Elastic Sketch under Random Stationary Streams: Limiting Behavior and Near-Optimal Configuration

Younes Ben Mazziane[1], Vinay Kumar B. R.[2], and Othmane Marfoq[3]

[1]University of Avignon, LIA, Avignon, France, younes.ben-mazziane@univ-avignon.fr
[2]IIT Bombay, Mumbai, India, vinaykumar.br@iitb.ac.in
[3]Meta, New York, USA, omarfoq@meta.com

Abstract

`Elastic-Sketch` is a hash-based data structure for counting item's appearances in a data stream, and it has been empirically shown to achieve a better memory-accuracy trade-off compared to classical methods. This algorithm combines a *heavy block*, which aims to maintain exact counts for a small set of dynamically *elected* items, with a light block that implements `Count-Min Sketch` (`CM`) for summarizing the remaining traffic. The heavy block dynamics are governed by a hash function β that hashes items into m_1 buckets, and an *eviction threshold* λ, which controls how easily an elected item can be replaced. We show that the performance of `Elastic-Sketch` strongly depends on the stream characteristics and the choice of λ. Since optimal parameter choices depend on unknown stream properties, we analyze `Elastic-Sketch` under a *stationary random stream* model—a common assumption that captures the statistical regularities observed in real workloads. Formally, as the stream length goes to infinity, we derive closed-form expressions for the limiting distribution of the counters and the resulting expected counting error. These expressions are efficiently computable, enabling practical grid-based tuning of the heavy and `CM` blocks memory split (via m_1) and the eviction threshold λ. We further characterize the structure of the optimal eviction threshold, substantially reducing the search space and showing how this threshold depends on the arrival distribution. Extensive numerical simulations validate our asymptotic results on finite streams from the Zipf distribution.

1 Introduction

Streaming algorithms process massive amounts of rapidly arriving data under strict resource constraints. They must support very fast per-update processing while using memory sublinear in the stream length, and thus return approximate answers [27, 21, 7, 2]. Such requirements arise routinely in network monitoring and clickstream analytics [1, 14], and underpin large-scale data-processing systems. A common task is the detection of ϕ-heavy hitters, i.e., items whose frequency exceeds a fraction ϕ of the total mass in a given time window. Under the streaming model above, this task typically involves approximate counting using either counter-based methods or sketch-based methods. Counter-based methods, such as `Misra-Gries/Frequent` [24, 11, 16], and `Space Saving` [23], maintain explicit counts for a dynamically chosen subset of items. On the other hand, sketch-based methods such as `Count-Sketch` [7] and `Count-Min` *Sketch* (`CM`) [8] update a compact hash-indexed array and provide fast estimates for any queried item. Sketches can be viewed as hash-based random projections of the high-dimensional frequency vector, with accuracy governed by the size of the hash-indexed array.

`Elastic Sketch` [32] combines counter-based and sketch-based ideas and exhibits a strong empirical memory-accuracy trade-off across a range of streaming tasks. It splits the available memory into two blocks: a *heavy* block that aims to filter and explicitly count popular items, and a *light* `Count-Min Sketch` (`CM`) block that summarizes the remaining traffic. In the heavy block, a hash function β maps items to m_1 buckets, each storing and monitoring a single *elected* item that may change over time. Updates are governed by an integer parameter λ, that we refer to as the *eviction threshold*: once an item

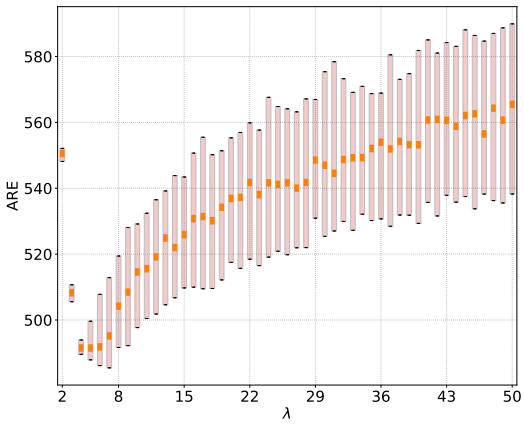


Figure 1: Average Relative Error (ARE) of `Elastic-Sketch` as a function of the eviction threshold λ, shown as box plots (boxes span Q1–Q3 and the center line indicates the median) over 100 runs, where each run is generated from an independent Zipf stream with skew parameter $\alpha = 1.2$; $m_1 = 50$, $n_{\mathcal{I}} = 2 \times 10^5$ items, stream length $\tau = 5 \times 10^5$, `CM` width 200.

is elected in a bucket, it remains so as long as its counter (scaled by λ) dominates the aggregate count of other items hashing to the same bucket. Upon an eviction from a bucket, the associated counters are reset, and the light block is updated to reflect the eviction. The estimated count of an item is obtained by adding its heavy-block counter (it is equal to 0 if the item is not tracked there) to its light-block sketch-based estimate.

`Elastic-Sketch`'s estimation accuracy is highly sensitive to the characteristics of the input stream, including both the item frequency distribution and the arrival order, as well as the choice of eviction threshold λ. For example, when $\lambda \to \infty$, no evictions occur in the heavy block, causing the first item in each bucket to remain permanently elected—making performance extremely sensitive to arrival order. Even for moderate values of λ, Figure 1 demonstrates substantial variability in counting error across different realizations of streams with identical frequency distributions, highlighting the algorithm's sensitivity to arrival patterns.

This variability presents a fundamental challenge for practical deployment. The algorithm must be configured (particularly the choice of λ) before observing the actual stream, yet the optimal configuration depends heavily on unknown stream properties. However, real-world workloads often exhibit predictable statistical regularities, such as Zipf-like frequency distributions commonly observed in web traffic [4, 6]. This suggests that, rather than optimizing for unknown worst-case scenarios, we should leverage these statistical patterns to guide algorithm configuration.

We therefore adopt a stationary random stream model where requests $\boldsymbol{R} = (R_t)_{t=1}^{\tau}$ are i.i.d. draws from a probability vector $\boldsymbol{p} = (p_i)_{i \in \mathcal{I}}$ over an item set $\mathcal{I}$ of size $n_{\mathcal{I}}$. This model captures the essential statistical properties of real workloads while remaining analytically tractable, enabling us to derive optimal parameter choices based on expected performance rather than pathological edge cases. Under this framework, we address the following questions:

1. *What is the memory-accuracy trade-off of* `Elastic-Sketch` *as a function of the vector* $\boldsymbol{p}$*?*
2. *How should one choose the memory split between the heavy and CM blocks (via* m_1*) and set the eviction threshold* λ *to minimize the expected counting error?*

1.1 Contributions

Our paper answers the above questions via three main contributions. First, we derive closed form expressions for the limiting distribution of the counters and the corresponding expected counting error. This permits efficient grid-based search for the optimal parameters (λ, m_1) minimizing the expected counting error. Second, we characterize the structure of the optimal eviction threshold for a fixed

β, substantially reducing the search space and showing how this threshold depends on the arrivals distribution $\boldsymbol{p}$. Third, we validate our results on finite streams drawn from a Zipf distribution. We detail each contribution and summarize the techniques employed.

Limiting behavior and near-optimal configuration Under the stationary random stream assumption, Theorem 1 derives closed-form expressions of the counters limiting distribution, as the stream length tends to infinity. It shows that the asymptotic behavior in the heavy block is bucket-dependent and falls into two regimes: (i) the elected item switches infinitely often, or (ii) some item becomes permanently elected, with the election probability admitting a closed-form expression. Corollary 1 then yields closed-form expressions of the expected counting error. It shows that the expected counting error decreases with the aggregate probabilities of permanently elected items. For a fixed realization of the hash function β, this expected error can be computed in $\mathcal{O}(n_{\mathcal{I}})$ time. This enables efficient comparisons across `Elastic-Sketch`'s configurations and near-optimal tuning of λ and m_1 via grid search.

The proof of Theorem 1 captures the evolution of each bucket via a discrete-time Markov chain that decomposes into asymmetric random-walk branches indexed by the currently elected item such that election switches correspond to transitions between branches. Recurrence versus transience of this chain yields regimes (i) and (ii) and provides explicit conditions for each.

Characterization of the optimal eviction threshold Theorem 2 shows that, for a fixed hash function β, the optimal eviction threshold $\lambda^*(\beta)$ that minimizes the expected counting error belongs to a finite candidate set of size at most m_1. It also shows that a near-optimal threshold that minimizes the sample average counting error over n_{samp} realizations of β can be chosen from at most $m_1\, n_{\text{samp}}$ candidate values. Finally, Theorem 3 provides probabilistic upper bounds on $\lambda^*(\beta)$ showing that, in most cases, the relevant candidates lie in a much smaller range than the worst-case bound $m_1\, n_{\text{samp}}$. Together, these results substantially reduce the cost of the grid search for tuning (λ, m_1).

The proof of Theorem 2 shows that, for a fixed β, the expected counting error as a function of λ is a piecewise continuous function and decreases on each piece. It then pinpoints the discontinuity values where the counting error can increase. The proof of Theorem 3 shows that the largest possible value of $\lambda^*(\beta)$ occurs when $\boldsymbol{p}$ is the uniform distribution. It also shows that $\lambda^*(\beta)$ in this case reduces to the maximum load in a *balls-and-bins* process, so classical results yield sharp high probability bounds.

Finite-time validation. For a fixed hash function β, we validate our asymptotic results on Zipf streams of length $\tau = 5 \times 10^5$ with $n_{\mathcal{I}} = 10^4$ items and skew parameter $\alpha \in \{0.8, 1.0, 1.2\}$. Figure 2 shows that the theoretical limits closely match the corresponding simulation estimates at finite τ. Moreover, the candidate set for $\lambda^*(\beta)$ identified in Theorem 2 is small in practice (less than 15 values) despite the worst-case upper bound of $m_1 = 200$. In the considered experimental setting, the asymptotically optimal eviction threshold $\lambda^*(\beta)$ is also optimal for the finite-time estimates. Moreover, the candidates set for the near-optimal eviction threshold, identified in Theorem 2, has less than 27 elements, far below the worst-case upper bound $m_1 n_{\text{samp}} = 2 \times 10^5$ with $n_{\text{samp}} = 10^3$.

1.2 Related work

`Elastic-Sketch`'s original paper [31] derives probabilistic bounds on the counting error that hold for a fixed stream. However, these bounds do not account for either the randomness of the hash function β nor the arrivals. As a result, it is unclear how (λ, m_1) can be configured given only the items' frequency profiles. In contrast, our analysis provides an average-case performance metric in terms of the frequency profile, which can be directly used to tune (λ, m_1). The authors in [9] characterize the memory accuracy trade-off of `CM` under a Zipf frequency profile with skew α. Specifically, they show that achieving ϵ error in count estimation, with `CM`, requires $\mathcal{O}(\epsilon^{-\min(1,1/\alpha)})$ memory. In `Elastic-Sketch`, existing `CM` trade-offs apply to the `CM` block only after replacing the original stream by the *filtered stream* obtained by removing updates absorbed by the heavy block. Hence, one needs to quantify the filtered-stream length

(or, equivalently, the fraction of updates absorbed). This is precisely the hard part: unlike `CM`, the heavy block is order dependent, so for a fixed frequency profile the absorbed mass can vary substantially across arrival orderings.

Our work contributes to a line of research that derives tractable performance characterizations for streaming algorithms, with the additional goal of enabling principled configuration in practice. For instance, [12] analyzes recycling Bloom filters, [22] studies `CM` with conservative updates, and [18] proposes methods to optimize and adjust on the fly the number of hash functions in sketching algorithms.

Finally, `Elastic-Sketch` was motivated by the fact that classical sketches struggle to adapt to line-rate processing on modern links. Moreover, a single sketch is often designed for a narrow family of queries, whereas network monitoring typically requires supporting a range of tasks, including per-item counts, frequency moments [3], distinct counts [13], entropy estimation [17], and change detection in traffic patterns. This has spurred a line of work—including `Elastic-Sketch`—that augments or redesigns sketching primitives to reduce per-packet processing cost and broaden functionality. Representative examples include `UnivMon` [20], which builds a universal monitoring approach capable of answering multiple streaming queries; `SketchVisor` [15], which introduces a fast path activated under high load; and `NitroSketch` [19], which lowers update cost by probabilistically updating only a subset of hashed counters rather than all of them.

1.3 Notation

Vectors and matrices are written in bold, stream-induced random variables in uppercase, sets in calligraphic font. For any integer $k \geq 1$, $[k] \triangleq \{1, 2, \ldots, k\}$, $\mathbb{N}$ designates the set of non-negative integers, $\mathbb{1}(\cdot)$ denotes the indicator function, and $\setminus$ denotes the set difference operator. Probabilities and expectations with respect to the probability space governing the infinite stream $\boldsymbol{R}$ (when $\tau \to \infty$) are written as $\Pr(\cdot)$ and $\mathbb{E}[\cdot]$, respectively. When we also account for the randomness of the hash functions $(h_\ell)_{\ell=1}^{d}$ and/or β, we write $\Pr_{\boldsymbol{h}}(\cdot)$, $\Pr_{\beta}(\cdot)$, and $\Pr_{\boldsymbol{h},\beta}(\cdot)$, and analogously for expectations. The abbreviations a.s., w.h.p., and i.i.d. denote almost surely, with high probability and independent and identically distributed, respectively.

1.4 Paper Outline

The rest of the paper is organized as follows: § 2 describes the `Elastic-Sketch` algorithm, introduces notation, and the adopted assumptions. § 3 presents the main results, § 4 validates them through numerical simulations, and § 5 provides the proof of Theorem 1. § 6 concludes the paper. The appendix contains additional proofs and technical details.

2 Problem Formulation

Consider a data stream $\boldsymbol{R} = (R_1, \ldots, R_\tau)$, of size τ, from a finite set $\mathcal{I}$ of size $n_{\mathcal{I}}$. For each item $i \in \mathcal{I}$ and time t, let $N_i(t)$ be the number of occurrences of i up to step t, and let $\boldsymbol{N}(t) \triangleq (N_i(t))_{i \in \mathcal{I}}$ be the corresponding count vector. We assume that each arrival is an i.i.d. draw from a probability distribution $\boldsymbol{p} = (p_i)_{i \in \mathcal{I}}$. Formally,

Assumption 2.1 (Random Stationary Stream)**.** *The random variables $(R_t)_{t=1}^{\tau}$ are i.i.d. such that the probability that $R_1 = i$, for any $i \in \mathcal{I}$, is equal to p_i, i.e., $\Pr(R_1 = i) = p_i$, and $p_i > 0$.*

As mentioned in the introduction, `Elastic-Sketch`'s performance depends strongly on the stream, making worst-case analysis of limited practical relevance and motivating an average-case study under a *distribution* over streams.

