Modeling Conservative Updates in Multi-Hash Approximate Count Sketches

Giuseppe Bianchi[a] , Ken Duffy[b], Douglas Leith[b], Vsevolod Shneer[c]
(a) CNIT / Univ. Roma Tor Vergata, Italy; (b) NUIM Hamilton Institute, Ireland; (c) Heriot-Watt University, UK
giuseppe.bianchi@uniroma2.it, ken.duffy@nuim.ie, doug.leith@nuim.ie, v.shneer@hw.ac.uk

***Abstract*—Multi-hash-based count sketches are fast and memory efficient probabilistic data structures that are widely used in scalable online traffic monitoring applications. Their accuracy significantly improves with an optimization, called conservative update, which is especially effective when the aim is to discriminate a relatively small number of heavy hitters in a traffic stream consisting of an extremely large number of flows. Despite its widespread application, a thorough understanding of the conservative update operation has lagged behind, perhaps because of the significant modeling complexity involved. In this work we attempt to fill this gap. Our proposed modeling approach builds on a practically important empirical finding: simulation results (as well as experimental ones over real traffic traces) obtained for skewed load scenarios exhibit a sharp waterfall-type behaviour. That is, the approximate count provided by the sketch response remains accurate until an "error floor" is reached. Flows below this error flow level are on average approximated by the *same* error floor count value, irrespective of their exact count. The error floor itself appears to be maximal in the case of uniform load. Leveraging the simplifications made possible when the load is uniform, we derive an analytic model capable of accurately predicting the transient growth behavior of the (tightly correlated) counters deployed in the data structure and obtain an upper bound on the error floor level.**

I. Introduction

Multi-hash-based approximate accounting algorithms, hereafter generically referred to as *count sketches*, are widely applied in monitoring applications. Their scalable and fast operation, $O(1)$, and limited memory requirements make their deployment appealing in high speed hardware implementations [1]. One of their most common fields of application is that of network measurement over high speed links. Such links are used by possibly *millions* of concurrent flows, making it hard to individually track flows with dedicated counters. In a seminal paper, Estan and Varghese [2] suggested a construction, called a multi-stage filter, for detecting *heavy hitters* without the need to keep per-flow state. Around the same time, data structures similar to that employed in [2] have been proposed by Cohen and Matias for storing multisets (Spectral Bloom filters [3]), and by Cormode and Muthukrishnan for processing streaming data (Count-Min sketches [4]). Apart from the different terminology and specific implementation choices, all of these structures resemble counting extensions [5] of the original Bloom filter's multi-hash idea [6].

Roughly speaking, the common idea behind these approaches is to leverage a relatively small (compared to the total number of flows) number m of shared counters. When the goal is to measure the number of packets emitted by a given flow, at each packet occurrence, $k << m$ counters are incremented. These k counters are selected by means of k independent hash functions applied to the same specific flow identifier extracted from the packet. This guarantees that multiple packets belonging to the same flow will hash over the same counters. Obviously, the price to pay is an *approximate* counting operation, as different flows may hash over the same subset of counters.

The first author was partially supported by the European Commission under the FP7 framework, project DEMONS, contract no. 257315.

An important optimization of this basic approach, known as *Conservative Update* [2] or *Minimum Increase* [3], consists of updating only those counters which attain the minimum value among the k selected for a given flow identifier, thus avoiding unnecessary counter increases. This optimization has been experimentally shown to yield improved counting accuracy [2], [3], and it has been employed in several subsequent constructions [7]–[11].

Despite the huge interest in this conservative update optimization, we are not aware of prior work which has succeeded in providing an accurate analytical modeling of its operation. Indeed, analytic hurdles emerge from the fact that, with the conservative update rule, a counter selected by an hash function is updated on the basis of the status of the *remaining selected counters*; in other words, the counters do not independently increase, but do so in a tightly correlated manner. To the best of our knowledge, [12] is the only related work which explicitly attempted to analytically model such an optimization. However, in order to apply known results in the area of probability models for shortest queue joins [13], [14] (also known as the *Supermarket model* [15], [16]), work [12] models a *different* system, where i) *exactly one counter* is updated at each insertion, whereas *all* the minimum valued counters should be updated, and ii) counters are decremented with time. Obviously, such a twofold change in the system operation is crucial, as the aspect that mostly characterizes the system operation (as it will become clear later) is the *growth* rate of the counters, which in such a simplified system model is assumed known a priori and set to the lowest possible increase rate of one increased counter per insertion.

II. System model and notation

We model a count sketch as an array $B[1:m]$ of m counters (*bins*). Each bin contains s bits and hence can assume a value in the range $0 \cdots 2^s - 1$. Bins are accessed via k hash functions

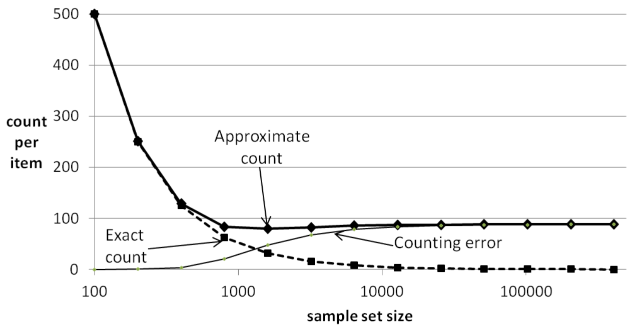

Fig. 1. Average count accuracy - uniform load scenario

$H_1(x) \cdots H_k(x)$, each of which maps an item x to one of the m bins within the bin array. We assume that each hash function extracts a digest value in the full-domain[1] range $[1 : m]$. Being interested in the case $m \to \infty$, we may neglect *collisions* among the k hash functions computed for a *same* item x, i.e. we assume $H_1(x) \neq H_2(x) \neq \cdots \neq H_k(x)$. Moreover, unless otherwise specified, for simplicity of notation we assume unbounded counter size, i.e. $s = \infty$.

