# Comparison between multistage filters and sketches for finding heavy hitters

Cristian Estan

March 19, 2004

## 1 Overview

The purpose of this write-up is to compare multistage filters [3] and sketches with respect to their ability to identify heavy hitters. In a nutshell, the conclusion is that multistage filters as I use them identify heavy hitters with *less memory than sketches*, but some sketches support important other operations, more specifically they can be added and subtracted without any need to re-read the data stream(s).

Both multistage filters and sketches work in the streaming model: data items with various identifiers make up a stream of traffic or updates that the data structures operate on. Both hash these updates to counters based on the identifiers. The main difference between sketches and multistage filters is that sketches see these counters as a summary of the traffic that can be used for many operations while filters use these counters only as a means for identifying the heavy hitters. After discussing the differences in more detail, I will compare the memory usage of these solutions for the heavy hitter problem.

Since there are many types of sketches and there are many things they are used for, I am going to focus on three papers that use sketches for detecting heavy hitters. [1] is a widely quoted paper that addresses exactly the problem of finding heavy hitters. [2] is a more recent paper that proposes improved sketches for finding heavy hitters. [4] applies sketches to detect big changes in network traffic, which is related to but different from finding heavy hitters. One thing worth pointing out is that many of the sketch papers papers (and all three discussed here) are concurrent with or published after my paper on multistage filters [3].

## 2 Comparison

Table 1 enumerates the main differences between multistage filters and various types of sketches. I will discuss each of these sketches separately and point out the differences.

The sketch in [4] is a multistage filter without conservative update and the other optimizations we have. It is used as a first block in detecting large changes. It does not have a stream memory as my solution, so *it does not actually give the identity of the heavy hitter*, but it can answer questions such as is X a heavy hitter or not. The advantage of this structure is that sketches can be added and subtracted and this allows easy change detection and various time series analyses (on series of sketches). The way in which the traffic of the heavy hitters is estimated is also different. With multistage filters we consider the smallest counter an identifier hashes to an upper bound on the size of its traffic and use it as the estimate that guides the creation of new entries in the "stream memory". Thus our estimate is biased

| Feature | Filters | [4] | [1] | [2] |
|---|---|---|---|---|
| Explicit set | "stream memory" | none | heap | none |
| Identity of h.h. | yes | no | yes | yes |
| Subtraction | no | yes | yes, extra cost | yes |
| Estimation method | minimum | median | median | not applicable |
| Bias | biased | unbiased | unbiased | not applicable |
| Memory usage | $O(k/\epsilon \log(n))$ | muddy | $O(k/\epsilon^2 \log(N/\delta)$ | $O(k \log(k/\delta) \log(m))$ |

Table 1: Main differences between sketches and multistage filters. In the memory usage formulas, $k$ is the ratio between the total traffic and the size of the heavy hitters, $\epsilon$ is the relative error in the traffic estimates, $\delta$ is the probability of failure, $N$ is the number of updates, $n$ is the number of distinct identifiers in the data and $m$ is the number of possible identifiers.

(the filter never underestimates). This sketch uses the median counter minus some correction factor, and their estimates are unbiased. The paper does have an analysis of the variance of the estimates, but the results depend on the distribution of stream sizes, and they give no rules for dimensioning the sketch so I cannot compare its memory usage with ours. However [1] is very similar and uses significantly more memory than multistage filters.

The sketch in [1] is very similar to the one from [4][1]. The estimate is the median of the counters. This sketch also has a heap with explicit entries for heavy hitters which operates very similarly to the "stream memory" used with multistage filters. Therefore, it can also give the actual identity of the heavy hitters since they are stored in the heap. One can subtract two sketches, but to also find the identities of the streams that change much, one needs a second traversal over the data. The memory cost scales as $1/\epsilon^2$ instead of $1/\epsilon$ as for the multistage filters, so this is a solution with much larger memory requirements. The basic reason is that they use a large number of counters in a stage to decrease the variance of the estimates (which decreases linearly with the number of counters) whereas we use the large number of counters to decrease the average size of the error linearly with the number of counters. The analysis of this sketch also contains a failure probability $\delta$. The results I presented do not capture the failure probability of multistage filters (which can only happen when the stream memory is full), but this does not affect the asymptotic memory requirements because we can reduce this probability by adding more "stream memory" and the correction factor is additive, not multiplicative because it does not affect the stages of counters (which dominate asymptoticly).

The sketch in [2] has a remarkable feature: one can compute the difference between two sketches and find the streams that changed much, no matter how small the total change is with respect to the total traffic described by the two sketches. I has no explicit set of heavy hitters (which would be useless in detecting change if the heavy hitters stay the same). It encodes the identity of the heavy hitters in the positions of the counters. I uses coding theory approaches to recover the identities of the heavy hitters from the positions of the large counters. The analysis of this sketch focuses on the probability that one can recover the identity of the heavy hitters. The paper does not talk at all about the error in estimating the traffic of these heavy hitters from the sketch, hence the $\epsilon$ is not in its asymptotic memory usage. My opinion is that one could add guarantees of the accuracy of the estimates at the cost of adding a multiplicative factor of $1/\epsilon$. The total memory usage of the sketch would be larger than that of multistage

[1] One difference is that the individual identifiers are also randomly mapped to a sign: some have their traffic added to the counters, some have it subtracted.

filters, but this sketch clearly provides significantly more functionality.

# References


[1] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *International Colloquium on Automata, Languages and Programming*, pages 693–703, 2002.

[2] Graham Cormode and S. Muthukrishnan. What's hot and what's not: Tracking most frequent items dynamically. In *Symposium on Principles of Database Systems*, pages 296–306, June 2003.

[3] Cristian Estan and George Varghese. New directions in traffic measurement and accounting. In *Proceedings of the ACM SIGCOMM*, August 2002.

[4] Balachander Krishnamurthy, Subhabrata Sen, Yin Zhang, and Yan Chen. Sketch-based change detection: Methods, evaluation, and applications. In *Internet Measurement Conference*, pages 234–247, October 2003.