`Elastic-Sketch`, among other streaming algorithms, aims to answer queries related to the number of appearances of items in the stream, such as the detection of ϕ-heavy hitters and estimation of $\boldsymbol{N}(t)$'s norms [31]. It combines a *heavy* block, for monitoring popular items, with a Count-Min Sketch (`CM`) block. We describe the dynamics of each block, as specified in Alg. 1.

Algorithm 1 `Elastic-Sketch`

Input: Stream $\boldsymbol{R} = (R_t)_{t=1}^T$, hash functions $\beta, (h_\ell)_{\ell\in[d]}$, eviction threshold λ,
Output: Estimated count $\hat{\mathbf{N}}$

Initialize $\boldsymbol{V}^+(0) \leftarrow \mathbf{0}$, $\boldsymbol{V}^-(0) \leftarrow \mathbf{0}$, $\boldsymbol{S}(0) \leftarrow \mathbf{-1}$, $\boldsymbol{Y}(0) \leftarrow 0$
for $t = 1$ to T **do**
  $(\boldsymbol{S}(t), \boldsymbol{V}^+(t), \boldsymbol{V}^-(t)) \leftarrow (\boldsymbol{S}(t-1), \boldsymbol{V}^+(t-1), \boldsymbol{V}^-(t-1))$
  $\boldsymbol{Y}(t) \leftarrow \boldsymbol{Y}(t-1)$
  $b \leftarrow \beta(R_t)$
  if $S_b(t-1) = -1$ or $S_b(t-1) = R_t$ **then**
    $S_b(t) \leftarrow R_t$
    $V_b^+(t) \leftarrow V_b^+(t-1) + 1$ ▷ Track in the heavy block
  else
    if $\lambda V_b^+(t-1) - V_b^-(t-1) > 0$ **then**
      $S_b(t) \leftarrow S_b(t-1)$
      $V_b^-(t) \leftarrow V_b^-(t-1) + 1$
      $Y_{\ell,h_\ell(R_t)}(t) \leftarrow Y_{\ell,h_\ell(R_t)}(t-1) + 1, \forall l \in [d]$ ▷ Track in `CM`
    else
      $Y_{\ell,h_\ell(R_t)}(t) \leftarrow Y_{\ell,h_\ell(R_t)}(t-1) + V_b^+(t-1), \forall l \in [d]$ ▷ Insert into `CM`
      $S_b(t) \leftarrow R_t$ ▷ Replace bucket item
      $V_b^+(t) \leftarrow 1$
      $V_b^-(t) \leftarrow 0$
    end if
  end if
  $\hat{N}_i(t) \triangleq \mathbb{1}\left(i \in \boldsymbol{S}(t)\right) V_{\beta(i)}^+(t) + \min_{\ell\in[d]} Y_{\ell,h_\ell(i)}(t), \forall i \in \mathcal{I}$
end for

Heavy block. This block uses a parameter λ, called the *eviction threshold*, and maintains a set $\mathcal{B}$ of m_1 buckets indexed by a hash function $\beta : \mathcal{I} \mapsto \mathcal{B}$—in practice the hash function produces an integer in $[m_1]$, such that each integer identifies a bucket $b \in \mathcal{B}$. Each bucket b maintains three components: the currently *elected* item S_b, a counter V_b^+ that tracks S_b, and a counter V_b^- used in the eviction rule for S_b. Define $\mathcal{I}_b$ as the set of items hashing to b, i.e., $\mathcal{I}_b \triangleq \{i \in \mathcal{I} : \beta(i) = b\}$ and n_b as its size ($n_b \triangleq |\mathcal{I}_b|$).

The elected item for each bucket b changes dynamically. An epoch for b is the interval from the moment an item is elected as S_b until its eviction. Throughout the epoch, upon each arrival mapped to b ($\beta(R_t) = b$ for some t), two counters are updated: V_b^+ (incremented on occurrences of S_b) and V_b^- (incremented on occurrences from items in $\mathcal{I}_b \setminus \{S_b\}$). The elected item changes when an item in $\mathcal{I}_b \setminus \{S_b\}$ arrives while λV_b^+ is equal to V_b^-. Formally, at $t = 0$, $S_b(0) = -1$, modeling the fact that no item is monitored, $V_b^+(0) = 0$, and $V_b^-(0) = 0$. At each step $t \geq 1$, the algorithm computes $b(t) = \beta(R_t)$. Only the bucket $b(t)$ is updated at step t, i.e.,

$$(S_b(t), V_b^+(t), V_b^-(t)) = (S_b(t-1), V_b^+(t-1), V_b^-(t-1)),\ \forall b \in \mathcal{B} \setminus \{b(t)\}. \tag{1}$$

We distinguish three cases.

1. $S_{b(t)}(t-1) \in \{-1, R_t\}$, i.e., there is no currently monitored item in the bucket $b(t)$, or R_t is itself the monitored item.
2. $S_{b(t)}(t-1) \notin \{-1, R_t\}$, i.e., there is an item monitored in bucket $b(t)$, but it is not R_t, and $\lambda V_{b(t)}^+(t-1) > V_{b(t)}^-(t-1)$.
3. $S_{b(t)}(t-1) \notin \{-1, R_t\}$, and $\lambda V_{b(t)}^+(t-1) = V_{b(t)}^-(t-1)$.

The update of bucket $b(t)$ depends on these three cases. In case 1, $S_{b(t)}$ starts/keeps monitoring R_t, and $V_{b(t)}^+$ is incremented by 1. In case 2, the bucket $b(t)$ keeps monitoring $S_{b(t)}(t-1)$, and in this case it is

$V^-_{b(t)}$ that is incremented by 1 and not $V^+_{b(t)}$. In case 3, $S_{b(t)}$ changes value to R_t, and the counts in $V^+_{b(t)}$ and $V^-_{b(t)}$ are reset to 1 and 0, respectively. Note that by construction of the algorithm, it is always true that $\lambda V^+_b(t) - V^-_b(t) \geq 0$, and thus all cases are covered. Formally,

$$(S_{b(t)}(t), V^+_{b(t)}(t), V^-_{b(t)}(t)) = \begin{cases} (R_t, V^+_{b(t)}(t-1)+1, V^-_{b(t)}(t-1)), & \text{if case 1,} \\ (S_{b(t)}(t-1), V^+_{b(t)}(t-1), V^-_{b(t)}(t-1)+1), & \text{if case 2,} \\ (R_t, 1, 0), & \text{if case 3.} \end{cases} \tag{2}$$

It will be useful later to define, for each bucket b, a vector notation for bucket indexed processes S_b, V^+_b, and V^-_b, e.g., $\boldsymbol{S}(t) \triangleq (S_b(t))_{b\in\mathcal{B}}$, $\mathcal{E}_\beta(t)$ as the set of elected items at step t, i.e., $\mathcal{E}_\beta(t) \triangleq \{S_b(t) : b \in \mathcal{B}\}$, and $\mu_b \triangleq \sum_{i\in\mathcal{I}_b} p_i$. Further define for each item i,

$$\lambda_i(\beta) \triangleq \frac{\mu_{\beta(i)}}{p_i} - 1, \tag{3}$$

and $\lambda^{(k)}_b(\beta)$ as the k-th smallest value of the set $\{\lambda_i(\beta) : i \in \mathcal{I}_b\}$, for any $b \in \mathcal{B}$, e.g., $\lambda^{(1)}_b(\beta) = \min_{i\in\mathcal{I}_b} \lambda_i$. Similarly, we define $p^{(k)}_b$ and $N^{(k)}_b(t)$ as the k-th largest values of the sets $\{p_i : i \in \mathcal{I}_b\}$ and $\{N_i(t) : i \in \mathcal{I}_b\}$, respectively.

`CM` block. This block uses Count-Min Sketch (`CM`) [10] to summarize counts of items not currently monitored in the heavy block, i.e., items in $\mathcal{I} \setminus \{S_b(t) : b \in \mathcal{B}\}$. We denote the value of the counters matrix of `CM` within `Elastic-Sketch` as $\boldsymbol{Y}(t) = (Y_{\ell,c}(t))_{(r,c)\in[d]\times[m_2]}$ where d and m_2 are the number of rows and columns.

In the absence of the heavy block ($m_1 = 0$), `Elastic-Sketch` reduces to `CM`. In this case, we denote the counters matrix of the `CM` block as $\boldsymbol{Y}^{\texttt{CM}}(t)$. `CM` uses d hash functions, $h_\ell : \mathcal{I} \mapsto [m_2]$ for $l \in [d]$, to map items to counters. Initially, $Y^{\texttt{CM}}_{\ell,c}(0) = 0$, and at any step $t \geq 1$, all counters associated to R_t, $(l, h_\ell(R_t))^d_{\ell=1}$, are incremented,

$$Y^{\texttt{CM}}_{\ell,c}(t) = Y^{\texttt{CM}}_{\ell,c}(t-1) + \mathbb{1}\left(h_\ell(R_t) = c\right). \tag{4}$$

On the other hand, when $m_1 > 0$, the update of $\boldsymbol{Y}$ depends on the three cases distinguished for the heavy block. The matrix $\boldsymbol{Y}$ is not updated on hits to the currently elected item at bucket $b(t)$ (case 1), but it is updated on arrivals of items mapped to $b(t)$ that are not $S_{b(t)}(t-1)$. In case 2, R_t is added to the sketch with value 1. Upon an eviction of the elected item at bucket $b(t)$ (case 3), the epoch ends: $V^+_b(t-1)$ holds the evicted item's total during the epoch, so this amount is added to the sketch. Formally, for any counter (l, c),

$$Y_{\ell,c}(t) = \begin{cases} Y_{\ell,c}(t-1), & \text{if case 1,} \\ Y_{\ell,c}(t-1) + \mathbb{1}\left(h_\ell(R_t) = c\right), & \text{if case 2,} \\ Y_{\ell,c}(t-1) + \mathbb{1}\left(h_\ell(R_t) = c\right) \cdot V^+_{b(t)}(t-1), & \text{if case 3.} \end{cases} \tag{5}$$

Count estimation. In `CM`, at any step t, the estimate for item i is the minimum of its d counters, namely $\min_{\ell\in[d]} Y^{\texttt{CM}}_{\ell,h_\ell(i)}(t)$, since each cell $(r, h_r(i))$ overestimates $N_i(t)$ and the minimum is the tightest. In Elastic Sketch, the same rule applies to items not currently monitored in the heavy block. However, if an item i is currently elected at some $b \in \mathcal{B}$, arrivals of i during the current epoch do not update the sketch; instead, they accumulate in $V^+_b(t)$. To preserve overestimation and account for this, `Elastic-Sketch` adds $V^+_b(t)$ to the sketch-based estimate. In summary, the estimation count for item i in `Elastic-Sketch` at step t, denoted $\hat{N}_i(t)$, is given by,

$$\hat{N}_i(t) \triangleq \mathbb{1}\left(i \in \mathcal{E}_\beta(t)\right) V^+_{\beta(i)}(t) + \min_{\ell\in[d]} Y_{\ell,h_\ell(i)}(t). \tag{6}$$

`Elastic-Sketch` is a randomized algorithm, due to the random choice of the hash functions β and $(h_\ell)^d_{\ell=1}$.

Assumption 2.2 (Ideal Hash Functions Model)**.** *For any $\ell \in [d]$, the sets $(\beta(i))_{i\in\mathcal{I}}$ and $(h_\ell(i))_{i\in\mathcal{I}}$ are i.i.d. uniform random variables in $[m_1]$ and $[m_2]$ respectively.*

While constructing hash functions that satisfy Assumption 2.2 can be costly, such idealized models have been shown to accurately predict the performance of algorithms using practical hash functions with weaker independence guarantees [26], and are widely adopted in the analysis of hash-based data structures and algorithms [5, 25].

Performance metric. Let $\mathrm{Err}_i(t)$ be the difference between the estimation and the true count of item i at step t, i.e., $\mathrm{Err}_i(t) = \hat{N}_i(t) - N_i(t)$. If an item i_0 is absent up to time t ($R_s \neq i$ for all $s \leq t$), then $\mathrm{Err}_{i_0}(t) = \min_{\ell\in[d]} Y_{\ell,h_\ell(i_0)}$. Under Assumption 2.2, $\{h_\ell(i_0)\}_{\ell\in[d]}$ are independent of $\boldsymbol{Y}(t)$, whose randomness is determined by $\{h_\ell(R_s)\}_{s\leq t,\, l\in[d]}$ and β. Hence $\mathrm{Err}_{i_0}(t)$ is identically distributed as the minimum of d row-wise uniformly sampled counters. We define the counting error $\mathrm{Err}_{(0)}(t)$ as follows,

$$\mathrm{Err}_{(0)}(t) \triangleq \min_{\ell\in[d]} Y_{\ell,X_\ell}(t), \tag{7}$$

where $(X_\ell)_{\ell\in[d]}$ are i.i.d. uniform random variables over $[w]$. It is clear then that, for any item i_0 absent from the stream, $\mathrm{Err}_{i_0}(t)$ is identically distributed to $\mathrm{Err}_{(0)}(t)$. Moreover, taking $X_\ell = h_\ell(i)$ for any item i, yields that, $\mathrm{Err}_{(0)}(t) - \mathrm{Err}_i(t) \geq N_i(t) - V^+_{\beta(i)} \geq 0$. Thus $\mathrm{Err}_{(0)}$ *stochastically dominates* Err_i for any i, i.e.,

$$\Pr\left(\mathrm{Err}_i(t) \geq x\right) \;\leq\; \Pr\left(\mathrm{Err}_{(0)}(t) \geq x\right),\ \forall x, t. \tag{8}$$

The quantity $\mathrm{Err}_{(0)}$ provides an upper bound on the estimation error for any item, and thus it can be used to bound standard performance metrics, such as the *average relative error* [32] and the expected false positive rate in the detection of ϕ-heavy-hitters.

The next section derives the limiting distribution of $\mathrm{Err}_{(0)}(t)/t$ under the random stationary stream model, thereby quantifying the memory-accuracy trade-off of `Elastic-Sketch` and yielding practical guidelines for tuning λ and m_1.

3 Main Results

This section is organized as follows. Section 3.1 derives the limiting distribution of the counters and the resulting asymptotic counting error. These expressions yield efficient numerical procedures for near-optimal tuning of (λ, m_1) via a grid-search. Section 3.2 further characterizes the optimal eviction threshold that minimizes the expected limiting counting error. This helps reduce the search space and offers insights about the optimal threshold for the arrival distribution $\boldsymbol{p}$.