Using the Conservative Update [2] optimization, *Increasing the count* for an item x consists of first determining the minimum value

$$C_{\min}(x) = \min_{i \in \{1 \cdots k\}} B[H_i(x)] \tag{1}$$

among the bins corresponding to the item x, and then incrementing by unity *only* the bins currently equal to $C_{\min}(x)$, i.e., $\forall i \in \{1 \cdots k\}$,

$$B[H_i(x)] \leftarrow \max\{B[H_i(x)], C_{\min}(x)+1\}.$$

Querying an accumulated quantity stored in the filter for an item x consists of just computing, as per Equation (1), the minimum value $C_{\min}(x)$ among the relevant bins.

III. Modeling assumptions and their practicality

The major contribution of this work is an *analytical approach which tightly captures the time evolution of the counters* composing the data structure. Our model takes advantage of a fluid approximation, which permits the system dynamics to be expressed in terms of ordinary differential equations.

To be tractable, the proposed model leverages two key assumptions. One is quite natural and consists of assuming a large number of counters; results show that very high accuracy is reached with just a few thousand counters, which is quite a small value for most real-world applications. The second appears less straightforward, and consists of assuming a *uniform* load across all counters. On the face of it, such an assumption is unrealistic for *any* scenario of practical interest: count sketches are mostly useful in contexts where flows have *different* statistics (e.g. for detecting heavy hitters). However, quite surprisingly, and perhaps unexpectedly, we find that the

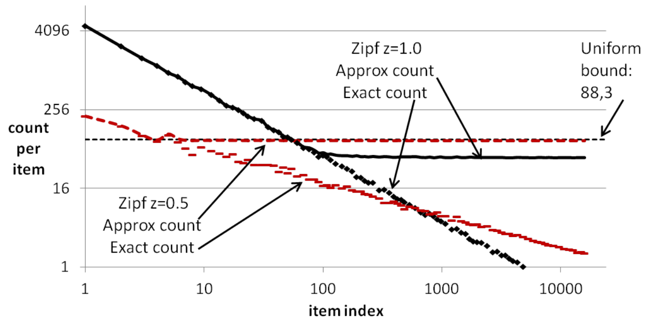

Fig. 2. Average count accuracy - skewed load scenario

results produced under the uniform load assumption yield crucial *practical* insights, duly applicable to the more general traffic scenario.

In support of this statement we present two different sets of simulations. Figure 1 presents data for scenario with uniform load and 50.000 arrivals (the trends do not change with a different total number of arrivals). Accounted items are randomly drawn from a set whose size, given on the x-axis, ranges from 100 to about 400.000 (note that in this last case only about 1/8 of the items will actually arrive and be counted). Averaging results over all the items, the plot reports i) the exact count per item, ii) the approximate count per item with a count sketch using $k = 4$ hash functions and $m = 1024$ bins, and iii) the resulting error, measured as the difference between the approximate and the exact count. It can be seen that once the the number of concurrently tracked items approaches and exceeds the filter size, the counting error increases. In more detail, when the number of items becomes much larger than the filter size, the approximate count *converges to a constant value*, irrespective of the actual exact count. For instance, given 50.000 total arrivals, with a set size of 10.000, one would expect an average exact count of about 5 arrivals per item, whereas the approximate count is as much as 88 arrivals per item. With 400.000 items, the approximate count remains about 88 arrivals per item even though the exact average count is as low as 0.125, i.e. on average seven out of eight items have not even arrived.

Figure 2 present results for the same sketch configuration ($k = 4, m = 1024$), but with a skewed distribution of items, rather than a uniform distribution. For a given sample set size (20.000 items in total), the figure reports the average approximate and exact count per item versus the item index. Items are sorted according to their relative frequency of arrival, item 1 being the most popular. The figure reports results obtained for two different Zipf distributions[2], with shape parameter $z = 0.5$ and $z = 1.0$. In both cases, two clearly distinguishable regions can be seen. Namely, less popular items, irrespective of their exact count, are approximated with the *same* value; conversely, the most popular items are accurately counted. In other words, the conservative update rule not only significantly reduces the

[1]We refer to, e.g., [5] for a discussion on the alternative, *partitioned*, organization, where the k hash functions access partitioned subsets of m/k bins; it is routine exercise to adapt our analysis to such case.

[2]In the Zipf distribution, being z a *shape* parameter, the frequency of item with index k is proportional to $1/k^z$. The greater z, the more skewed the distribution. For $z = 0$, the Zipf distribution converges to the uniform one.

counting error, but actually shapes it so as to "waterfall" all of the less popular items to the same level, which we refer to as the *error floor*. Whereas it is well known that, under the conservative update rule, the counting error experienced by the most popular items is lower than that of less popular ones, to the best of our knowledge ours is the first paper which clearly points out the emergence of this clear-cut waterfall-type shape in the error performance highlighted in figure 2.

Quantifying the error floor is a of considerable practical importance since it determines the average response level around which the sketch output becomes inaccurate. Importantly, Figure 2 indicates that the error floor in the case of a uniform load distribution appears to provide an upper bound on the error floor for skewed distributions. It is this property which makes analysis of the uniform load case of practical relevance.

IV. Analytical Model

A. *state space*

We assume a discrete time scale where time is clocked by the arrival of items, i.e., time $t \in \{1, 2, \cdots\}$ is defined as the time of arrival of the t-th element. An *obvious* modeling approach would consist in tracking the status $b_i(t)$ of each bin $B[i]$ versus time (an example is shown in Figure 3). However, with this choice the dimension of the state space grows rapidly with the number of bins m. Instead, leveraging the assumption of uniform traffic load (i.e., bins being statistically undistinguishable), we adopt an *alternative* state space description, which consists of tracking the *number* of bins (in essence, interpreted as queues, see Figure 4) whose level *exceeds* a threshold parameter, irrespective of their specific position i in the filter. On the face of it, this choice seems to increase modeling complexity as the number of dimensions in the state vector becomes infinite (since the bin size s is unbounded). In practice, however, the opposite holds true: as shown later on, this state representation is the key to achieving a convenient and tractable model.

