Pandora: An Efficient and Rapid Solution for Persistence-Based Tasks in High-Speed Data Streams

WEIHE LI, University of Edinburgh, United Kingdom

In data streams, persistence characterizes items that appear repeatedly across multiple non-overlapping time windows. Addressing persistence-based tasks, such as detecting highly persistent items and estimating persistence, is crucial for applications like recommendation systems and anomaly detection in high-velocity data streams. However, these tasks are challenging due to stringent requirements for rapid processing and limited memory resources. Existing methods often struggle with accuracy, especially given highly skewed data distributions and tight fastest memory budgets, where hash collisions are severe. In this paper, we introduce Pandora, a novel approximate data structure designed to tackle these challenges efficiently. Our approach incorporates the insight that items absent for extended periods are likely non-persistent, increasing their probability of eviction to accommodate potential persistent items more effectively. We validate this insight empirically and integrate it into our update strategy, providing better protection for persistent items. We formally analyze Pandora's error bounds to validate its theoretical soundness. Through extensive trace-driven tests, we demonstrate that Pandora achieves superior accuracy and processing speed compared to state-of-the-art methods across various persistence-based tasks. Additionally, we further accelerate Pandora's update speed using Single Instruction Multiple Data (SIMD) instructions, enhancing its efficiency in high-speed data stream environments. The code for our method is open-sourced.

CCS Concepts: • **Theory of computation → Sketching and sampling**.

Additional Key Words and Phrases: Data stream processing, approximate data structure, persistence, persistent items

ACM Reference Format:
Weihe Li. 2025. Pandora: An Efficient and Rapid Solution for Persistence-Based Tasks in High-Speed Data Streams. *Proc. ACM Manag. Data* 3, 1 (SIGMOD), Article 61 (February 2025), 26 pages. https://doi.org/10.1145/3709711

1 Introduction

Detecting items that exhibit specific patterns in high-velocity data streams poses a critical challenge in data mining applications. These streams, sourced from networks, system logs, and user behavior, often harbor valuable insights into system health [31, 55], security threats [2], user experience [52], business intelligence [37], and more. However, traditional deterministic analysis methods face difficulties in efficiently processing these data streams due to their high velocity and volume [6, 49]. Consequently, an emerging paradigm of probabilistic analysis has surfaced, emphasizing the utilization of approximate data structures (sketches) for efficient processing of massive data under constrained memory budgets [11, 17, 33, 48, 57, 60, 64].

Existing sketch-based work primarily focuses on data characteristics such as frequency [42, 49, 51, 58], cardinality [20, 30, 46, 50], and quantiles [15, 22, 24, 29]. Recently, another crucial data characteristic—persistence—has garnered significant attention [16, 18, 62]. Given a data stream

Author's Contact Information: Weihe Li, weihe.li@ed.ac.uk, University of Edinburgh, Edinburgh, Scotland, United Kingdom.

This work is licensed under a Creative Commons Attribution International 4.0 License.

© 2025 Copyright held by the owner/author(s).
ACM 2836-6573/2025/2-ART61
https://doi.org/10.1145/3709711

divided into , non-overlapping, consecutive time windows, the persistence of an item e is the number of distinct windows in which it appears. Persistence is often crucial for various practical applications, including anomaly traffic detection [21, 54], user behavior analysis [23], and click fraud detection [45]. For instance, persistent user behaviors, like consistently used features or repeatedly visited sections, indicate users' strong preferences and can guide developers in optimizing user interface and experience. Another instance involves automated bots repeatedly clicking on advertisements to boost advertiser revenue in pay-per-click online advertising systems [27].

There are two main detection tasks associated with persistence: (i) *persistence estimation*, which provides an approximate estimate of persistence for each item in the data stream, and (ii) *persistent item lookup*, which identifies items whose persistence exceeds a predetermined threshold. In addition to these tasks, there is a challenge that has been rarely considered: (iii) *mining persistent items with high frequency*. This task has significant practical applications in various domains. For instance, in financial security, detecting accounts or credit cards with persistent and frequent transaction patterns can aid in identifying fraudulent activities or money laundering schemes. In cybersecurity, identifying IP addresses that consistently and frequently send requests can facilitate early detection and mitigation of distributed denial-of-service (DDoS) attacks.

1.1 Limitations of Prior Art

While some work exists on persistence, primarily focusing on persistent item lookup, these methods can generally be categorized into three main types: sample-based [13, 32], coding-based [18], and sketch-based [33, 62]. Each method aims to balance memory usage, accuracy, and update speed, yet they encounter significant limitations, particularly under constrained memory conditions. Below, we briefly outline the key conceptual differences and limitations of these approaches:

(i) Sample-Based Approaches [13, 32]: These methods rely on sampling to track potential persistent items. Although they offer reduced memory usage by focusing on a subset of items, they often include too many non-persistent items, leading to memory inefficiency. The reliance on sampling also means accuracy suffers when memory is constrained, as the reduced sample size introduces larger estimation errors.