3.1 Limiting distribution of the counters

As highlighted in Fig. 1, `Elastic-Sketch`'s strong sensitivity to the stream ordering is primarily due to the heavy block. Indeed, when the heavy block is absent ($m_1 = 0$), `Elastic-Sketch` reduces to `CM`, and under the random stationary stream model (Assumption 2.1), the *Strong Law of Large Numbers* yields $\frac{1}{t}Y^{\texttt{CM}}_{\ell,c}(t) \xrightarrow{a.s.} \sum_{i\in\mathcal{I}} \mathbb{1}(h_\ell(i) = c)p_i$, when t tends to infinity. Thus, all stream realizations regardless of ordering, lead to the same asymptotic counter values.

Unfortunately, previous analysis of `Elastic-Sketch` fails to capture its strong dependency on the stream. Indeed, [32, Thm. 4] derives a probabilistic upper bound on Err_i for a fixed stream, over the randomness of the hash functions $(h_\ell)_{\ell=1}^d$. The bound is linear in $t - V^+_{\mathcal{B}}(t)$, where $V^+_{\mathcal{B}}(t) \triangleq \sum_{b\in\mathcal{B}} V^+_b(t)$. However, this offers limited insight into *average* performance because, even among streams with the same final count vector, $V^+_{\mathcal{B}}(t)$ can vary from m_1 up to $\sum_{b\in\mathcal{B}} N^{(1)}_b(t)$. The minimum occurs when the arrivals of popular items are dispersed, triggering frequent evictions of items from the heavy block, and resets of V^+_b (see (2)). The maximum occurs when they arrive early in bursts, so V^+_b accumulates with time and there are no evictions.

To gain insight into the average performance of `Elastic-Sketch`, we study the asymptotic distribution of $S_b(t)$, $V_b^+(t)/t$, and $Y_{\ell,c}(t)/t$, under the random stationary stream assumption. In order to describe our result we introduce a few notations and definitions. Define $\mathcal{B}_\beta^+(\lambda)$ as the set of buckets b where either $\lambda_i(\beta)$ is strictly smaller than λ for any item i with $\beta(i) = b$, or for which there is a single item hashing to it, i.e., $n_b = 1$. Likewise, let $\mathcal{B}_\beta^0(\lambda)$ be the set of buckets b with $n_b = 0$, and $\mathcal{B}_\beta^-(\lambda)$ be the remaining buckets. Formally,

$$\mathcal{B}_\beta^0(\lambda) \triangleq \{b \in \mathcal{B} : \mathcal{I}_b = \emptyset\},\ \mathcal{B}_\beta^+(\lambda) \triangleq \left\{b \in \mathcal{B} :\ \lambda_b^{(1)}(\beta) < \lambda,\ \text{or } n_b = 1\right\},$$
$$\mathcal{B}_\beta^-(\lambda) \triangleq \mathcal{B} \setminus \left(\mathcal{B}_\beta^+(\lambda) \cup \mathcal{B}_\beta^0(\lambda)\right). \tag{9}$$

Let $\phi : [0,1] \times [1,+\infty) \mapsto \mathbb{R}^+$ be defined as $\phi(x,\lambda) \triangleq \frac{x^{\lambda+1}-1}{x-1}$ and let

$$r(\lambda, z) \triangleq \begin{cases} 1, & \text{if } \lambda \le z-1, \\ \text{root of } \phi(\cdot,\lambda) - z \text{ in } [0,1], & \text{otherwise,} \end{cases}. \tag{10}$$

Define the functions $w : [1,+\infty) \times [1,+\infty] \mapsto \mathbb{R}^+$ and $w_{i,\beta} : [1,+\infty) \mapsto \mathbb{R}^+$ as,

$$w(\lambda, z) \triangleq \left(1 - (r(\lambda,z))^\lambda\right), \qquad \text{and} \qquad w_{i,\beta}(\lambda) \triangleq p_i w\left(\lambda, \frac{\mu_{\beta(i)}}{p_i}\right). \tag{11}$$

Observe that $w_{i,\beta}(\lambda) > 0$ for any $\lambda > -1 + \mu_{\beta(i)}/p_i = \lambda_i(\beta)$.

Theorem 1 asserts that for any bucket $b \in \mathcal{B}_\beta^+(\lambda)$, an item $i \in \mathcal{I}_b$ eventually becomes permanently elected, whereas for any bucket $b \in \mathcal{B}_\beta^-(\lambda)$, S_b switches states infinitely often. Define

$$S_b^\infty \triangleq \begin{cases} \lim_{t\to\infty} S_b(t) & \text{for } b \in \mathcal{B}_\beta^+(\lambda), \\ -1 & \text{for } b \in \mathcal{B}_\beta^-(\lambda) \cup \mathcal{B}_\beta^0(\lambda). \end{cases} \tag{12}$$

Additionally, let $p_{-1} \triangleq 0$.

The theorem also establishes that for any item $i \in \mathcal{I}$ the probability that $S_b^\infty = i$, is equal to $a_{i,\beta}(\lambda)$, where $a_{i,\beta} : (0,+\infty) \mapsto \mathbb{R}$ is a piecewise function defined as,

$$a_{i,\beta}(\lambda) \triangleq \begin{cases} 1, & \text{if } n_{\beta(i)} = 1, \\ 0, & \text{if } \lambda \in (0, \lambda_i(\beta)] \text{ and } n_{\beta(i)} \ge 2, \\ \frac{w_{i,\beta}(\lambda)}{\sum_{j:\ \beta(j)=\beta(i)} w_{j,\beta}(\lambda)}, & \text{else.} \end{cases} \tag{13}$$

Theorem 1 (Counters limiting distribution)**.** *Under Assumption 2.1,*

1. *For any bucket $b \in \mathcal{B}_\beta^-(\lambda)$, $(S_b(t))_t$ switches state infinitely often, and for any bucket $b \in \mathcal{B}_\beta^+(\lambda)$, $S_b(t)$ converges a.s. in t, such that for any item $i \in \mathcal{I}_b$,*

$$\Pr\left(S_b^\infty = i\right) = a_{i,\beta}(\lambda), \tag{14}$$

   where $a_{i,\beta}(\lambda)$ is defined in (13)*.*

2. *For any bucket $b \in \mathcal{B}$, the counter V_b^+ satisfies,*

$$\frac{V_b^+(t)}{t} \xrightarrow[t\to\infty]{a.s.} \mathbb{1}\left(b \in \mathcal{B}_\beta^+(\lambda)\right) p_{S_b^\infty}. \tag{15}$$

3. For any cell (ℓ, c) in the `CM` *block, the counter $Y_{\ell,c}$ satisfies,*

$$\frac{Y_{\ell,c}(t)}{t} \xrightarrow[t\to\infty]{a.s.} \sum_{i\in\mathcal{I}} \mathbb{1}\left(h_\ell(i)=c\right) p_i - \sum_{b\in\mathcal{B}^+_\beta(\lambda)} \mathbb{1}\left(h_\ell(S_b^\infty=c)\right) p_{S_b^\infty}. \tag{16}$$

Sketch of the proof. To prove item 1 of the theorem, we capture the evolution of each bucket as a discrete-time Markov chain $M_b(t) \triangleq (S_b(t), U_b(t))$ such that $U_b(t) \triangleq \lambda V_b^+(t) - V_b^-(t)$. The chain decomposes into asymmetric random-walk branches indexed by the currently elected item, and election switches correspond to transitions between branches. The long-run stability of the elected item is then characterized by the transience/recurrence of this chain: recurrence leads to infinitely many switches ($b \in \mathcal{B}^-_\beta$), whereas absorption in a branch yields a permanently elected item ($b \in \mathcal{B}^+_\beta$). In the latter case, computing the branch absorption probabilities in M_b yields (14).

To prove items 2 and 3 of the theorem, we first establish in Lemma 5 (Section 5) a finite-time characterization of the counter values in terms of the count vector $\boldsymbol{N}(t)$ and the last eviction time in each bucket. We then combine this characterization with the asymptotic behavior of $(S_b(t))$ to derive the corresponding asymptotic results for the counters.

The detailed proof is presented in Section 5. □

Algorithm 2 Pseudo code for the computation of $g_\beta(\lambda)$

Input: Items set $\mathcal{I}$, probabilities vector $\boldsymbol{p}$, hash function β, eviction threshold λ
Output: $g_\beta(\lambda)$

```
Build bucket lists (I_b)_{b∈B} by hashing each i ∈ I                ▷ O(n_I)
Compute μ_b ← Σ_{i∈I_b} p_i for all b ∈ B                           ▷ O(n_I)
Compute λ_b^(1)(β) for all b ∈ B                                    ▷ O(n_I)
g ← 0
for b ∈ B : (n_b ≥ 1 and λ_b^(1)(β) < λ) do                         ▷ O(n_b) per bucket
    Compute w_{i,β}(λ) for all i ∈ I_b using (11)–(10)
    s_p ← Σ_{i∈I_b} w_{i,β}(λ) p_i
    s ← Σ_{i∈I_b} w_{i,β}(λ)
    g ← g + s_p/s
end for
return g                                                            ▷ Time complexity O(n_I)
```

When the `CM` block uses a single hash function, i.e., $d = 1$, Corollary 1 characterizes the expected limiting counting error, $\overline{\mathrm{Err}}^\infty_{(0)} \triangleq \lim_{t\to\infty} \min_{\ell\in[d]} Y_{\ell,X_\ell}(t)/t$ in terms of the function $g_\beta : [1, +\infty) \to \mathbb{R}^+$, defined as,

$$g_\beta(\lambda) \triangleq \sum_{b\in\mathcal{B}^+_\beta(\lambda)} g_{b,\beta}(\lambda), \quad \text{where} \quad g_{b,\beta}(\lambda) \triangleq \sum_{i\in\mathcal{I}_b} a_{i,\beta}(\lambda)\, p_i. \tag{17}$$

Corollary 1 (Expected Limiting Counting Error)**.** *Under Assumption 2.1 and when $d = 1$, the following holds,*

$$\mathbb{E}_{\boldsymbol{h},\beta}\left[\overline{\mathrm{Err}}^\infty_{(0)}\right] = \frac{1}{m_2}\left(1 - \mathbb{E}_\beta\left[g_\beta(\lambda)\right]\right). \tag{18}$$

For any $d \geq 1$, the same right-hand side remains a valid upper bound, since `CM` estimate is a minimum over d rows and thus is upper-bounded by any single-row estimate. Importantly, the choice $d = 1$ aligns with implementation considerations in [32, Sec. 4], where it is recommended to favor throughput over

marginal accuracy gains. In the absence of the heavy block, the corresponding term equals $1/m_2$. Hence, the quantity $(1 - \mathbb{E}_\beta[g_\beta(\lambda)]) \in [0,1]$ quantifies the average reduction relative to the baseline $1/m_2$.

Given an arrival distribution $\boldsymbol{p}$, Corollary 1 provides an explicit asymptotic objective for evaluating and tuning `Elastic-Sketch`. It enables direct comparison of configurations jointly in the eviction threshold λ and the memory split between the heavy and `CM` blocks, parametrized by m_1 and m_2. A near-optimal configuration for $\boldsymbol{p}$ can then be obtained numerically by grid searching over (λ, m_1, m_2), and maximizing the Monte Carlo estimate,

$$\widehat{\mathbb{E}}_\beta^{n_{\text{samp}}}[g_\beta(\lambda)] \triangleq \frac{1}{n_{\text{samp}}} \sum_{k=1}^{n_{\text{samp}}} g_{\beta_k}(\lambda), \tag{19}$$

where the hash functions β_k are obtained by varying the hash seed and n_{samp} is the total number of seeds. As n_{samp} grows, (19) converges to $\mathbb{E}_\beta[g_\beta(\lambda)]$, yielding the configuration that minimizes the expected limiting counting error. Note that even under an ideal hashing model such as Assumption 2.2, computing $\mathbb{E}_\beta[g_\beta(\lambda)]$ exactly is intractable because of the large space of hash functions and the non-linearity of the function g_β. A Monte Carlo approach as described above is more suitable since it estimates performance under the hash family used in practice, via different seeds, rather than under an idealized hashing model.

The brute-force tuning of `Elastic-Sketch` over a grid of configurations (λ, m_1) has time complexity $\mathcal{O}(n_{\text{grid}}\, n_{\text{samp}}\, n_{\mathcal{I}})$, where n_{grid} is the number of tested couples. This follows because Theorem 1 provides closed form expressions for the election probabilities $a_{i,\beta}(\lambda)$, yielding an $\mathcal{O}(n_I)$ time computation of $g_\beta(\lambda)$, as shown in Alg. 2. While under a fixed memory budget, the feasible values of m_1 are bounded, λ can in principle be arbitrarily large and may therefore dominate n_{grid}. To address this, in the next section, we derive properties of the optimal eviction threshold for a given β. These properties restrict the search range for λ.

3.2 Optimal eviction threshold

For a fixed value of m_1, define λ^* as the smallest eviction threshold that maximizes the expectation of the function $g_\beta(\lambda)$, and equivalently minimizes the expected limiting counting error when $d = 1$, as shown in Corollary 1. We settle instead for an approximation of λ^*, denoted $\widehat{\lambda}^*$, and defined as,

$$\widehat{\lambda}^* \triangleq \min \arg\max_{\lambda \in \mathbb{N}\setminus\{0\}} \widehat{\mathbb{E}}_\beta^{n_{\text{samp}}}[g_\beta(\lambda)]\,. \tag{20}$$

Similarly, we define $\lambda^*(\beta)$ as the smallest optimal eviction threshold for a fixed realization of the hash function β, i.e.,

$$\lambda^*(\beta) \triangleq \min \arg\max_{\lambda \in \mathbb{N}\setminus\{0\}} g_\beta(\lambda). \tag{21}$$

We further define the set $\Lambda(\beta)$ as follows,

$$\Lambda(\beta) \triangleq \left\{ \lfloor \lambda_b^{(1)}(\beta) \rfloor + 1 :\ b \in \mathcal{B} \right\}. \tag{22}$$

Theorem 2 narrows down the search for $\lambda^*(\beta)$ and $\widehat{\lambda}^*$ to at most m_1 and $m_1\, n_{\text{samp}}$ candidate values, respectively. The proof uses Lemma 1.