More precisely, we represent the status of the filter at an arbitrary discrete time t by means of the following unbounded state vector:

$$\vec{d}(t) = \{d_0(t), d_1(t), d_2(t), \cdots, d_n(t), \cdots\} \tag{2}$$

where $d_n(t)$ is defined as the number of bins whose level (occupancy) is *greater or equal* than n. For example, at time $t = 0$, an empty filter will have state $\vec{d}(0) = \{m, 0, 0, 0, 0, \cdots\}$, as no bin has value greater or equal than $1, 2, 3$, etc, whereas obviously $d_0(0) = m$, since all the m bins have value greater or equal than 0 (which holds for any time t - indeed the component $d_0(t)$ is included in the state representation just for notational convenience). Figure 4 presents an example of a non empty filter comprising $m = 8$ bins, which we represent by means of the (infinite) state vector $\vec{d}(t) = \{8, 7, 5, 4, 3, 3, 2, 0, 0, 0, \cdots\}$.

For this state space model, the following structural properties trivially hold:

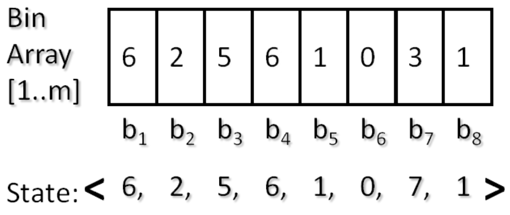

Fig. 3. Obvious state representation: $\langle \cdots, \text{value of bin } i, \cdots \rangle$, $i \in [1, m]$

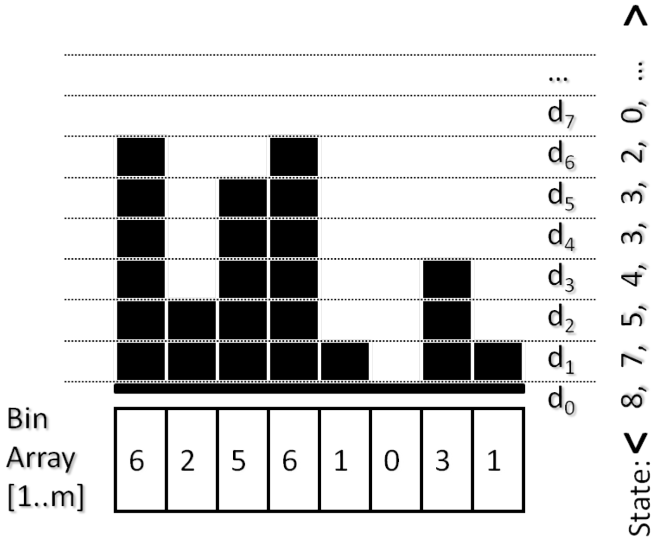

Fig. 4. Alternative state representation: $\langle \cdots, \#\text{bins} \geq \text{n}, \cdots \rangle$, $n \in [0, \infty]$

- $d_n(t)$ is defined for all $n \in (0, \infty)$, i.e., n is unbounded (under our assumption of unbounded counter size s);
- $d_n(t)$ is bounded and lies in the range $[0 : m]$;
- $d_{n+r}(t) \leq d_n(t)$ for any $r \geq 0$;
- $d_n(t + \Delta) \geq d_n(t)$ for any positive time interval Δ;
- the difference $d_n(t) - d_{n+1}(t)$ is the number of bins whose size at time t is *strictly equal to* n.

B. *Markov Chain*

Under the assumption of independent random arrivals, the stochastic process $\vec{d}(t)$ introduced in (2) is a discrete-time Markov chain. Thanks to the selected space state representation, this process exhibits the property that every state transition affects *only one state vector component (i.e. one "level" of bins' occupancy) at a time*. For example, if k=3, and an arrival hits bins whose level is 7, 7, 9, then, owing to the Conservative Update rule, only the two bins with level 7 will increment to 8. In turn, this implies that the number of bins whose level is greater or equal than 7, by definition d_7, does *not* change; the number of bins whose level is greater or equal than 8, namely d_8, will increase of two units, and the number of bins whose level is greater or equal than 9 again remains unaffected.

We now proceed in deriving the relevant time-dependent transition probabilities. When affected by a state transition, a level $n+1$, with $n \geq 0$ increases by a random variable x, lying in the range $[1 : k]$. In fact, at least one bin must increase upon an arrival, and at most k bins may increment. The resulting transition probability is given by:

$$P\left\{\vec{d}(t+1) = \{d_0, d_1, \cdots, d_n, d_{n+1}+x, d_{n+2}, \cdots\} \mid \vec{d}(t) = \{d_0, d_1, \cdots, d_n, d_{n+1}, d_{n+2}, \cdots\}\right\} = = \frac{\binom{d_n}{k}}{\binom{m}{k}} \cdot \frac{\binom{d_n - d_{n+1}}{x}\binom{d_{n+1}}{k-x}}{\binom{d_n}{k}} \tag{3}$$

where the first factor is the probability that all selected k bins have at least level n, and the second factor[3] is the probability that $x \geq 1$ of them have *exactly* level n (and where we on purpose do not make the obvious algebraic simplification, for reasons that will be clear below). We remark that this transition probability depends only on two state values: d_n and d_{n+1}; indeed this is a key fact exploited below for deriving a recursive system of differential equations.