(ii) Coding-Based Approaches [18]: These methods encode every item observed in each time window to identify persistent items. They come with high memory overhead, especially when encoding a large number of non-persistent items. As the number of time windows increases, memory usage scales linearly, making these methods impractical for environments with limited memory. Additionally, they require computationally intensive operations, such as matrix multiplication for encoding and decoding, which significantly slow down updates, making them unsuitable for high-speed data streams.

(iii) Sketch-Based Approaches [33, 62]: Sketches leverage probabilistic data structures to estimate persistence with lower memory requirements and faster update speed. However, the eviction strategies in most existing sketch-based methods often misclassify non-persistent items as persistent during hash collisions, reducing accuracy, especially in memory-constrained environments.

1.2 Challenges

In practice, accurately and swiftly detecting persistence in high-speed data streams presents a complex and demanding task. This complexity arises from two primary factors:

(i) In real-world applications, small memory sizes (e.g., less than 64KB) are crucial in resource-constrained environments like embedded systems, IoT devices, and edge computing [53]. These systems have limited memory and processing power but are responsible for critical tasks such as network monitoring, traffic analysis, and real-time anomaly detection. In such scenarios, where memory efficiency is essential, traditional methods often struggle to maintain detection accuracy.

Using smaller memory portions (e.g., 16KB or 32KB) not only conserves resources but also improves efficiency with faster query time for retrieving persistent items. Achieving high accuracy under stringent resource constraints is particularly challenging, and a method that excels in these conditions demonstrates its robustness and suitability for deployment across diverse platforms with varying resource limitations.

(ii) As an item arrives within a time window, its persistence only increases by 1, regardless of how many times it appears. Consequently, the persistence value is typically much smaller than the item frequency. Due to the highly-skewed data distribution in practical scenarios, where most items have small persistence, the eviction of persistent items from buckets by non-persistent ones is more likely, resulting in low detection accuracy.

1.3 Our Solution - Pandora

To overcome these challenges, we propose a novel approach, Pandora, for diverse persistence-based detection tasks. Our method achieves high detection accuracy and fast processing speed simultaneously, even with limited memory resources. Technically, our method is based on a key insight: *the longer a tracked item is missing, the less likely it is to be a persistent item.* We refer to the number of time windows a recorded item is missing as its *inactivity degree.* To the best of our knowledge, this is the first study to employ the concept of inactivity degree for various detection tasks in high-speed data streams. During the update processing, when an incoming item arrives, Pandora adjusts the replacement probability dynamically in response to hash collisions, taking into account the inactivity degree. If the inactivity degree of the tracked item is high, the probability of the incoming item successfully replacing the tracked item in the bucket is elevated; conversely, if the inactivity degree is low, the probability is reduced. This approach facilitates easy eviction of items exhibiting low persistence from buckets, making space for potentially persistent items, thereby ensuring high memory efficiency and accurate lookups. The update procedure is straightforward, requiring no additional data structures, which contributes to its high update speed. The elegant design also enables further acceleration through the utilization of SIMD instructions, enhancing its update speed even further.

We rigorously derive the error bound of Pandora to establish its theoretical accuracy. Extensive trace-driven evaluations demonstrate Pandora's significant improvements in detection accuracy across various tasks compared to state-of-the-art methods. For persistence estimation, Pandora reduces the error by up to 96.2% compared to On-Off Sketch. In persistent item lookup, Pandora achieves the highest F1 score among considered baselines, improving accuracy by 57.65% over On-Off Sketch. Moreover, Pandora maintains high performance when identifying persistent items with high frequency, achieving an F1 score of 0.8 under tight 16KB memory constraints. Additionally, our method achieves the highest update speed in persistent item lookup, being 41.65% faster than the competitive On-Off Sketch. Furthermore, our method adapts easily to other frequency-based tasks while maintaining superiority. With the aid of SIMD instructions, we further enhance the update speed by 19.08%. The code for Pandora can be found at [1].

2 Problem Definition

(i) Data Stream: Given a high-velocity data stream $\mathcal{S}$ consisting of various items $\mathcal{S} = \{4_1 \cdot 4_2 \cdot \text{""} \cdot 4_=\}$, where each item 4_8 can be denoted as a key-value pair $(:_8 \cdot E_8)$. The key $:_8$ is the unique identifier of the item, and the value E_8 corresponds to a metric associated with the item. For instance, in the network domain, the key could represent a source-destination IP address pair, while the value could represent the item's persistence or frequency.

(ii) Persistence Estimation: For a data stream $\mathcal{S}$, we segment it into , consecutive time windows based on a specified interval. For example, in a network trace, a window can be defined as 2000