Lemma 1. *For any bucket $b \in \mathcal{B}$ The function $g_{b,\beta}$ is decreasing over the interval $\left(\lambda_b^{(1)}(\beta), +\infty\right)$.*

Proof. The proof is presented in Appendix B. □

Theorem 2 (Candidate values of $\lambda^*(\beta)$ and $\widehat{\lambda}^*$)**.** *The following holds,*

1. *For a fixed β, the optimal eviction threshold $\lambda^*(\beta)$ satisfies,*

$$\lambda^*(\beta) \in \Lambda(\beta). \tag{23}$$

2. *The near-optimal eviction threshold $\widehat{\lambda}^*$ satisfies,*

$$\widehat{\lambda}^* \in \bigcup_{k=1}^{n_{samp}} \Lambda(\beta_k). \tag{24}$$

Proof. For each bucket b, from the definition of the absorption probabilities $a_{i,\beta}(\lambda)$ in (13), we deduce that,

$$g_{b,\beta}(\lambda) = \begin{cases} 0, & \text{if } \lambda \leq \lambda_b^{(1)}(\beta), \\ p_b^{(1)}, & \text{if } \lambda \in \left(\lambda_b^{(1)}(\beta), \lambda_b^{(2)}(\beta)\right), \end{cases} \tag{25}$$

and for any $\lambda > \lambda_b^{(1)}(\beta)$, $g_{b,\beta}(\lambda)$ is a convex combination of the probabilities $(p_i)_{i:\,\beta(i)=b}$, and thus $p_b^{(1)}$ is its maximum value. Lemma 1 shows that $g_{b,\beta}$ is decreasing over $\left(\lambda_b^{(1)}(\beta), +\infty\right)$. Consequently, $g_\beta(\lambda) = \sum_{b\in\mathcal{B}} g_{b,\beta}(\lambda)$ can only increase when λ crosses one of the points $\lambda_b^{(1)}(\beta)$, $b \in \mathcal{B}$; between such points, g is decreasing. With the restriction of λ being an integer, we deduce that the smallest value of λ that maximizes g lies in the set $\Lambda(\beta)$.

The near optimal eviction threshold $\widehat{\lambda}^*$ is a maximizer of the function $\sum_{k=1}^{n_{samp}} \sum_{b\in\mathcal{B}} g_{b,\beta_k}(\lambda)$ (see (19) and (20)). Using Lemma 1, each function g_{b,β_k} only increases at the discontinuity points $\lambda_b^{(1)}(\beta_k)$ and decreases afterwards. Thus, using the same arguments as in the proof of the first item of the theorem, we deduce the second item. This finishes the proof.

□

Using Theorem 2, one can compute $\lambda^*(\beta)$ and $\widehat{\lambda}^*$ in $\mathcal{O}(m_1\, n_{\mathcal{I}})$ and $\mathcal{O}(m_1\, n_{\mathcal{I}}\, n_{\text{samp}})$ time, respectively. Indeed, Alg. 2 evaluates g_β for a fixed λ in $\mathcal{O}(n_{\mathcal{I}})$ time, and $\lambda^*(\beta)$ is a maximizer of g_β over the set $\Lambda(\beta)$ which has at most m_1 values. In practice, the cost can be substantially smaller because we only consider integer λ and many candidates coincide across buckets and across samples. In particular, in the experiments of Section 4, we observe that $|\Lambda(\beta)| \leq 15$ even though the worst case bound is $m_1 = 200$.

Next, we characterize the $\lambda^*(\beta)$ when $\boldsymbol{p}$ is the uniform distribution, denoted $\lambda^*_{\text{unif}}(\beta)$. This characterization provides insights on how the optimal eviction threshold depends on the problem parameters. It also enables to derive probabilistic upper bounds on $\widehat{\lambda}^*$. These bounds can be used to further support that the empirical observation $|\Lambda(\beta)| \ll m_1$ is not an artifact of the experiments in Section 4.

Lemma 2 shows that $\lambda^*_{\text{unif}}(\beta)$ coincides with the maximum load in the classical *balls and bins* process with $n_{\mathcal{I}}$ balls and m_1 bins. It also shows that, among all distributions with support size equal to $n_{\mathcal{I}}$, $\lambda^*(\beta) \leq \lambda^*_{\text{unif}}(\beta)$.

Lemma 2 (Eviction threshold for uniform arrivals)**.** *When $\boldsymbol{p}$ is the uniform distribution, the following holds,*

1. *The optimal eviction threshold for a fixed β corresponds to the largest number of collisions in the heavy block, i.e., $\lambda^*_{unif}(\beta) = \max_{b\in\mathcal{B}} n_b$.*
2. *$\lambda^*(\beta) \leq \lambda^*_{unif}(\beta)$.*

Proof. When $\boldsymbol{p}$ is the uniform distribution, $w_{i,\beta}(\lambda)$ is constant across items from the same bucket, i.e., $w_{i,\beta}(\lambda) = w_{j,\beta}(\lambda)$ whenever $\beta(i) = \beta(j)$. It follows that, $a_{i,\beta}(\lambda) = 1/n_{\beta(i)}$, and thus,

$$g_{b,\beta}(\lambda) = \begin{cases} 0, & \text{if } \lambda \leq \lambda_b^{(1)}(\beta), \\ \frac{1}{n_{\mathcal{I}}}, & \text{if } \lambda \in \left(\lambda_b^{(1)}(\beta), +\infty\right). \end{cases} \tag{26}$$

Note that when $\boldsymbol{p}$ is uniform, $\lambda_b^{(1)}(\beta) = n_b - 1$, and thus, $\lambda^*_{\text{unif}}(\beta) = \max_{b\in\mathcal{B}} n_b$. Moreover, using Theorem 2, we can write,

$$\lambda_b^{(1)}(\beta) = \frac{\sum_{j\in\mathcal{I}_b} p_j}{p_b^{(1)}} - 1 \leq n_b - 1 \implies \lambda^*(\beta) \leq \max_{b\in\mathcal{B}} n_b = \lambda^*_{\text{unif}}(\beta). \tag{27}$$

This finishes the proof.

□

Under Assumption 2.2, and in a regime where $n_{\mathcal{I}} \gg m_1$ such that $m_1 \to \infty$, combining Lemma 2 with the sharp high-probability bounds on the maximum load in a balls and bins process [28, Thm. 1], justifies the following approximation,

$$\lambda^*_{\text{unif}}(\beta) \approx \frac{n_{\mathcal{I}}}{m_1} + \Theta\left(\sqrt{\frac{n_{\mathcal{I}} \ln(m_1)}{m_1}}\right). \tag{28}$$

Using Lemma 2, Theorem 3 derives then a probabilistic upper bounds on $\lambda^*(\beta)$ that hold for any $(m_1, n_{\mathcal{I}})$. These bounds can then be used to derive probabilistic bounds, under Assumption 2.2, on $\widehat{\lambda}^*$, as follows,

$$\Pr_{\{\beta_k\}_{k=1}^{n_{\text{samp}}}}\left(\widehat{\lambda}^* \leq x\right) \;\geq\; \Pr_{\{\beta_k\}_{k=1}^{n_{\text{samp}}}}\left(\max_{k\in[n_{\text{samp}}]} \lambda^*(\beta_k) \leq x\right) \;=\; \left(\Pr_\beta\left(\lambda^*(\beta) \leq x\right)\right)^{n_{\text{samp}}}. \tag{29}$$

Theorem 3 (High-probability upper bound on $\lambda^*(\beta)$)**.** *Under Assumptions 2.2, for any m_1, $n_{\mathcal{I}}$, and $\delta \in (0,1)$, w.p. at least $1-\delta$, the following holds,*

$$\lambda^*(\beta) \leq \frac{n_{\mathcal{I}}}{m_1} + \sqrt{\frac{2n_{\mathcal{I}}}{m_1}\ln\left(\frac{m_1}{\delta}\right)} + \frac{1}{3}\ln\left(\frac{m_1}{\delta}\right). \tag{30}$$

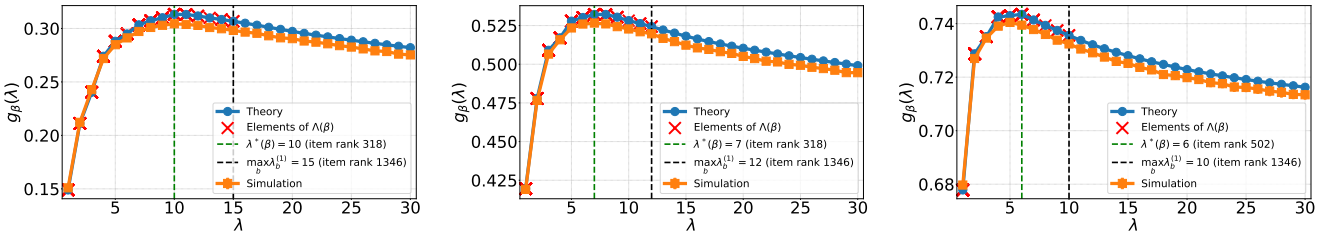
Proof. We show that the relation holds for $\lambda^*_{\text{unif}}(\beta)$. Under Assumption 2.2, n_b is the sum of $n_{\mathcal{I}}$ i.i.d. random variables that each indicates whether item i hashes to bucket b or not. Each of these random variables have a mean and variance equal to $\frac{1}{m_1}$ and $\frac{1}{m_1}(1-\frac{1}{m_1}) \leq \frac{1}{m_1}$, respectively. Thus, applying *Bernstein*'s inequality [29, Corollary 7.3] yields that, w.p. at least $1 - \frac{\delta}{m_1}$,

$$n_b \leq \frac{n_{\mathcal{I}}}{m_1} + \sqrt{\frac{2n_{\mathcal{I}}\ln(m_1/\delta)}{m_1}} + \frac{1}{3}\ln(m_1/\delta). \tag{31}$$

Moreover for any x, $\Pr\left(\max_{b\in\mathcal{B}} n_b \leq x\right) = 1 - \Pr\left(\exists b \in \mathcal{B} :\; n_b \geq x\right) \geq 1 - m_1 \Pr\left(n_b \geq x\right)$. From Lemma 2, we obtain the target result. This finishes the proof.

□

The high-probability bounds above on $\lambda^*(\beta)$ are distribution-agnostic and therefore do not capture how the skew of $\boldsymbol{p}$ affects $\lambda^*(\beta)$. A more $\boldsymbol{p}$-sensitive route is presented in Appendix C.



(a) $\alpha = 0.8$ (b) $\alpha = 1.0$ (c) $\alpha = 1.2$

Figure 2: Estimation of $\mathbb{E}\left[\overline{V_{\mathcal{B}}}(\tau)\right]$ via $g_\beta(\lambda)$ for Zipf request distributions with different skew parameters α, $n_{\mathcal{I}} = 10^4$, $\tau = 5 \times 10^5$, $n_{\text{runs}} = 100$, $m_1 = 200$.

4 Simulations

In this section, we validate our results on arrivals from the Zipf distribution with skew parameter α, i.e., $\mathcal{I} = [n_{\mathcal{I}}]$ and $p_i \propto 1/i^\alpha$. In practice, `Elastic-Sketch` is run on finite streams, or over sliding windows. We therefore evaluate how accurately the asymptotic characterizations in Theorem 1 and Corollary 1 predict the finite-τ expected error. We also assess how well the eviction threshold tuned to minimize the asymptotic expected error performs when the objective is the finite time expected error.

Similarly to Corollary 1, one can use Lemma 5 (Section 5.3) to obtain a finite time expression for the expected error,

$$\frac{1}{\tau}\mathbb{E}_{\boldsymbol{h}}\left[\text{Err}_{(0)}(\tau)\right] = \frac{1}{m_2}\left(1 - \mathbb{E}\left[\overline{V_{\mathcal{B}}}(\tau)\right]\right) : \quad \overline{V_{\mathcal{B}}}(\tau) \triangleq \frac{1}{\tau}\sum_{b\in\mathcal{B}} V_b^+(\tau). \tag{32}$$

Hence, minimizing the finite time expected error is equivalent to maximizing $\overline{V_{\mathcal{B}}}(\tau)$. We therefore validate our approach through the metric $\overline{V_{\mathcal{B}}}(\tau)$.

Figure 2 reports estimates of $\mathbb{E}\left[\overline{V_{\mathcal{B}}}(\tau)\right]$ as a function of λ for the same hash function β. The curve labeled "**Simulations**" is obtained by averaging $\overline{V_{\mathcal{B}}}(\tau)$ over $n_{\text{runs}} = 100$ independent streams drawn form a Zipf distribution with skew parameter $\alpha \in \{0.8, 1.0, 1.2\}$. The curve labeled "**Theory**" corresponds to the asymptotic prediction $\mathbb{E}\left[\overline{V_{\mathcal{B}}}(\tau)\right] \approx g_\beta(\lambda)$ provided by Theorem 1 (for large τ). The figure also shows the candidates set $\Lambda(\beta)$ from Theorem 2. In particular, it highlights as vertical lines (i) the maximizer of $g_\beta(\lambda)$ over $\Lambda(\beta)$, namely $\lambda^*(\beta)$, and (ii) the largest element of $\Lambda(\beta)$, i.e., $\max_{b\in\mathcal{B}} \lambda_b^{(1)}(\beta)$. Both quantities are induced by a specific bucket b; we additionally report the rank of the highest-probability item among those hashing to b.

Even for a finite time stream length $\tau = 5\times10^5$ with $n_{\mathcal{I}} = 10^4$, the asymptotic prediction $g_\beta(\lambda)$ closely tracks the Monte Carlo estimate of $\mathbb{E}\left[\overline{V_{\mathcal{B}}}(\tau)\right]$. Moreover, evaluating $g_\beta(\lambda)$ is substantially cheaper. It can be computed in $\mathcal{O}(n_{\mathcal{I}})$ for a fixed (λ, β), via Alg. 2, whereas simulations require $\mathcal{O}(n_{\text{runs}}\,\tau)$. We also observe that in general the theoretical curve is slightly above the empirical estimates.

Although the worst-case size of $\Lambda(\beta)$ is at most $m_1 = 200$, in this experiment it is much smaller: $|\Lambda(\beta)| = 15$, 12, and 10 for $\alpha \in \{0.8, 1.0, 1.2\}$, respectively. This reduction is mainly due to the restriction of λ being an integer, which makes the maximization over $\Lambda(\beta)$ extremely fast. Moreover, For the three values of α, the value of λ that maximizes the empirical estimate coincides with the asymptotically optimal threshold $\lambda^*(\beta)$.