C. Fluid approximation

First, note that, for large m, the first factor in equation (3), namely the probability that k selected bins are from the group d_n, is closely approximated by the case of independent extractions with reinsertions (or, alternatively, the same approximation may be derived via Stirling's formula):

$$\frac{\binom{d_n}{k}}{\binom{m}{k}} \approx \left(\frac{d_n}{m}\right)^k$$

Let us now derive the conditional expectation

$$E\left[d_{n+1}(t+1) - d_{n+1}(t) | \vec{d}(t)\right] = = \sum_{x=1}^{k} x \cdot \frac{\binom{d_n(t)}{k}}{\binom{m}{k}} \cdot \frac{\binom{d_n(t)-d_{n+1}(t)}{x}\binom{d_{n+1}(t)}{k-x}}{\binom{d_n(t)}{k}} = = \frac{\binom{d_n(t)}{k}}{\binom{m}{k}} \cdot k \cdot \frac{d_n(t) - d_{n+1}(t)}{d_n(t)} \approx \approx \left(\frac{d_n(t)}{m}\right)^k \cdot k \cdot \left(1 - \frac{d_{n+1}(t)}{d_n(t)}\right) \tag{4}$$

We now introduce a new stochastic process which is a rescaled version of (2) in terms of both state (normalized with respect to m, i.e., a *density process* [17]) as well as time (normalized with respect to m/k):

$$\vec{\psi}(\tau) = \frac{\vec{d}\left(\tau \frac{m}{k}\right)}{m}$$

The conditional expectation (4) can be rewritten for such rescaled process as:

$$E\left[m \cdot \psi_{n+1}(\tau + k/m) - m \cdot \psi_{n+1}(\tau) | \vec{\psi}(\tau)\right] = = \psi_n(\tau)^k \cdot k \cdot \left(1 - \frac{\psi_{n+1}(\tau)}{\psi_n(\tau)}\right) \tag{5}$$

[3]To avoid cumbersome details concerning the actual possible range of x, we adopt in equation (3) the notational assumption that $\binom{a}{b} = 0$ when $b > a$.

which can be rearranged as

$$\frac{E\left[\psi_{n+1}(\tau + k/m) - \psi_{n+1}(\tau) | \vec{\psi}(\tau)\right]}{k/m} = = \psi_n(\tau)^k \left(1 - \frac{\psi_{n+1}(\tau)}{\psi_n(\tau)}\right) \tag{6}$$

Besides the infinite dimensionality of the state vector, the density process $\vec{\psi}(\tau)$ would otherwise follow the assumptions required in [18] so that, for large m, the density process $\vec{\psi}(\tau)$ converges in probability to a deterministic trajectory. In practice, this implies that convergence holds for any arbitrarily large finite choice of the counter size s[4]. This deterministic trajectory can be computed, for large m, by replacing the left side of equation (6) with the derivative:

$$\psi'_{n+1}(\tau) = \psi_n(\tau)^k \left(1 - \frac{\psi_{n+1}(\tau)}{\psi_n(\tau)}\right) \tag{7}$$

D. Differential system

The above derived expression (7) is the main result of this paper. It holds for every $n \geq 0$ and for large values of m (relative to a constant, typically small, k). It defines a system of first-order differential equations, which can be recursively solved for increasing values of n. Indeed, the equation that governs the dynamics for the level $n+1$ does depend only on the previous level n. The boundary conditions are:

- $\psi_0(\tau) = 1$ for all τ; we recall that all bins are greater or equal than 0 for the whole time evolution of the system;
- $\psi_n(0) = 0$ for $n > 0$; we assume that the count sketch starts from the empty state.

Unfortunately, even if the differential equations are linear, their time-dependent coefficients do not permit us to obtain a handy closed form solution, and thus we resort to numerical solution. The only exception is the very first equation

$$\psi'_1(\tau) = \psi_0(\tau)^k \left(1 - \frac{\psi_1(\tau)}{\psi_0(\tau)}\right) = 1 - \psi_1(\tau)$$

which has the straightforward solution $\psi_1(\tau) = 1 - e^{-\tau}$.

Figure 5 plots, from left to right, the solutions $\psi_n(\tau)$ of the system versus (normalized) time, for the case $k = 3$. We recall that $\psi_n(\tau)$ represents the percentage of counters whose size is greater or equal than n. As expected, it can be seen that for a given n the percentage $\psi_n(\tau)$ increases with time and converges to 1, meaning that almost all of the counters have a size larger than n. Perhaps less expected, it can also be seen that as n grows the shape of the curves $\psi_n(\tau)$ appear to converge towards an invariant. Also aided by numerical results (minimum square difference between the translated curves significantly reducing), we conjecture that

[4]In practice, the size s, in bits, of the counters deployed in the filter limits the state space representation to 2^s maximum dimensions. Recalling the definitions and discussion carried out in Section IV-A, we remark that a finite-size sketch would be modeled by a density process $\vec{\psi}_s(\tau)$ which is simply the truncation of $\vec{\psi}(\tau)$ after its component with index $2^s - 1$, i.e. $\vec{\psi}_s(\tau)$ does *not* differ from $\vec{\psi}(\tau)$ in any of its first 2^s components.

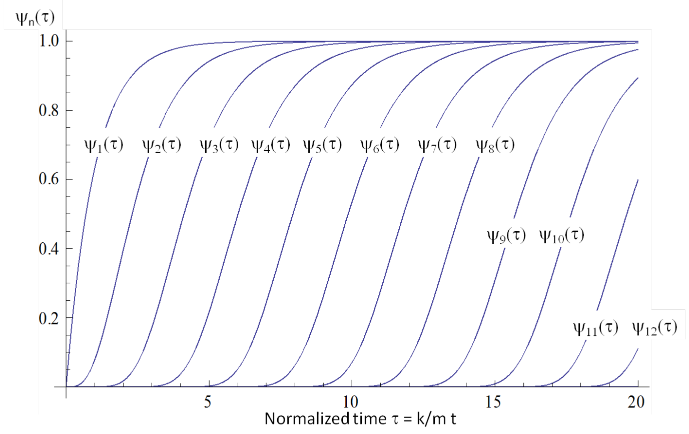

Fig. 5. Numerical solution of the differential system $\psi_n(\tau)$.

