xu Li, Jiuzhen Zeng, Weimin Yin, and Ping Zhou. Prob-cs: A probabilistic cuckoo sketch for accurate network traffic measurement. *IEEE Internet of Things Journal*, 2024.

[51] Tong Yang, Haowei Zhang, Jinyang Li, Junzhi Gong, Steve Uhlig, Shigang Chen, and Xiaoming Li. Heavykeeper: an accurate algorithm for finding top-k elephant flows. *IEEE/ACM Transactions on Networking*, 27(5):1845–1858, 2019.

[52] Vibhaalakshmi Sivaraman, Srinivas Narayana, Ori Rottenstreich, Shan Muthukrishnan, and Jennifer Rexford. Heavy-hitter detection entirely in the data plane. In *Proceedings of the Symposium on SDN Research*, pages 164–176, 2017.

[53] Peiqing Chen, Dong Chen, Lingxiao Zheng, Jizhou Li, and Tong Yang. Out of many we are one: Measuring item batch with clock-sketch. In *Proceedings of the 2021 International Conference on Management of Data*, pages 261–273, 2021.

[54] Qun Huang, Patrick PC Lee, and Yungang Bao. Sketchlearn: Relieving user burdens in approximate measurement with automated statistical inference. In *Proceedings of the 2018 Conference of the ACM Special Interest Group on Data Communication*, pages 576–590, 2018.

[55] Zheng Zhong, Shen Yan, Zikun Li, Decheng Tan, Tong Yang, and Bin Cui. Burstsketch: Finding bursts in data streams. In *Proceedings of the 2021 International Conference on Management of Data*, pages 2375–2383, 2021.

[56] Joshua Plasse, Henrique Hoeltgebaum, and Niall M Adams. Streaming changepoint detection for transition matrices. *Data Mining and Knowledge Discovery*, 35(4):1287–1316, 2021.

[57] Peter J Denning. The working set model for program behavior. *Communications of the ACM*, 11(5):323–333, 1968.

[58] Will E Leland, Murad S Taqqu, Walter Willinger, and Daniel V Wilson. On the self-similar nature of ethernet traffic (extended version). *IEEE/ACM Transactions on networking*, 2(1):1–15, 1994.

[59] Leslie Lamport. Time, clocks, and the ordering of events in a distributed system. In *Concurrency: the Works of Leslie Lamport*, pages 179–196. 2019.

[60] Sem Borst, Varun Gupta, and Anwar Walid. Distributed caching algorithms for content distribution networks. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. IEEE, 2010.

[61] Theodore Johnson and Dennis Shasha. 2q: A low overhead high performance buffer management replacement algorithm. In *Proceedings of the 20th VLDB Conference*, pages 439–450, 1994.

[62] Elizabeth J O'neil, Patrick E O'neil, and Gerhard Weikum. The lru-k page replacement algorithm for database disk buffering. *Acm Sigmod Record*, 22(2):297–306, 1993.

[63] Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. *ACM SIGCOMM computer communication review*, 29(4):251–262, 1999.

[64] Predrag R Jelenković. Asymptotic approximation of the move-to-front search cost distribution and least-recently used caching fault probabilities. *The Annals of Applied Probability*, 9(2):430–464, 1999.

[65] Alex Rousskov and Duane Wessels. High-performance benchmarking with web polygraph. *Software*, 34(2):p.187–211, 2004.

[66] Yikai Zhao, Wenrui Liu, Fenghao Dong, Tong Yang, Yuanpeng Li, Kaicheng Yang, Zirui Liu, Zhengyi Jia, and Yongqiang Yang. P4lru: towards an lru cache entirely in programmable data plane. In *Proceedings of the ACM SIGCOMM 2023 Conference*, pages 967–980, 2023.