Finally, the function $g_\beta(\lambda)$ varies sharply for small λ, but becomes comparatively flat after the optimum and then decreases gradually as λ grows. This suggests that, in practice, one may replace the exact optimizer by a simpler, tractable surrogate such as $\lambda_{\text{ub}}(\beta) \triangleq \max_{b\in\mathcal{B}} \lambda_b^{(1)}(\beta)$. Indeed, for the considered experiments, $g_\beta(\lambda^*(\beta)) \approx g_\beta(\lambda_{\text{ub}}(\beta))$. Moreover, as shown in § 3.2 and Appendix C, λ_{ub} is amenable to probabilistic analysis accounting for the skewness of $\boldsymbol{p}$. Nonetheless, the apparent flatness of $g_\beta(\lambda)$ beyond the optimum can be misleading: while the average changes little, large values of λ can induce substantial variability in performance, as observed in Fig. 1 of the introduction.

Using "xxHash" hash functions, and generating $\cup_{k=1}^{n_{\text{samp}}} \Lambda(\beta_k)$, with $n_{\text{samp}} = 1000$, we observe only 27, 25, and 22 distinct elements for $\alpha \in \{0.8, 1.0, 1.2\}$, respectively. This confirms that, in practice, the near-optimal eviction threshold $\widehat{\lambda}^*$ can be computed efficiently.

5 Proof of Theorem 1

Section 5.1 introduces additional definitions and notation used in the proof. Section 5.2 proves item 1 of Theorem 1, and Section 5.3 proves items 2 and 3 of the theorem.

5.1 Definitions and Notation

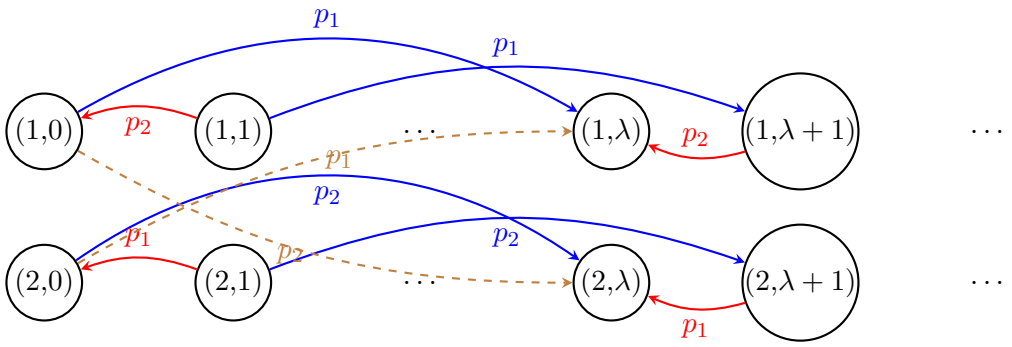


Figure 3: Illustration of the Markov chain M_b when $n_b = 2$.

Define $\mathcal{T}_b$ be the set of eviction times in bucket b, i.e.,

$$\mathcal{T}_b \triangleq \{t \in \mathbb{N} : \ S_b(t) \neq S_b(t+1)\}, \tag{33}$$

and $T_b^{\text{evict}}(t)$ is the last eviction time at step t, i.e.,

$$T_b^{\text{evict}}(0) = 0, \ T_b^{\text{evict}}(t) \triangleq \max\left(\mathcal{T}_b \cap \{0, 1, \ldots, t-1\}\right), \ \forall t \geq 1. \tag{34}$$

To study the limiting behavior of $\boldsymbol{S}$, we define the stochastic process M_b for each bucket b as,

$$M_b(t) \triangleq (S_b(t), U_b(t)), \ U_b(t) \triangleq \lambda V_b^+(t) - V_b^-(t). \tag{35}$$

The score $U_b(t)$ governs updates of $S_b(t)$ (see line 10 in Alg. 1). Conditioned on the hash functions $\{h_\ell\}_{\ell=1}^d$ and β, the evolution of M_b is determined solely by the stream $\boldsymbol{R}$. Hence, under Assumption 2.1, M_b is a time-homogeneous discrete time Markov chain on $\mathcal{I}_b \cup \{-1\} \times \mathbb{N}$. Initially, $M_b(0) = (-1, 0)$, and the one step transitions are given by,

$$(S_b(t+1), U_b(t+1)) = \begin{cases} (S_b(t), U_b(t)), & \text{w.p. } 1 - \mu_b, \\ (S_b(t), U_b(t) + \lambda), & \text{w.p. } p_{S_b(t)}, \text{ if } U_b(t) > 0, \\ (S_b(t), U_b(t) - 1), & \text{w.p. } \mu_b - p_{S_b(t)}, \text{ if } U_b(t) > 0, \\ (i, \lambda), & \text{w.p. } p_i, \text{ if } U_b(t) = 0, \ \forall i \in \mathcal{I}_b, \end{cases} \tag{36}$$

In words, upon a request for an item not in the bucket in question, $M_b(t)$ remains unchanged (w.p. $1 - \mu_b$). Moreover, when $U_b(t)$ is positive, $S_b(t)$ remains unchanged, and either $U_b(t)$ increases by λ (with probability $p_{S_b(t)}$) or decreases by 1 (w.p. $\mu_b - p_{S_b(t)}$). Once $U_b(t)$ hits 0, $S_b(t)$ may switch to a new item i w.p. p_i and U_b increases by λ anyway. Figure 3 illustrates the one step transitions of this Markov process when the total number of items is 2 and the number of buckets is 1. We denote the set of states with $U_b = 0$ as $\overline{\mathbf{0}_b}$, i.e., $\overline{\mathbf{0}_b} \triangleq \{(i, 0) : \ i \in \mathcal{I}_b \cup \{-1\}\}$.

5.2 Proof of item 1 in Theorem. 1

We prove that, for any bucket $b \in \mathcal{B}_\beta^-(\lambda)$, $(S_b(t))_t$ switches state infinitely often, and for any bucket $b \in \mathcal{B}_\beta^+(\lambda)$, $S_b(t)$ converges a.s. in t, such that for any item $i \in \mathcal{I}_b$, $\Pr(S_b^\infty = i) = a_{i,\beta}(\lambda)$. For any bucket $b \in \mathcal{B}_\beta^0(\lambda)$, $S_b(t) = -1$ at any step t. Lemma 3 shows that the transience/recurrence of M_b determines the limiting behavior of S_b; S_b converges and $\lim_{t\to\infty} T_b^{\text{evict}}(t)/t = 0$ a.s. when M_b is transient, otherwise, S_b switches state infinitely often and $\lim_{t\to\infty}(t - T_b^{\text{evict}}(t))/t = 0$ a.s.. Furthermore, Claim 1 shows that for any bucket $b \in \mathcal{B}_\beta^+(\lambda)$ $(\lambda > \lambda_b^{(1)}(\beta))$, the Markov chain M_b is transient, and it is recurrent for any bucket $b^- \in \mathcal{B}_\beta^-(\lambda)$ $(\lambda \le \lambda_b^{(1)}(\beta))$. Combining these two results, we deduce that, for any bucket $b \in \mathcal{B}_\beta^-(\lambda)$, $(S_b(t))_t$ switches state infinitely often, and for any bucket $b \in \mathcal{B}_\beta^+(\lambda)$, $S_b(t)$ converges a.s. in t. The distribution of S_b^∞ follows directly from Lemma 4. This proves item 1 of Thm. 1. Below, we present Lemmas 3 and 4, and Claim 1, along with their proofs.

Lemma 3 (Relation between M_b and S_b)**.** *Under Assumption 2.1, the transience/positive recurrence of M_b, for any $b \in \mathcal{B}_\beta^+(\lambda) \cup \mathcal{B}_\beta^-(\lambda)$, determines the limiting behavior of S_b:*

- *If M_b is recurrent, then $(S_b(t))_t$ switches state infinitely often ($|\mathcal{T}_b| = \infty$), and $\lim_{t\to\infty} \frac{T_b^{evict}(t)}{t} = 1$, a.s..*
- *If M_b is transient, then $(S_b(t))_t$ converges ($|\mathcal{T}_b|$ is finite), and $\lim_{t\to\infty} \frac{T_b^{evict}(t)}{t} = 0$, a.s..*

Proof of Lemma 3. Define $\mathcal{Z}_b$ as the set of time instants where M_b visits $\overline{\mathbf{0}}_b$, i.e.,

$$\mathcal{Z}_b \triangleq \{t \in \mathbb{N} : U_b(t) = 0\}. \tag{37}$$

M_b **is transient.** It follows that $\mathcal{Z}_b$ is finite almost surely (a.s.). Moreover, $\mathcal{T}_b \subset \mathcal{Z}_b$, because S_b can only change value when M_b is in $\overline{\mathbf{0}}_b$ (see (36)). It follows then that $\mathcal{T}_b$ is finite a.s.. Writing $T_b^{\text{evict}}(t) = \max \mathcal{T}_b \cap \{0, \dots, t-1\}$, we obtain the almost sure limit $\lim_{t\to\infty} T_b^{\text{evict}}(t) = t^*$ with $t^* = \max \mathcal{T}_b$ if $\mathcal{T}_b \neq \emptyset$ and it is equal to 0 otherwise. It follows that S_b stabilizes a.s. after t^*+1, i.e., $S_b(t) = S_b(t^*+1)$, for all $t \ge t^* + 1$, and $\lim_{t\to\infty} T_b^{\text{evict}}(t)/t = 0$, a.s..

M_b **is positive recurrent.** Since M_b is positive recurrent, the set $\mathcal{Z}_b$ of return times is infinite a.s.; write $\mathcal{Z}_b = (z_i)_{i\in\mathbb{N}}$ with $z_i < z_{i+1}$. By the strong Markov property, the increments $(z_{n+1} - z_n)_{n\in\mathbb{N}}$ are i.i.d. with finite mean. Next, we show that $\mathcal{T}_b$ is also infinite and write $\mathcal{T}_b = (t_i)_{i\in\mathbb{N}}$. We also show that there exist identically distributed random variables $(X_n)_{n\in\mathbb{N}}$ with $\mathbb{E}[X_0] < \infty$ such that

$$t_{n+1} - t_n \le X_n \quad \text{a.s. for all } n. \tag{38}$$

For $t \ge 0$, let $l(t)$ be the unique index such that $t_{l(t)} \le t < t_{l(t)+1}$. Then $T_b^{\text{evict}}(t) = t_{l(t)}$ and

$$0 \le t - T_b^{\text{evict}}(t) < t_{l(t)+1} - t_{l(t)} \le X_{l(t)}. \tag{39}$$

Since $X_{l(t)} \overset{d}{=} X_0$ with $\mathbb{E}[X_0] < \infty$ and $t \to \infty$, we conclude that $\lim_{t\to\infty}(t - T_b^{\text{evict}}(t))/t = 0$.
Proof that $\mathcal{T}_b$ is infinite and (38) **holds.** We represent $\mathcal{T}_b$ by thinning the renewal process $\mathcal{Z}_b$ with i.i.d. uniform random variables. Let $\{u_z\}_{z\in\mathcal{Z}_b}$ be i.i.d. Unif$[0,1]$, independent of $\mathcal{Z}_b$, and define

$$\mathcal{T}_b = \left\{z \in \mathcal{Z}_b : u_z \le 1 - p_{S(z)}\right\}. \tag{40}$$

This is valid because, whenever $U_b = 0$, the probability that S_b changes state at the next step is $1 - p_{S_b}$ (see (36)). Recall that $p_b^{(1)} \triangleq \max_{i\in\mathcal{I}_b} p_i$ and define the dominated thinning

$$\mathcal{T}' \triangleq \left\{ z \in \mathcal{Z}_b : u_z \le 1 - p_b^{(1)} \right\}. \tag{41}$$

Clearly $\mathcal{T}' \subseteq \mathcal{T}_b$. Writing $\mathcal{T}' = (t'_n)_{n\in\mathbb{N}}$, the inter-arrivals $(t'_{n+1} - t'_n)$ are i.i.d., and $t'_1 - t'_0 \overset{d}{=} \sum_{k=1}^{G_n}(z_k - z_{k-1})$, where $G_n \sim \text{Geom}(1 - p_b^{(1)})$ on $\{1, 2, \dots\}$, and $(z_k - z_{k-1})$ are i.i.d., independent of G_n. Hence

$$\mathbb{E}\left[t'_{n+1} - t'_n\right] = \mathbb{E}\left[G_0\right]\mathbb{E}\left[z_1 - z_0\right] = \frac{1}{1 - p_b^{(1)}}\mathbb{E}\left[z_1 - z_0\right] < \infty, \tag{42}$$

so $\mathcal{T}'$ is infinite a.s. and has i.i.d. inter-arrivals with finite mean.

Since $\mathcal{T}' \subseteq \mathcal{T}_b$, every inter-arrival in $\mathcal{T}_b$ is bounded by a (possibly larger) enclosing inter-arrival of $\mathcal{T}'$: for each n, there exists an index $j(n)$ with $t_{n+1} - t_n \leq t'_{j(n)+1} - t'_{j(n)}$. Set $X_n \triangleq t'_{j(n)+1} - t'_{j(n)}$. Then (X_n) are identically distributed as $t'_1 - t'_0$ and satisfy (38) with $\mathbb{E}\left[X_0\right] < \infty$.

M_b is null recurrent. The reasoning used to prove that $\mathcal{T}_b$ is infinite when M_b is positive recurrent applies here as well, but the inter-arrivals have infinite mean. Nonetheless, they are finite a.s., and thus, X_0/t is 0, allowing to deduce that $(t - T_b^{\text{evict}}(t))/t$ is 0 as well. This finishes the proof.