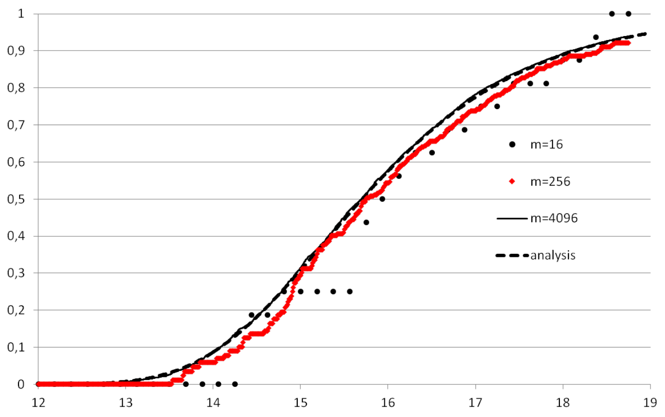

Fig. 6. $\psi_9(\tau)$: comparison with simulation results, $m = 16, 256, 4096$.

this may hold[5] for $k \geq 2$, although at the time of writing we lack a formal proof.

To validate the accuracy of the analytical solution, Figure 6 compares the analytical results with simulations. To avoid overcrowding the plot, we focus on the case of $\psi_9(\tau)$. Simulation results are obtained by plotting, versus (normalized) time, the fraction of counters whose level is greater or equal than 9, for the cases $m = 16, 256$ and 4096 (results obtained for larger values of m further get closer to the analytical predictions). As m increases, it can be seen that the agreement between analysis and simulation is impressive, thus helping to confirm the extremely good accuracy of the fluid approximation proposed here. Not only does the analysis closely follow the shape of the simulation curve (for $m = 4096$), but it also perfectly matches its time lag (we on purpose selected an intermediate value $n = 9$ instead of a very small one).

V. Growth Rate and Error Floor

The model introduced in Section IV allows us to address at least two important *practical* questions: i) what is the growth rate of the count sketch with Conservative update when subject to uniform load, and ii) what is the expected error floor (as defined in section III).

[5]for the case $k = 1$ it is easy to show that convergence does not hold, by direct solution of the differential system - immediate to solve with $k = 1$. Indeed, this was expected as the counters act as independent queues.

A. Growth dynamics

Let us define the filter's *growth rate* $g(t)$ as the average number of units added to the (whole) filter by the t-th insertion. In ordinary count sketches *not* employing the conservative update optimization, $g(t) = k$, as every insertion increases of one unit *each* of the k selected bins, irrespective of the considered t-th insertion. We further conveniently define with

$$G(t) = \frac{1}{m}\int_0^t g(\theta)\mathrm{d}\theta$$

the total accumulated quantity in the filter at the t-th insertion, normalized with respect to the number m of deployed bins. For an ordinary count sketch, $G(t) = kt/m$ obviously holds.

$G(t)$ can be somewhat considered a rough indicator (see Section V-C for a more precise performance metric) of the level of inaccuracy that the filter exhibits for an increasing number t of inserted random items. For instance, in the case of a filter with $k = 4$ hash functions and $m = 1024$ bins, after $t = 4096$ insertions the sum of all the counter values is $kt = 16,384$, which implies that a new item *first* arriving to the filter at time t finds each of the selected k bins filled around the average level $G(4096) = 4096 \cdot 4/1024 = 16$.

With the conservative update optimization, instead of being *strictly equal to* k, the number of counters increased at each insertion is *at most* k, and can be as little as one[6] in the best case of the minimum value attained by just one among the k selected counters. Quantifying the growth rate and understand whether it is closer to the worst case of k units/insertion, or to the best case of just 1 unit/insertion is thus meaningful, and would be indeed crucial to dimension the filter size which guarantees stability for periodically decremented count sketch constructions [10].

Referring to the normalized time scale $\tau = tk/m$ introduced in the previous section IV-C, it is straightforward to show that

$$G(\tau) = \sum_{i=1}^{\infty} \psi_n(\tau). \tag{8}$$

Indeed, this expression also has a very intuitive graphical explanation by looking at figure 4: the total units accumulated in the whole filter (namely, the number of black bricks in the figure) is given by the sum of the vector state components $d_1 + d_2 + \cdots$. By normalizing with respect to m, and making explicit the dependence on time, we recognize equation (8).

Figure 7 plots $G(\tau)$ versus (normalized) time, for a filter using the conservative update optimization with $k = 300$ hash functions. For sake of comparison, the growth behavior for the case of the filter without conservative update is also reported (owing to the normalized time scale, $G(\tau) = \tau$). The unusually large value k chosen in the plot permits to clearly appreciate growth fluctuations which would have likely remained unnoticed with a more practical (small) k. For the conservative update case, a simulation trace is also plotted in the same figure, using a filter of size $m = 2^{16}$. Since

[6]By construction, this would be the *a-priori* growth rate if we were to model the conservative update operation using the approach presented in [12].

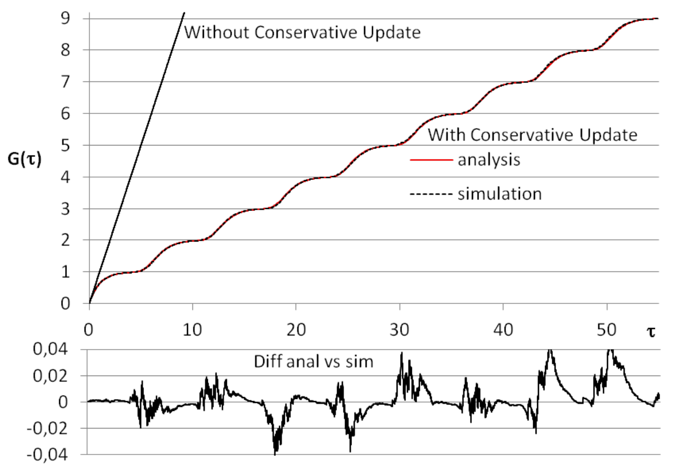

Fig. 7. Top: growth fluctuations with Conservative Update ($k = 300$); bottom: detailed difference between analysis and simulation

the simulation trace is hardly distinguishible from analytical results, the bottom plot reports the detailed difference between analysis and simulation versus time: besides the remarkable accuracy (error only occasionally exceeding $\pm$ 0.02 - we stress that the simulation is a single run, not an average of multiple runs), the comparison does not exhibits any drift as time elapses (error still close to zero at the end of the plot).

B. Average growth rate

Figure 7 shows that the growth rate for a count sketch with conservative update may significantly depend upon time. This was expected, at least at the start of the process: when most/all bins are empty, it is likely that all k bins selected by an insertion are empty, and thus they are all updated. Perhaps, it was less expected, at least to us[7], that the filter apparently fails to converge to a stable (relative) distribution: results extended to longer time (not shown for reasons of space) in fact suggest that such fluctuations are structural in the growth dynamic, i.e. they do not reduce with time; rather, they do appear to stabilize to a periodic pattern. This is consistent with the conjecture we anticipated while discussing figure 5.

Despite this, in practice it may be convenient to summarize, also in the conservative update case, the growth rate with a single, easily understandable, *average* parameter, analogous to the k units/insertion growth rate encountered for an ordinary count sketch. Since the direct computation of such average growth rate $\bar{g}$ as

$$\bar{g} = \lim_{\tau\to\infty} k \cdot \frac{G(\tau)}{\tau}$$

encounters slow convergence issues and requires to compute $\psi_n(\tau)$ for large values n, we resort to an alternative approach which exploits the quasi-periodicity of the growth behavior.

We first remark that $\psi_n(\tau)$ has a very convenient *alternative* interpretation. Let us define a continuous random variable X_n

[7]Indeed, our first (failed) modeling attempt was targeted at determining a steady state distribution of the filter's bins relative to the maximum value among all bins. A posteriori, the growth dynamic documented in Figure 7 somewhat justifies why such apparently straightforward modeling approach was not as successful as we expected.

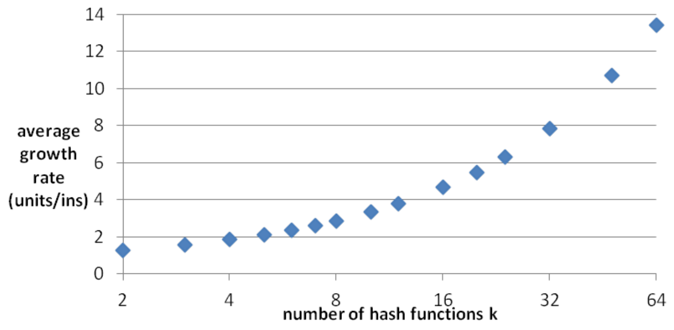

Fig. 8. Filter's growth rate (units per insertion) for varying k.

as the (normalized) time elapsing until a randomly chosen bin reaches level n. It is easy to see that $\psi_n(\tau)$ is indeed the cumulative distribution function for such random variable X_n. Hence, from the knowledge of $\psi_n(\tau)$, we can compute the expected value of X_n as

$$E[X_n] = \int_0^\infty (1 - \psi_n(\tau))\,\mathrm{d}\tau$$

Now, define

$$D_n = (E[X_n] - E[X_{n-1}]) \cdot \frac{m}{k}.$$

D_n is the average time, measured in number of insertions (the factor m/k restores the original time scale), for each bin starting at level $n-1$ to reach level n, i.e. increase its size of one unit. Since there are m bins, the average growth rate $\bar{g}_n$ during the step $(n-1) \to n$ is expressed as

$$\bar{g}_n = \frac{m}{D_n} = \frac{k}{E[X_n] - E[X_{n-1}]}$$

The computation of each term $\bar{g}_n$ relies on just the knowledge of the functions up to $\psi_n(\tau)$. Moreover, numerical results show that convergence is very fast[8]. Figure 8 shows the average growth rate for varying values of the number of hash functions k. With two hash functions, the rate is about 1.61, meaning that, on average, 61% of the insertions update both selected bins whereas only the remaining 39% update just one bin. As the number of hash function grows, the efficiency in the bins' update improves. With $k = 3$ each update increases approximately half of the selected bins (more precisely, 1.53), whereas with $k = 10$ only about one third (3.31) are increased.

C. Error Floor

As discussed in section III, the average error floor is a convenient *single-value* quantification of the worst-case average additive error introduced by the filter operation. Results in section III have shown that a same average error floor applies to low popularity items, irrespective of their exact count. Hence, the error floor can be easily quantified by querying the filter with *new* items, i.e. not earlier accounted.

[8]in the worst case of $k = 2$, convergence expressed as a relative difference lower than 10^{-7} occurs after about $n = 50$ terms whereas with, say, $k = 8$ convergence is attained after less than 20 terms.

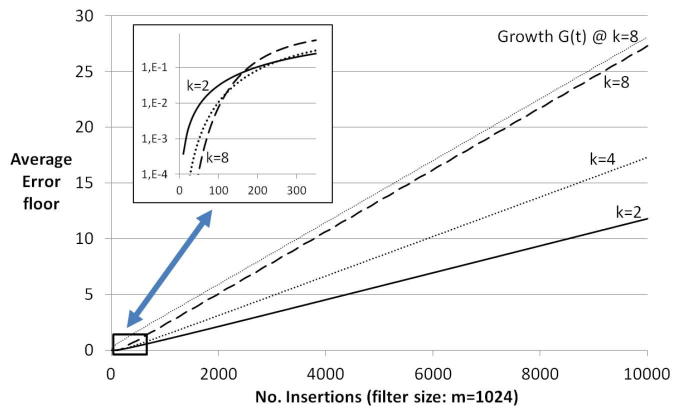

Fig. 9. Average error floor versus number of insertions for varying k.