□

Claim 1 (Transience/recurrence of M_b)**.** *The comparison between λ and $\lambda_b^{(1)}(\beta)$ determines the transience/positive recurrence of M_b:*

- *If $b \in \mathcal{B}_\beta^-(\lambda)$, i.e., $\lambda \leq \lambda_b^{(1)}(\beta)$, then $(M_b(t))_t$ is recurrent.*
- *If $b \in \mathcal{B}_\beta^+(\lambda)$, i.e., $\lambda > \lambda_b^{(1)}(\beta)$, then $(M_b(t))_t$ is transient.*

Lemma 4 (Probability distribution of S_b^∞)**.** *For any bucket $b \in \mathcal{B}_\beta^+(\lambda)$, and for any item $i \in \mathcal{I}_b$, $\Pr\left(S_b^\infty = i\right) = a_{i,\beta}(\lambda)$, such that the function $a_{i,\beta}$ is defined in* (13)*.*

To prove Claim 1 and Lemma 4, we define the stochastic process G on $\mathbb{N}$ as follows,

$$G(t+1) = \begin{cases} G(t), & \text{w.p. } 1 - \mu, \\ \max\left(G(t) - 1, 0\right), & \text{w.p. } \mu - p, \\ G(t) + \lambda, & \text{w.p. } p. \end{cases} \tag{43}$$

and $0 < p \leq \mu \leq 1$. The process G captures the evolution of M_b for a fixed value of U_b. Define r_n as the return probability of G to 0 when starting at $n > 0$, and T_a as the hitting time of G on $a \geq 0$,

$$r_n \triangleq \Pr\left(T_0 < \infty \mid G(0) = n\right), \tag{44}$$

$$T_a \triangleq \inf\{t \geq 0 :\ G(t) = a\}, \qquad a \in \mathbb{N}. \tag{45}$$

Claim 2 (Hitting probabilities of G: Exact expression)**.** *For all $n \in \mathbb{N} \setminus \{0\}$, $r_n = (r_1)^n$. Moreover,*

$$r_1 = \begin{cases} 1, & \text{if } \lambda \geq \mu/p - 1 \\ \text{Unique root in } (0,1) \text{ of } \phi(\cdot, \lambda) - \mu/p, & \text{otherwise.} \end{cases} \tag{46}$$

Proof of Claim 2. Note that for any $n \geq 1$, $\mathbb{E}\left[G(1) - G(0) \mid G(0) = n\right] = p\left(\lambda + 1\right) - \mu$, which is strictly negative whenever $\frac{\mu}{p} > \lambda + 1$. Using a *Foster's criterion* [30, Thm. 114], we deduce that G is positive recurrent, and thus $r_n = 1$ for any n in this case.

To derive r_n for other values of λ, we prove that,

$$\forall n, m \geq 1,\ r_{n+m} = r_n r_m. \tag{47}$$

This implies that $r_n = r_1^n$. Moreover, from the Markov property, we obtain,

$$r_n = \begin{cases} (1-\mu)r_1 + pr_{1+\lambda} + (\mu - p), & \text{if } n = 1, \\ (1-\mu)r_n + pr_{n+\lambda} + (\mu-p)r_{n-1}, & \text{if } n \geq 2. \end{cases} \tag{48}$$

Developing (48), we obtain that r_1 must be a root in $[0,1]$ of the function f given by,

$$f(x) \triangleq px^{\lambda+1} - \mu x + \mu - p = (x-1)\left(p\phi(x,\lambda) - \mu\right), \tag{49}$$

where $\phi(x,\lambda) = \sum_{k=0}^{\lambda} x^k$. For $x \in [0,1]$, $\phi(x,\lambda)$ takes values in $[0, \lambda+1]$, and it is strictly increasing in x. Thus, (i) when $\lambda \leq \frac{\mu}{p} - 1$, 1 is the unique root of f, and therefore $r_n = 1$, $\forall n$, i.e., G is recurrent, and (ii), when $\lambda > \frac{\mu}{p} - 1$, f has two distinct roots, 1, and $x_0 \in (0,1)$, such that $\phi(x_0,\lambda) = \mu/p$. Next, we prove that $r_1 < 1$ when $\frac{\mu}{p} < \lambda + 1$, to deduce that r_1 is the solution to the equation $\phi(\cdot,\lambda) = \mu/p$.

Below, we prove (47) and then that $r_1 < 1$ when $\lambda > \frac{\mu}{p} - 1$.

Proof of (47)**.** Because downward moves in G are *skip-free* (only -1), any path that reaches 0 from $n+m$ must visit m first; hence $\{T_0 < \infty\} \subseteq \{T_m < \infty\}$, (see (45) for the definition of T_a for $a \in \mathbb{N}$) and therefore

$$\begin{aligned} r_{n+m} &= \Pr\left(T_0 < \infty \mid G(0) = n+m\right) && (50) \\ &= \Pr\left(T_0 < \infty \mid T_m < \infty,\ G(0) = n+m\right) \Pr\left(T_m < \infty \mid G(0) = n+m\right), && (51) \end{aligned}$$

since $\Pr\left(T_0 < \infty \mid T_m = \infty,\ G(0) = n+m\right) = 0$.

On $\{T_m < \infty\}$ and $G(0) = n+m$, we have $T_0 > T_m$ and thus $T_0 = \inf\{t \geq 0 :\ G(t+T_m) = 0\}$. Because T_m is a stopping time, the strong Markov property gives that $(G(t+T_m))_{t\geq 0} | \{G(0) = n+m, T_m < \infty\})$ has the same law as $(G(t))_{t\geq 0} | \{G(0) = m\})$. Hence

$$\Pr\left(T_0 < \infty \mid T_m < \infty,\ G(0) = n+m\right) = \Pr\left(T_0 < \infty \mid G(0) = m\right) = r_m. \tag{52}$$

By a space-homogeneity argument, we deduce that $\Pr\left(T_m < \infty \mid G(0) = n+m\right) = r_n$. Combining this with (51) and (52) finishes the proof.

Proof that $r_1 < 1$ **when** $\frac{\mu}{p} < \lambda + 1$**.** Let $\{J(t)\}_{t\geq 0}$ be the $\mathbb{Z}$-valued random walk with i.i.d. increments $(\zeta_j)_{j\geq 1}$ such that,

$$\Pr\left(\zeta_1 = \lambda\right) = p, \qquad \Pr\left(\zeta_1 = -1\right) = \mu - p, \qquad \Pr\left(\zeta_1 = 0\right) = 1 - \mu, \tag{53}$$

and $J(t) = n + \sum_{s=1}^{t} \zeta_s$ for some $n > 0$. Let μ be the expectation of ζ_1, i.e., $\gamma = p(\lambda+1) - \mu$. We also denote the moment generating function of ζ_1 as $\psi(\theta)$, i.e.,

$$\psi(\theta) \triangleq \mathbb{E}\left[e^{\theta\zeta_1}\right] = pe^{\theta\lambda} + (\mu - p)e^{-\theta} + (1-\mu). \tag{54}$$

When $\frac{\mu}{p} < \lambda + 1$, $\psi'(0) = \mu > 0$. And since $\psi(0) = 1$, then there exists $\theta_0 > 0$ such that $\psi(-\theta_0) \in (0,1)$.

When $G(0) = n$, G and J visit 0 for the first time at the same time, because $G(t) = \max\left(J(t), 0\right)$ (see (43)). Thus, we can bound r_n as follows,

$$r_n = \Pr\left(\exists t \geq 0 :\ J(t) = 0\right) \leq \sum_{t=0}^{\infty} \Pr\left(J(t) = 0\right) \leq \sum_{t=0}^{\infty} \mathbb{E}\left[e^{-\theta_0 J(t)}\right] = \sum_{t=0}^{\infty} e^{-\theta_0 n}\ \left(\psi(-\theta_0)\right)^t = \frac{e^{-\theta_0 n}}{1 - \psi(-\theta_0)}. \tag{55}$$

It follows that there exists $n_0 \geq 1$ such that $r_{n_0} < 1$, which means that G is transient, and thus $r_n < 1$ for any $n \geq 1$ when $\frac{p}{p} < \lambda + 1$. We deduce then that r_1 is the unique solution to the equation $\phi(\cdot,\lambda)$ in the interval $(0,1)$, which finishes the proof.

□

Now, we are ready to leverage Claim 2 to prove Claim 1 that characterizes the transience/recurrence of M_b based on the comparison between λ and $\lambda_b^{(1)}(\beta)$.

Proof of Claim 1. When $n_b = 1$, M_b is transient, because U_b diverges w.p. 1.

The objective is to prove that M_b is transient when $\lambda > \lambda_b^{(1)}(\beta)$, or equivalently, $\lambda > \lambda_i(\beta)$ for any item i hashing to bucket b, i.e., $\forall i \in \mathcal{I}_b$. Otherwise M_b is recurrent.

Define the probability of hitting $\overline{\mathbf{0}}_b$, when starting in state (i,n) as $r_{i,n}$, and $\bar{r}$ is the probability of hitting $\overline{\mathbf{0}}_b$ when starting from $\overline{\mathbf{0}}_b$. Formally,

$$r_{i,n} \triangleq \Pr\left(\exists t > 0 :\ M_b(t) \in \overline{\mathbf{0}}_b \mid M_b(0) = (i,n)\right), \qquad i \in \mathcal{I}_b,\ n \in \mathbb{N} \setminus \{0\} \tag{56}$$

$$\bar{r} \triangleq \Pr\left(\exists t > 0 :\ M_b(t) \in \overline{\mathbf{0}}_b \mid M_b(0) \in \overline{\mathbf{0}}_b\right). \tag{57}$$

Clearly M_b is an aperiodic and irreducible Markov chain (see (36)). Thus, M_b is transient, if and only if $\bar{r} < 1$, and it is recurrent if and only if $\bar{r} = 1$. From (36), starting at $\overline{\mathbf{0}}_b$ the chain either stays at $\overline{\mathbf{0}}_b$ with probability $1 - \mu_b$ or jumps to (i, λ) with probability p_i. Hence thanks to the Markov property, we can write,

$$\bar{r} = (1 - \mu_b) + \sum_{i \in \mathcal{I}_b} p_i\, r_{i,\lambda}. \tag{58}$$

we deduce that M_b is recurrent if and only if $r_{i,\lambda} = 1$ for all $i \in \mathcal{I}_b$, and it is transient otherwise.

When M_b is in state (i,n) with $n > 0$, the Markov chain G defined in (43) with $p = p_i$, and $\mu = \mu_{\beta(i)}$, captures the stochastic evolution of M_b until it reaches the state $(i,0)$. Thus, using Claim 2 we have that $r_{i,\lambda} = (r_{i,1})^{\lambda}$ with

$$r_{i,1} = \begin{cases} 1 & \text{if } \lambda \leq \frac{\mu_{\beta(i)}}{p_i} - 1 \\ \text{Unique root in } (0,1) \text{ of } \phi(\cdot,\lambda) - \mu_{\beta(i)}/p_i, & \text{otherwise.} \end{cases} \tag{59}$$

Noting that $\lambda_i(\beta) = \frac{\mu_{\beta(i)}}{p_i} - 1$ concludes the proof.

□

Proof of Lemma 4. The objective is to prove that $\Pr\left(S_b^{\infty} = i\right) = a_{i,\beta}(\lambda)$, where $a_{i,\beta}$ is defined in (13).

Recall that $r_{i,n}$, defined in (56) is the hitting probability of M_b on $\overline{\mathbf{0}}_b$, when starting in state (i,n) with $n \geq 1$, and $\bar{r}$, defined in (57), is the hitting probability of M_b on $\overline{\mathbf{0}}_b$ when starting from $\overline{\mathbf{0}}_b$. Both quantities were introduced in the proof of Claim 1. In particular, we observed that $r_{i,n}$ is also the hitting probability of G on 0 when starting at state n, when $\mu = \mu_{\beta(i)}$, and $p = p_i$. Thus, using Claim 2, we deduce the expression of $r_{i,n}$, which in turn enables us to deduce $\bar{r}$ (see (58)).

We use the notation $\Pr_{(i,n)}(\cdot)$ to indicate the conditional probability on $M_b(0) = (i,n)$. Initially $M_b(0) \in \overline{\mathbf{0}}_b$. We also use the short notation A_i^{∞} to indicate the event that $S_b^{\infty} = i$. We compute the probability of the event A_i^{∞} when starting at $\overline{\mathbf{0}}_b$, namely $\Pr_{\overline{\mathbf{0}}_b}(A_i^{\infty})$, as follows,

$$\Pr_{\overline{\mathbf{0}}_b}(A_i^{\infty}) = p_i \Pr_{(i,\lambda)}(A_i^{\infty}) + \sum_{j \in \mathcal{I}_b \setminus \{i\}} p_j \Pr_{(j,\lambda)}(A_i^{\infty}) + (1 - \mu_b)\Pr_{\overline{\mathbf{0}}_b}(A_i^{\infty}). \tag{60}$$

Define $T_{\overline{\mathbf{0}}_b}$ as the hitting time of M_b on $\overline{\mathbf{0}}_b$. Formally,

$$T_{\overline{\mathbf{0}}_b} \triangleq \inf\left\{t \geq 0 :\ M_b(t) \in \overline{\mathbf{0}}_b\right\}. \tag{61}$$

We then condition on whether $T_{\overline{\mathbf{0}}_b}$ is finite or infinite,

$$\begin{aligned} \Pr_{(i,\lambda)}(A_i^{\infty}) &= \Pr_{(i,\lambda)}\left(A_i^{\infty} \mid T_{\overline{\mathbf{0}}_b} < \infty\right) \Pr_{(i,\lambda)}\left(T_{\overline{\mathbf{0}}_b} < \infty\right) + \Pr_{(i,\lambda)}\left(A_i^{\infty} \mid T_{\overline{\mathbf{0}}_b} = \infty\right) \Pr_{(i,\lambda)}\left(T_{\overline{\mathbf{0}}_b} = \infty\right) \\ &= \Pr_{\overline{\mathbf{0}}_b}(A_i^{\infty})\, r_{i,\lambda} + 1 \cdot (1 - r_{i,\lambda}) \\ &= \Pr_{\overline{\mathbf{0}}_b}(A_i^{\infty})\, r_{i,\lambda} + (1 - r_{i,\lambda}), \end{aligned} \tag{62}$$

where we have used the strong Markov property to write that $\Pr_{(i,\lambda)}\left(A_i^\infty \mid T_{\overline{\mathbf{0}}_b} < \infty\right) = \Pr_{\overline{\mathbf{0}}_b}(A_i^\infty)$. Moreover, the equality $r_{i,\lambda} = \Pr_{(i,\lambda)}\left(T_{\overline{\mathbf{0}}_b} < \infty\right)$ follows from the definition of $r_{i,\lambda}$ in (56). Similarly for $j \neq i$:

$$\begin{aligned}\Pr_{(j,\lambda)}(A_i^\infty) &= \Pr_{(j,\lambda)}\left(A_i^\infty \mid T_{\overline{\mathbf{0}}_b} < \infty\right)\Pr_{(j,\lambda)}\left(T_{\overline{\mathbf{0}}_b} < \infty\right) + \Pr_{(j,\lambda)}\left(A_i^\infty \mid T_{\overline{\mathbf{0}}_b} = \infty\right)\Pr_{(j,\lambda)}\left(T_{\overline{\mathbf{0}}_b} = \infty\right)\\ &= \Pr_{\overline{\mathbf{0}}_b}(A_i^\infty)\, r_{j,\lambda} + 0 \cdot (1 - r_{j,\lambda})\\ &= \Pr_{\overline{\mathbf{0}}_b}(A_i^\infty)\, r_{j,\lambda}. \end{aligned} \tag{63}$$

Combining (60), (62), and (63), we deduce that,

$$\Pr_{\overline{\mathbf{0}}_b}(A_i^\infty) = \frac{p_i(1 - r_{i,\lambda})}{\mu_b - \sum_{j\in\mathcal{I}_b} p_j r_{j,\lambda}} = \frac{p_i(1 - r_{i,\lambda})}{\sum_{j\in\mathcal{I}_b} p_j(1 - r_{j,\lambda})}. \tag{64}$$

Observe that $r_{i,\lambda} = (r_{i,1})^\lambda$, and $r(\mu_{\beta(i)}/p_i, \lambda) = r_{i,1}$, where the function $r(\cdot,\cdot)$ is defined in (10). Thus, $\Pr_{\overline{\mathbf{0}}_b}(A_i^\infty) = a_{i,\beta}(\lambda)$, which finishes the proof.