For a randomly chosen element, a query to the filter done at time τ consists of selecting k random counters, and determine their minimum value $Z_{\min}(\tau)$. Owing to the discussion carried out in section V-B, $\psi_n(\tau)$ is the probability that a randomly chosen counter has size greater or equal than n at time τ. Hence, the probability that *all* k selected counters are greater than n, namely the complementary cumulative distribution function of the r.v. $Z_{\min}(\tau)$, is $\psi_{n+1}(\tau)^k$, and the expected value returned by such a query is thus

$$E[Z_{\min}(\tau)] = \sum_{n=1}^{\infty} \psi_n(\tau)^k \tag{9}$$

Figure 9 plots the average error floor, formula (9), using a non-normalized time scale, with each time unit representing an insertion in the filter (i.e., $t = \tau \cdot m/k$). For concreteness, we use $m = 1024$ for the filter size, but we remark that the parameter m does not affect results but only the time scale.

The curve confirms that, as time elapses, a larger number k of hash yields a greater average error floor. This is well expected from our previous discussion on the average growth rate: the larger k, the greater the number of increments per insertion, the faster the filter fills up. As an exception, when the number of insertions is small (say, below 1/3 of the filter size m, e.g., the zoom plot in figure 9 - note the logarithmic y-axis scale), the opposite holds: the error floor is lower for a *larger* value k. The obvious explanation is that when most filter counters are still empty, a larger k increases the chance that the minimum value among such selected counters is zero as well.

Finally, in the case $k = 8$, the plot further includes the average counter size $G(t)$. Interestingly, $G(t)$ always remains quite close to the error floor (difference never greater than 0.85 in the case $k = 8$, and slightly smaller in the remaining cases). The reason is that the increase, for larger k, in the number of counters from which the minimum is attained, trades off with a decrease in the spread of the counter occupancy distribution (the slope of the $\psi_n(.)$ distributions becoming steeper).

This last remark is *very convenient for practical purposes*. Indeed, knowledge of the summary values reported in figure 8 permits to tightly bound (within one unit in excess) the error

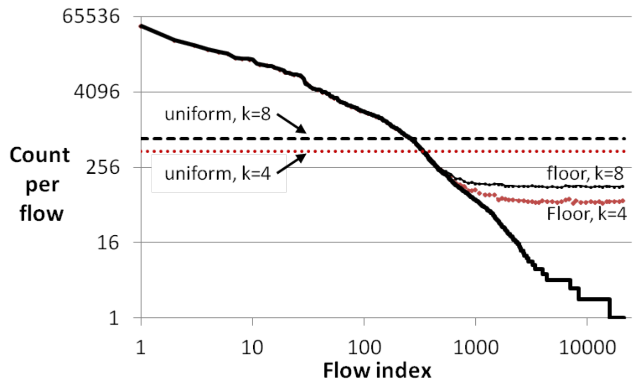

Fig. 10. Experimental results: average error floor, $m = 4096$, $k = 4, 8$.

floor in the uniform case *with no need to solve any differential equation*! For instance, with reference to the experimental setting discussed in the next section VI (see figure 10) involving 1,050,000 packets inserted in the filter, the error floor uniform case bound would be $1.816 \times 1.05 \cdot 10^6/4096 \approx 466$ for the case $k = 4$, and $2.843 \times 1.05 \cdot 10^6/4096 \approx 729$ for $k = 8$.

VI. Conclusions and next steps

The contribution of this paper is mostly theoretical, and focuses on a new analytic approach to modelling count sketches with conservative update. By unveiling how the error behaves in such data structures, we believe that the analysis paves the ground for new insights into how we can practically handle, and dimension, such count sketches in real traffic scenarios. However, we leave this to future work. That said, and with no pretense of making any *claim*, we feel it may be useful to finish by highlighting a number of practical aspects which, to the best of our knowledge, are first raised here and which build on this paper's findings.

A. Experimental playground

To shed some light on the problems emerging in practical situations, we have run experimental results on a traffic trace collected from a Point-Of-Presence backbone of a relatively small (regional) operator. We have specifically taken a 5 minute traffic trace, including about 1,050,000 packets. For test purposes, we counted the number of packets per IP source-destination pair (hereafter somewhat improperly referred to as a flow). We found 21,390 distinct flows in the trace. The exact per-flow count is reported in Figure 10, tick solid line. The figure also reports the *measured* error floor when using count sketches having a very compact size (as little as $m = 4096$ for a trace with more a million packets and with the number of flows more than 5 times the available bins). Two sketch configurations, in terms of number of hash functions employed, are considered: $k = 4$ and $k = 8$. Results are obtained by *first* adding all of the packet trace to the filter, and *then*, at the end of the trace, by querying each flow found in the trace to determine its relevant accumulated count.

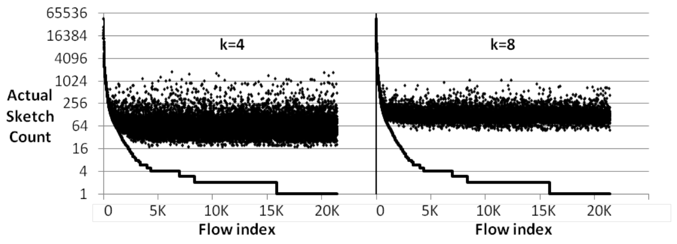

Fig. 11. Experimental results: raw per-flow sketch response.

B. Closing the gap between uniform and actual error floor

Figure 10 confirms that the average error floor computed using our proposed approach (uniform assumption) provides an upper bound, but it also shows that this bound is quite loose. For instance, with $k = 8$, the uniform analysis yields a bound of 728.4 versus an actual error floor of 127.2, obtained by averaging the returned counts for all the flows but the first top-1000. There is an obvious reason for such a significant overestimate: whereas the uniform model assumes that every new arrival, by randomly selecting k bins, contributes to increase the average counters occupancy, heavy flows will repeatedly hit the *same* bins.