□

5.3 Proof of items 2 and 3 of Theorem 1

We prove the limiting distribution for V_b^+ and $Y_{l,c}$ in equations (15) and (16). We derive in Lemma 5 a finite time characterization of the counters V_b^+ and $Y_{\ell,c}$ in terms of the counts vector $\boldsymbol{N}$ and the last eviction time T_b^{evict}.

Lemma 5. *The following holds at any step t,*

$$V_b^+(t) = N_{S_b(t)}(t) - N_{S_b(t)}(T_b^{evict}(t)), \tag{65}$$

$$Y_{\ell,c}(t) = Y_{\ell,c}^{\texttt{CM}}(t) - \sum_{b=1}^{m_1}\left(N_{S_b(t)}(t) - N_{S_b(t)}(T_b^{evict}(t))\right)\mathbb{1}\left(h_\ell(S_b(t)) = c\right), \tag{66}$$

The proof of Lemma 5 is presented in Section A of the appendix.

Combining Lemmas 3 and 5, and Claim 1, for any $b^+ \in \mathcal{B}_\beta^+(\lambda)$ as $t \to \infty$, the following holds a.s.,

$$\lim_{t\to\infty}\frac{1}{t}\left(N_{S_{b^+}(t)}(t) - N_{S_{b^+}(t)}(T_{b^+}^{\text{evict}}(t))\right) = p_{S_{b^+}(\infty)} - \lim_{t\to\infty}\frac{N_{S_{b^+}(t)}(T_{b^+}^{\text{evict}}(t))}{t} = p_{S_{b^+}^\infty}, \tag{67}$$

where we have used the fact that $\lim_{t\to\infty} N_i(t)/t = p_i$, and $N_i(T_b^{\text{evict}}(t)) \le T_b^{\text{evict}}$ for any item i. On the other hand, for any $b^- \in \mathcal{B}_\beta^-(\lambda)$, we have that a.s.,

$$\lim_{t\to\infty}\frac{1}{t}\left(N_{S_{b^-}(t)}(t) - N_{S_{b^-}(t)}(T_{b^-}^{\text{evict}}(t))\right) \le \lim_{t\to\infty}\frac{t - T_{b^-}^{\text{evict}}(t)}{t} = 0, \tag{68}$$

where we have used the fact that $\left(N_i(t) - N_i(T_{b^-}^{\text{evict}}(t))\right)$ is the number of appearances of item i between time steps t and $T_{b^-}^{\text{evict}}(t) + 1$. Plugging (67) and (68) in Lemma 5, we prove (15) and (16).

6 Conclusion

We studied the limiting behavior of `Elastic-Sketch` under a random stationary stream model and derived closed-form expressions for the asymptotic expected counting error. These expressions enable an efficient grid search for the near-optimal configuration of the eviction threshold λ and the memory split between the heavy and `CM` blocks controlled via m_1. Moreover, our characterization of the optimal threshold restricts the search for λ to a small candidate set, substantially reducing its computational cost. In future work, we aim to move beyond grid search by deriving explicit tuning rules for (λ, m_1) as a function of the arrival distribution $\boldsymbol{p}$.

References

[1] R. Basat, G. Einziger, M. Mitzenmacher, and S. Vargaftik. Salsa: Self-adjusting lean streaming analytics. In *2021 IEEE 37th International Conference on Data Engineering (ICDE)*, 2021. doi:10.1109/ICDE51399.2021.00080.

[2] Ran Ben-Basat, Gil Einziger, Roy Friedman, and Yaron Kassner. Heavy hitters in streams and sliding windows. In *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9. IEEE, 2016.

[3] Vladimir Braverman, Ran Gelles, and Rafail Ostrovsky. How to catch l2-heavy-hitters on sliding windows. *Theoretical Computer Science*, 554:82–94, 2014.

[4] L. Breslau, Pei Cao, Li Fan, G. Phillips, and S. Shenker. Web caching and zipf-like distributions: evidence and implications. In *IEEE INFOCOM '99. Conference on Computer Communications. Proceedings. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies. The Future is Now (Cat. No.99CH36320)*, volume 1, pages 126–134 vol.1, 1999. doi:10.1109/INFCOM.1999.749260.

[5] Andrei Broder and Michael Mitzenmacher. Network Applications of Bloom Filters: A Survey. *Internet Mathematics*, 1, 2003.

[6] Pei Cao and Sandy Irani. Cost-aware www proxy caching algorithms. In *Proceedings of the USENIX Symposium on Internet Technologies and Systems on USENIX Symposium on Internet Technologies and Systems*, USITS'97, page 18, USA, 1997. USENIX Association.

[7] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. *Theoretical Computer Science*, 312, 2004. doi:10.1016/S0304-3975(03)00400-6.

[8] Graham Cormode and S. Muthukrishnan. An improved data stream summary: the count-min sketch and its applications. *Journal of Algorithms*, 55(1):58–75, 2005. doi:10.1016/j.jalgor.2003.12.001.

[9] Graham Cormode and S. Muthukrishnan. Summarizing and mining skewed data streams. In *Proceedings of the SIAM International Conference on Data Mining (SDM)*, 2005. doi:10.1137/1.9781611972757.5.

[10] 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.

[11] Erik D. Demaine, Alejandro López-Ortiz, and J. Ian Munro. Frequency estimation of internet packet streams with limited space. In *Algorithms - ESA 10th Annual European Symposium*, volume 2461 of *Lecture Notes in Computer Science*, pages 348–360, 2002. doi:10.1007/3-540-45749-6_33.

[12] Kahlil Dozier, Loqman Salamatian, and Dan Rubenstein. Analysis of false negative rates for recycling bloom filters (yes, they happen!). *Proc. ACM Meas. Anal. Comput. Syst.*, 8(2):21:1–21:34, 2024. doi:10.1145/3656005.

[13] Éric Fusy and Frécéric Giroire. Estimating the number of active flows in a data stream over a sliding window. In *2007 Proceedings of the Fourth Workshop on Analytic Algorithmics and Combinatorics (ANALCO)*, pages 223–231. SIAM, 2007.

[14] Sahil Garg, Amritpal Singh, Gagangeet Singh Aujla, Sukhdeep Kaur, Shalini Batra, and Neeraj Kumar. A probabilistic data structures-based anomaly detection scheme for software-defined internet of vehicles. *IEEE Transactions on Intelligent Transportation Systems*, 22, 2021. doi:10.1109/TITS.2020.2988065.

[15] Qun Huang, Xin Jin, Patrick P. C. 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*, SIGCOMM '17, page 113–126, 2017. doi:10.1145/3098822.3098831.

[16] Richard M. Karp, Scott Shenker, and Christos H. Papadimitriou. A simple algorithm for finding frequent elements in streams and bags. *ACM Trans. Database Syst.*, 28:51–55, 2003. doi:10.1145/762471.762473.

[17] Ashwin Lall, Vyas Sekar, Mitsunori Ogihara, Jun Xu, and Hui Zhang. Data streaming algorithms for estimating entropy of network traffic. *ACM SIGMETRICS Performance Evaluation Review*, 34(1):145–156, 2006.

[18] Jiaqian Liu, Ran Ben Basat, Louis De Wardt, Haipeng Dai, and Guihai Chen. Disco: A dynamically configurable sketch framework in skewed data streams. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)*, pages 4801–4814, 2024. doi:10.1109/ICDE60146.2024.00365.

[19] 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.

[20] 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.

[21] Gurmeet Singh Manku and Rajeev Motwani. Approximate frequency counts over data streams. In *Proceedings of 28th International Conference on Very Large Data Bases, VLDB*, pages 346–357. Morgan Kaufmann, 2002. doi:10.1016/B978-155860869-6/50038-X.

[22] Younes Ben Mazziane and Othmane Marfoq. Universal and tight bounds on counting errors of count-min sketch with conservative updates. *Proc. ACM Meas. Anal. Comput. Syst.*, 9(2):1–32, 2025. doi:10.1145/3727135.

[23] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. Efficient computation of frequent and top-k elements in data streams. In *Database Theory - ICDT*, pages 398–412, 2005.

[24] Jayadev Misra and David Gries. Finding repeated elements. *Sci. Comput. Program.*, 2(2):143–152, 1982.

[25] Michael Mitzenmacher and Eli Upfal. *Probability and computing: Randomization and probabilistic techniques in algorithms and data analysis*. Cambridge university press, 2017.

[26] Michael Mitzenmacher and Salil P Vadhan. Why simple hash functions work: exploiting the entropy in a data stream. In *SODA*, volume 8, pages 746–755. Citeseer, 2008.

[27] Jelani Nelson. Sketching and streaming algorithms for processing massive data. *XRDS*, 19(1):14–19, sep 2012. doi:10.1145/2331042.2331049.

[28] Martin Raab and Angelika Steger. "balls into bins" - A simple and tight analysis. In *Randomization and Approximation Techniques in Computer Science, Second International Workshop, RANDOM'98, Barcelona, Spain, October 8-10, 1998, Proceedings*, volume 1518 of *Lecture Notes in Computer Science*, pages 159–170. Springer, 1998. doi:10.1007/3-540-49543-6_13.

[29] Patrick Rebeschini. Bernstein's concentration inequalities. fast rates. Lecture notes (Algorithmic Foundations of Learning, Lecture 7), University of Oxford, December 2021. Version: December 8, 2021. URL: https://www.stats.ox.ac.uk/~rebeschi/teaching/AFoL/22/material/lecture07.pdf.

[30] Richard Serfozo. *Basics of applied stochastic processes*. Springer Science & Business Media, 2009.

[31] 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*, SIGCOMM '18, 2018. doi:10.1145/3230543.3230544.

[32] Tong Yang, Jie Jiang, Peng Liu, Qun Huang, Junzhi Gong, Yang Zhou, Rui Miao, Xiaoming Li, and Steve Uhlig. Adaptive measurements using one elastic sketch. *IEEE/ACM Transactions on Networking*, 27(6):2236–2251, 2019. doi:10.1109/TNET.2019.2943939.

A Proof of Lemma 5

Proof of Lemma 5. We first observe that (66) is equivalent to,

$$Y_{\ell,c}(t) = \sum_{i\in\mathcal{I}} \delta_{\ell,c}(i)\left[(1-\delta_i(t,\beta))N_i(t) + \delta_i(t,\beta)N_i(T^{\text{evict}}_{\beta(i)}(t))\right],\ \delta_i(t,\beta) = \mathbb{1}\left(S_{\beta(i)}(t) = i\right), \tag{69}$$

where $\delta_{\ell,c}(i) = \mathbb{1}(h_l(i) = c)$. Since $Y^{\texttt{CM}}_{\ell,c}(t) = \sum_{i\in\mathcal{I}} \delta_{\ell,c}(i)N_i(t)$, the difference $Y^{\texttt{CM}}_{\ell,c}(t) - Y_{\ell,c}(t)$ from (69) can be partitioned into m_1 terms each corresponding to items in $\mathcal{I}_b$. In each one of these partitions, $\delta_i(t,\beta)$ is only equal to 1 for $S_b(t)$ and hence (66) is equivalent to (69). The proof of (69) follows an induction argument.

Base case $t = 0$**.** All vectors are 0 (or -1), hence both sides of (65) and (66) vanish.

Induction step. Let $b = \beta(R_t)$ and assume (66) and (65) hold at step $t-1$. We show they still hold after processing R_t.

Case 1: $S_b(t-1) = -1$ **(empty bucket).** The algorithm stores $S_b(t) = R_t$, sets $V_b^+(t) = 1$, leaves $\boldsymbol{Y}$ unchanged, and $T_b^{\text{evict}}(t) = t-1$. Because $N_{R_t}(t) = N_{R_t}(t-1) + 1$, the right–hand side of (65) equals $N_{R_t}(t) - N_{R_t}(t-1) = 1 = V_b^+(t)$. For (66), the item R_t switches from unmonitored to monitored ($\delta_{R_t}(t,\beta) = 1$ and $\delta_{R_t}(t-1,\beta) = 0$) its contribution changes from $N_{R_t}(t-1)$ to $N_{R_t}(T_b^{\text{evict}}(t)) = N_{R_t}(t-1)$. Hence the sum is unchanged, matching the unchanged $\boldsymbol{Y}$.

Case 2: $S_b(t-1) = R_t$ **(bucket hit).** Here $S_b(t-1) = S_b(t)$, $T_b^{\text{evict}}(t) = T_b^{\text{evict}}(t-1)$, and $V_b^+(t) = V_b^+(t-1)+1$. Thus (65) is preserved because $N_{R_t}(t) - N_{R_t}(T_b^{\text{evict}}(t)) = N_{R_t}(t-1)+1-N_{R_t}(T_b^{\text{evict}}(t-1))$. $\boldsymbol{Y}$ is untouched and R_t remains monitored, so its term in (66) is still $N_{R_t}(T_b^{\text{evict}}(t-1))$. Therefore, the sum does not change and (66) holds.

Case 3: $S_b(t-1) \neq R_t$ **and** $\lambda V_b^+(t-1) - V_b^-(t-1) > 0$ **(mismatch, vote passes).** The bucket content is not modified ($S_b(t-1) = S_b(t)$, $T_b^{\text{evict}}(t-1) = T_b^{\text{evict}}(t)$, and $V_b^+(t) = V_b^+(t-1)$), only V_b^- increases, and $\boldsymbol{Y}$ is updated by incrementing the cells corresponding to R_t by 1. Thus both sides of equation (65) remains unchanged. Moreover, $\delta_{R_t}(t,\beta) = \delta_{R_t}(t-1,\beta) = 0$ and hence the RHS of (66) for all cells $(\ell, h_\ell(R_t))$ with $r \in [d]$ increase by 1 and thus (66) holds at step t.

Case 4: $S_b(t-1) \neq R_t$ **and** $\lambda V_b^+(t-1) - V_b^-(t-1) \leq 0$ **(mismatch, vote fails, eviction).** Let $j = S_b(t-1)$. The algorithm evicts j, adds $V_b^+(t-1)$ to every CMS cell $(\ell, h_\ell(j))$, then sets $S_b(t) = R_t$, $V_b^+(t) = 1$, $V_b^-(t) = 0$, and $T_b^{\text{evict}}(t) = t-1$. Since $N_{R_t}(t) = N_{R_t}(t-1) + 1$, the right–hand side of (65) becomes $N_{R_t}(t) - N_{R_t}(t-1) = 1 = V_b^+(t)$, so (65) holds. For (66), note that 1) For any cell $(\ell, c) \neq (\ell, h_\ell(j))$, the term for R_t changes by $\delta_{\ell,c}(R_t)\left(N_{R_t}(T_b^{\text{evict}}(t)) - N_{R_t}(t-1)\right) = 0$, and no other contributions change, so $Y_{\ell,c}(t) = Y_{\ell,c}(t-1)$, 2) For each cell $(\ell, h_\ell(j))$, the term for j increases by $N_j(t) - N_j(T_b^{\text{evict}}(t-1)) = V_b^+(t-1)$, matching the `CM` update. Hence (66) holds at step t in all cells.

All cases have been tested, which finishes the proof.

□

B Proof of Lemma 1

For lighter notation, we drop the subscript referring to the hash function β. The objective is to prove that g_b is decreasing over $(\lambda_b^{(1)}, \infty)$. To this end, we prove that for any $\ell \geq 2$,

1. The left limit of g_b at $\lambda_b^{(\ell)}$ is larger than the right limit at the same point, i.e.,

$$\lim_{\lambda \to \lambda^{(\ell)}-} g_b(\lambda) \geq \lim_{\lambda \to \lambda^{(\ell)}+} g_b(\lambda) \tag{70}$$

2. The function g_b is decreasing in the interval $(\lambda_b^{(\ell)}, \lambda_b^{(\ell+1)}]$.

Proof of (70)**.** For brevity write $\lambda^{(\ell)} = \lambda_b^{(\ell)}$, $p^{(j)} = p_b^{(j)}$, $w^{(j)} = p^{(j)} w\left(\lambda^{(\ell)}, \mu_b/p^{(j)}\right)$, $u_\ell = \sum_{j=1}^{\ell-1} w^{(j)}$, and $v_\ell = \sum_{j=1}^{\ell-1} p^{(j)} w^{(j)}$. For $\lambda < \lambda^{(\ell)}$ there are $\ell - 1$ active items (any item i such that $a_i(\lambda) > 0$) in bucket b, and for $\lambda > \lambda^{(\ell)}$ there are ℓ active items. Therefore the left and right limits of g_b at $\lambda^{(\ell)}$ can be written as,

$$\lim_{\lambda \to \lambda^{(\ell)}-} g_b(\lambda) = \frac{v_\ell}{u_\ell}, \quad \lim_{\lambda \to \lambda^{(\ell)}+} g_b(\lambda) = \frac{v_\ell + p^{(\ell)} w^{(\ell)}}{u_\ell + w^{(\ell)}}. \tag{71}$$

Therefore,

$$\lim_{\lambda \to \lambda^{(\ell)}-} g_b(\lambda) - \lim_{\lambda \to \lambda^{(\ell)}+} g_b(\lambda) = \frac{w^{(\ell)} v_\ell - p^{(\ell)} w^{(\ell)} u_\ell}{u_\ell \left(u_\ell + w^{(\ell)}\right)} = \frac{w^{(\ell)}}{u_\ell \left(u_\ell + w^{(\ell)}\right)} \sum_{j=1}^{\ell-1} w^{(j)} \left(p^{(j)} - p^{(\ell)}\right) > 0. \tag{72}$$

Proof that g_b is decreasing over $(\lambda_b^{(\ell)}, \lambda_b^{(\ell+1)}]$. Define the function $\zeta : [1, +\infty)^2 \mapsto \mathbb{R}$ as,

$$\zeta(\lambda, C) \triangleq \frac{\partial}{\partial \lambda} \ln w(\lambda, C). \tag{73}$$

We prove in Lemma 6 that if $\zeta(\lambda, \cdot)$ is increasing, then g_b is decreasing over $(\lambda_b^{(\ell)}, \lambda_b^{(\ell+1)}]$, and we prove in Lemma 7 that $\zeta(\lambda, \cdot)$ is indeed increasing, to deduce that g_b is decreasing over $(\lambda_b^{(\ell)}, \lambda_b^{(\ell+1)}]$. The two lemmas are presented below along with their proofs.

Lemma 6. *Let $\lambda \in (\lambda_b^{(\ell)}, \lambda_b^{(l+1)}]$ for some l. If $\zeta(\lambda, \cdot)$ is increasing then g_b is decreasing in λ.*

Proof of Lemma 6. Here, we drop the subscripts that indicate the choice of the hash function β for lighter notation. Using the definitions of the functions in (11), (13), and (17), g_b can be written as,

$$g_b(\lambda) = \frac{\sum_{j \in \mathcal{A}} p_j w_j(\lambda)}{\sum_{k \in \mathcal{A}} w_k(\lambda)}, \tag{74}$$

where $\mathcal{A}$ is the set of active items in bucket b, i.e., $\mathcal{A} \triangleq \{i \in \beta_b^{-1} : \ \lambda_i < \lambda_b^{(l+1)}\}$. Further define the sets $\mathcal{A}^{2,+}$ and $\mathcal{A}^{2,-}$ as,

$$\mathcal{A}^{2,+} \triangleq \{(j,k) \in \mathcal{A}^2 : p_j > p_k\}, \ \mathcal{A}^{2,-} \triangleq \{(j,k) \in \mathcal{A}^2 : p_j < p_k\}, \tag{75}$$

and $C_j = P_{\beta(j)}/p_j$. The sign of the derivative of g_b matches the sign of γ, that is given by,

$$\begin{aligned}
\gamma &= \sum_{(j,k)\in\mathcal{A}^2} p_j w_j^{'}(\lambda) w_k(\lambda) - \sum_{(j,k)\in\mathcal{A}^2} p_j w_j(\lambda) w_k^{'}(\lambda) && (76)\\
&= \sum_{(j,k)\in\mathcal{A}^2} p_j w_j^{'}(\lambda) w_k(\lambda) - \sum_{(j,k)\in\mathcal{A}^2} p_k w_k(\lambda) w_j^{'}(\lambda) && (77)\\
&= \sum_{(j,k)\in\mathcal{A}^2} w_k(\lambda) w_j^{'}(\lambda)\,(p_j - p_k) && (78)\\
&= \sum_{(j,k)\in\mathcal{A}^{2,+}} (p_j - p_k) w_k(\lambda) w_j^{'}(\lambda) + \sum_{(j,k)\in\mathcal{A}^{2,-}} (p_j - p_k) w_k(\lambda) w_j^{'}(\lambda) && (79)\\
&= \sum_{(j,k)\in\mathcal{A}^{2,+}} (p_j - p_k) w_k(\lambda) w_j^{'}(\lambda) + \sum_{(j,k)\in\mathcal{A}^{2,+}} (p_k - p_j) w_j(\lambda) w_k^{'}(\lambda) && (80)\\
&= \sum_{(j,k)\in\mathcal{A}^{2,+}} (p_j - p_k) \left(w_k(\lambda) w_j^{'}(\lambda) - w_j(\lambda) w_k^{'}(\lambda) \right) && (81)\\
&= \sum_{(j,k)\in\mathcal{A}^{2,+}} (p_j - p_k) p_k p_j \left(w(\lambda, C_k) \partial_\lambda w(\lambda, C_j) - w(\lambda, C_j) \partial_\lambda w(\lambda, C_k) \right) && (82)\\
&= \sum_{(j,k)\in\mathcal{A}^{2,+}} (p_j - p_k) p_k p_j \left(w(\lambda, C_j) w(\lambda, C_k) \right) \left(\zeta(\lambda, C_j) - \zeta(\lambda, C_k) \right). && (83)
\end{aligned}$$

Observe that any two items j and k in $\mathcal{A}^{2,+}$ hash to the same bucket with $p_j > p_k$, and thus $C_j < C_k$. Therefore, if $\zeta(\lambda, \cdot)$ is increasing, then g_b is decreasing, which finishes the proof. □

Lemma 7. *For any $\lambda \geq 1$, the function $\zeta(\lambda, \cdot)$, defined in (73), is increasing.*

Proof. We first derive the partial derivatives of the function r, defined in (10), and deduce that r is decreasing in λ and increasing in C. We then leverage these expressions to deduce that ζ can be written as,

$$\zeta(\lambda, C) = f(r(\lambda, C)) : \; f(x) \triangleq \frac{-x^\lambda \ln x}{\lambda x^{\lambda+1} - (\lambda+1)x^\lambda + 1}. \tag{84}$$

Next, we prove that f is increasing over $(0, 1)$, and given that r is increasing in C, we deduce that ζ is increasing in C.

Partial Derivatives of r. For brevity write $r = r(\lambda, C)$ and assume that $\lambda > C - 1$ Let $\partial_\lambda r$ and $\partial_C r$ designate the partial derivatives of r (see (10)) with respect to λ and C, respectively. We prove that,

$$\partial_\lambda r = \frac{(r^{\lambda+1} \ln r)(1 - r)}{\lambda r^{\lambda+1} - (\lambda+1) r^\lambda + 1}, \;\; \partial_C r = \frac{(r-1)^2}{\lambda r^{\lambda+1} - (\lambda+1) r^\lambda + 1}. \tag{85}$$

We differentiate the equation $\phi(r, \lambda) = C$ to deduce that,

$$\partial_\lambda r = -\frac{\partial_2 \phi(r, \lambda)}{\partial_1 \phi(r, \lambda)}, \;\; \partial_C r = \frac{1}{\partial_1 \phi(r, \lambda)}, \tag{86}$$

where $\partial_1 \phi$ and $\partial_2 \phi$ designate the partial derivatives of ϕ with respect to the first and second components, respectively. Direct calculation yields,

$$\partial_2 \phi(r, \lambda) = \frac{r^{\lambda+1} \ln r}{r - 1}, \;\; \partial_1 \phi(r, \lambda), \frac{\lambda r^{\lambda+1} - (\lambda+1) r^\lambda + 1}{(r-1)^2}, \tag{87}$$

which enables us to deduce (85). Note that $x \mapsto \lambda x^{\lambda+1} - (\lambda+1)x^{\lambda} + 1$ is decreasing over $(0,1)$—direct computation of the derivative—and its value at 0 and 1 are positive, and thus $\partial_\lambda r \leq 0$ and $\partial_C r \geq 0$.

Proof of (84)**.** From $\phi(r,\lambda) = C$ we get,

$$C - 1 = \frac{r - r^{\lambda+1}}{1-r} = \frac{rw(\lambda, C)}{1-r} \implies w(\lambda, C) = (C-1)\frac{1-r}{r} \implies \zeta(\lambda, C) = \frac{\partial_\lambda r}{r(1-r)}. \tag{88}$$

Using the expression in (86), we deduce (84).

Proof that f **is increasing over** $(0,1)$**.** Let $z = -\ln x \in (0,\infty)$ so that $x = e^{-z}$. Then

$$f\left(e^{-z}\right) = \frac{-e^{-\lambda z}\ln\left(e^{-z}\right)}{\lambda e^{-(\lambda+1)z} - (\lambda+1)e^{-\lambda z} + 1} = \frac{z}{\alpha(z)}, \tag{89}$$

$$\alpha(z) \triangleq e^{\lambda z} - (\lambda+1) + \lambda e^{-z}. \tag{90}$$

We have $\alpha(0) = 0$ and for $z > 0$, $\alpha'(z) = \lambda\left(e^{\lambda z} - e^{-z}\right) > 0$, so $\alpha(z) > 0$. Moreover, $\alpha''(z) = \lambda^2 e^{\lambda z} + \lambda e^{-z} > 0$, hence α is convex on $(0,\infty)$. Fix $0 < u < z$ and set $\theta = u/z \in (0,1)$. By convexity,

$$\alpha(u) = \alpha\left(\theta z + (1-\theta)\,0\right) \leq \theta\alpha(z) + (1-\theta)\,\alpha(0) = \frac{u}{z}\alpha(z), \implies \frac{z}{\alpha(z)} \leq \frac{u}{\alpha(u)}. \tag{91}$$

Thus $z \mapsto f\left(e^{-z}\right) = z/\alpha(z)$ is decreasing on $(0,\infty)$. Since $z = -\ln x$ is decreasing in x on $(0,1)$, it follows that $x \mapsto f(x)$ is increasing on $(0,1)$. This finishes the proof.

□

C Probabilistic Bounds for the Optimal Eviction Threshold

The high-probability bound on $\lambda^*(\beta)$ in Theorem 3 is distribution-agnostic and therefore does not capture how the skew of the arrival distribution $\boldsymbol{p}$ affects $\lambda^*(\beta)$. A more $\boldsymbol{p}$-sensitive route is to upper-bound $\max_{b\in\mathcal{B}} \lambda_b^{(1)}(\beta)$ and then apply a union bound over buckets.

Assume for simplicity that $\mathcal{I} = [n_{\mathcal{I}}]$ and $p_1 > p_2 > \cdots > p_{n_{\mathcal{I}}}$. Fix a bucket $b \in \mathcal{B}$ and condition on its *dominant* item, i.e., the highest-probability item among those hashing to b. The probability that the dominant item in bucket b is item j occurs with probability $\frac{1}{m_1}\left(1 - \frac{1}{m_1}\right)^{j-1}$ .Conditioned on this event (so that $p_b^{(1)} = p_j$), $\lambda_b^{(1)}$ can be written as,

$$\lambda_b^{(1)}(\beta) = \frac{\mu_b}{p_j} - 1 = \sum_{k=j+1}^{n_{\mathcal{I}}} \mathbb{1}\{\beta(k) = b\}\,\frac{p_k}{p_j}. \tag{92}$$

The right-hand side is a weighted sum of independent Bernoulli indicators (each $\mathbb{1}\{\beta(k) = b\}$ has mean $1/m_1$). Thus, for each fixed j, one can apply Bernstein's inequality to control $\lambda_b^{(1)}(\beta)$ under the conditioning. Averaging over j and union-bounding over buckets yields a $\boldsymbol{p}$-dependent high-probability upper bound on $\lambda^*(\beta)$.