This suggests that a tighter error floor estimation algorithm might suitably discount packets generated by flows whose count is "large", and hence which only affect a subset of bins. For instance, if we were to compute the uniform bound using only the 303,910 packets remaining once removing the 746,090 packets generated by the 206 flows accounting for at least 1000 packets (hence well above the uniform floor level in figure 10), we would obtain the significantly lower value of 211. We believe that iterative online algorithms somewhat mimicking (and of course improving) this suggested heuristic may be devised.

C. Error floor: beyond averages

The results earlier presented in figure 9 suggest that, in practical settings, a small value of k is preferable. Such a conclusion, based as it is on the *average* error floor, might be misleading when accounting for the actual error distribution. For instance, figure 11 (and, more precisely, the complementary cumulative distribution functions shown in figure 12 obtained for all the flows but the first top-1000) clearly shows that $k = 8$ may be a preferred filter setting to reduce the probability that a query for a "small" flow makes it look like a "large" count one. This implies that, in addition to the *average* error floor value, a thorough dimensioning and practical usage should be guided by some criterion that allows us to determine to what extent the response of the filter when queried with a *specific* flow entry is reliable or meaningless. Note that the usual Bloom filter formulae may not be directly applicable to model this case.

D. From packet counts to byte counts

Finally, a further challenge consists in adapting our proposed modeling approach to the more general byte-count scenario [2]. This appears far from being simple. But, perhaps,

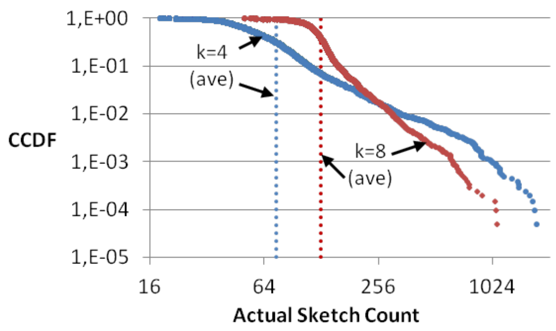

Fig. 12. Experimental results: per-flow approximate count for small flows - complementary cumulative distribution function.

some findings may be extended to such more general scenarios (for instance the error behaviour may indeed be qualitatively the same – experimental results not shown for space reasons).

References

[1] A. Kirsch, M. Mitzenmacher, and G. Varghese, "Hash-based techniques for high-speed packet processing," in *Algorithms for Next Generation Networks*, ser. Computer Communications and Networks, G. Cormode and M. Thottan, Eds. Springer London, 2010, pp. 181–218.

[2] C. Estan and G. Varghese, "New directions in traffic measurement and accounting," *SIGCOMM Comput. Commun. Rev.*, vol. 32, pp. 323–336, August 2002.

[3] S. Cohen and Y. Matias, "Spectral bloom filters," in *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, ser. SIGMOD '03. New York, NY, USA: ACM, 2003, pp. 241–252.

[4] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *J. Algorithms*, vol. 55, pp. 58–75, April 2005.

[5] A. Broder and M. Mitzenmacher, "Network applications of bloom filters: A survey," in *Internet Mathematics*, 2002, pp. 636–646.

[6] B. H. Bloom, "Space/time trade-offs in hash coding with allowable errors," *Commun. ACM*, vol. 13, pp. 422–426, July 1970.

[7] S. Narayanasamy, T. Sherwood, S. Sair, B. Calder, and G. Varghese, "Catching accurate profiles in hardware," in *Proc. 9th Int. Symp. on High-Performance Computer Architecture (HPCA '03*, 2003, pp. 269–.

[8] P. Kolaczkowski, "Memory efficient algorithm for mining recent frequent items in a stream," in *Rough Sets and Intelligent Systems Paradigms*, ser. Lecture Notes in Computer Science, M. Kryszkiewicz, J. Peters, H. Rybinski, and A. Skowron, Eds. Springer Berlin / Heidelberg, 2007, vol. 4585, pp. 485–494.

[9] F. Raspall and S. Sallent, "Adaptive shared-state sampling," in *Proceedings of the 8th ACM SIGCOMM conference on Internet measurement*, ser. IMC '08. New York, NY, USA: ACM, 2008, pp. 271–284.

[10] G. Bianchi, E. Boschi, S. Teofili, and B. Trammell, "Measurement data reduction through variation rate metering." in *INFOCOM'10*, 2010, pp. 2187–2195.

[11] A. Goyal, J. Jagarlamudi, H. Daumé, III, and S. Venkatasubramanian, "Sketching techniques for large scale nlp," in *Proc. of the NAACL HLT 2010 Sixth Web as Corpus Workshop, WAC-6 '10*, 2010, pp. 17–25.

[12] Y. Chabchoub, C. Fricker, and H. Mohamed, "Analysis of a bloom filter algorithm via the supermarket model," in *Teletraffic Congress, 2009. ITC 21 2009. 21st International*, sept. 2009, pp. 1 –8.

[13] N. Vvedenskaya, R. Dobrushin, and F. Karpelevich, "ueueing systems with selection of the shortest of two queues: an asymptotic approach," *Prob. Inf. Trans.*, vol. 32, pp. 15–27, 1996.

[14] C. Graham, "Chaoticity on path space for a queueing network with selection of the shortest queue among several," *J. Appl. Prob.*, vol. 37, pp. 198–211, 2000.

[15] M. Mitzenmacher, "The power of two choices in randomized load balancing," *PhD Dissertation, Berkeley*, 1996.

[16] M. J. Luczak and C. McDiarmid, "On the maximum queue length in the supermarket model," *Annals of Probability*, vol. 34, p. 493, 2006.

[17] F. M. Buckley and P. K. Pollett, "Limit theorems for discrete-time metapopulation models," *Probability Surveys*, vol. 7, pp. 53–83, 2010.

[18] R. W. R. Darlings and J. R. Norris, "Differential equation approximations for markov chains," *Probability Surveys*, vol. 5, pp. 37–79, 2008.