Count-Min Sketch with Sampling Module

Author: Yuwei Xia (28267331)

Abstract

Count-Min sketch is a widely used method in the field of flow size estimation. It utilizes hash functions to map flows into a two-dimensional array, called sketch, and estimates flow size by finding the minimum value of counters in the sketch related to that flow. The problem of Count-Min sketch is that, due to the use of hash function, it is possible that different flows are mapped into the same counter, and thus increase the total estimation error. In this paper, we add a sampling probability to the recording procedure of Count-Min sketch and hope this sample module can help increase estimation accuracy of Count-Min sketch by reducing collisions. After conducting experiments on different sampling probabilities and different sketch sizes, the conclusion of our experiments is sample module does not work well in reducing estimation error of Count-Min sketch.

1 Introduction

Nowadays, because of the extensive use of internet, data is no longer as small and easy to analyze as before. Especially when it comes to internet data streaming, where properties of large streams of data should be extracted and analyzed to acquire certain information which could improve business. But how to store these flows so that query and update operations are efficient is a problem. Sketch, a data structure that can compactly deal with large volume of flows and support fast query and easy update, is utilized [8]. However, the number of distinct flows could be so big that make it difficult to remember the flow size of each flow precisely due to various challenges such as limited space of on-chip model and processing of all kinds of logs in off-chip model. To deal with this, a notable approach called Count-Min (CM) sketch, is proposed by Cormode and Muthukrishnan in 2005 [1]. CM sketch is a simple method which can estimate flow size using a relatively small space and guarantee high-efficiency update and query. Instead of estimating flow size, CM sketch can also be used in applications such as change detection and multidimensional streams [9].

In this paper, we change CM sketch method a little bit considering it records all flows without hesitation and might lead to lots of collisions under certain circumstances. Because hash functions are used to map each flow into the sketch, there is a high probability of collision when the number of distinct flow ids is much larger than the number of columns of sketch. Inspired by bitmap with sampling [3], we wonder if we only record a portion of flows, the accuracy of flow-size measurement will decrease or increase under a given memory allocation. A sampling probability between zero and one is utilized as the probability of mapping a flow into the sketch.

Main contributions of this paper are: 1) To reduce estimation error of CM sketch, we apply a sample module on recording procedure of CM sketch by adding a sampling probability; 2) Conduct several experiments on CM sketch with sampling by changing sketch size and sampling probability, and come to the conclusion that sample module does not improve estimation accuracy of CM sketch.

2 Background

In this section, we will first explain the flow model, including the definition of flow and the properties of flow such as flow size and flow spread. Then we will describe the problem we investigate precisely.

2.1 Flow Model

Flow can be defined as a sequence of packets passing across a network between two end points [2], a source and a destination. These two end points can be per-source and multi-destination, multi-source and per-destination as well as per-source and per-destination. Each flow contains a flow id and an element and is written as $< f, e >$, where f denotes flow id and e denotes element. For per-source flows, flow id is the source address and element is the destination address of the flow. Per-destination flows are the opposite: flow id is the destination address and element is the source address.

Each flow has two properties that worth analyzing, flow size and flow spread. Flow size means the number of elements in each flow. Flow spread is the number of distinct elements in each flow. To help get a better understanding the difference between flow size and flow spread, two examples on per-source flow will be given. First, consider a source A and a destination B. If A sends 1 flow which contains 10,000 packets to B, then the flow size is 10,000 and flow spread is 1. Then consider a source A and 100 destinations $B_1, B_2, \dots, B_{100}$. If A sends 1 packet to each $B_i (i \in [1,100])$, then flow size is 100, flow spread is also 100.

2.2 Problem definition

CM sketch is a general-purpose method that can be used to measure flow size of large number of potential flows under certain error bound. The data structure used by this method is sketch, a compact two-dimensional array which provides high-efficiency update and query. Usually, a sketch will not have too many rows and columns, or it might lose its advantage of being compact. Nevertheless, data stream might be huge and contains lots of flows. Majority flows are in small size, while a small portion of flows are large. When we calculate which counter to record a flow, hash functions are applied to map the flow id into the sketch. And here is the problem: different flows might be hashed into the same counter due to limited size of sketch. If there are too many collisions, the performance of CM will decrease and lead to huge overestimation error.

When using Bitmap method counting flow spread, sampling can be used to increase the number of zeros in the bitmap and further increase the accuracy of estimation. Rather than putting all packets into the bitmap, sampling module helps pick some packets with a certain probability p and thus decrease the density of ones of bitmap. So, if we add a sampling module on CM sketch, what will happen to the estimation accuracy?

3 Sketch Design

In this section, the original CM sketch method will first be defined and explained in detail. Then, CM will be modified by adding sampling module as the probability of a packet. This new method will also be described carefully.

Before the discussion of specific algorithms, there are some symbols and definitions need to be clarified in Chart 1.

Symbol	Definition
F	The set of flows.
f	Any flow in F.
n_f	The actual size of flow f.
N	The sum of size of flows in F.
$C[k][w]$	A sketch with k rows and w columns, initially set to all zeros.
$H_j()$	The jth random hash function, $j \in [0, k-1]$
$\hat{n_f}$	The estimation size of flow f.
n	Number of flows in F.
p	Sampling probability.

Chart 1: Symbols and definitions

3.1 Count-Min Sketch

Now that all symbols are clear, we can discuss about the procedures of CM sketch method. CM sketch includes three steps:

- Recording. For each single flow f in flow set F, we use each hash function to hash it and add the corresponding position in the sketch C by 1. To be more specific, for each hash function $H_j()$, we perform $H_j(f)$, and increase $C[j][H_j(f)]$ by 1. The recording algorithm is shown in Algorithm 1.
- Query. Query is quite similar to recording. First, for each flow f, we need to find values stored in all corresponding k counters. Then, we select the minimum value as the estimation flow size of f. The query algorithm is shown in Algorithm 2.
- Error calculation. Now we have obtained both actual flow size and estimated flow size, we can calculate average error of all flows in F. This can be done by formular:

$$\frac{\sum(\hat{n_f} - n_f)}{n}. \quad (1)$$

Algorithm 1 Recording

Input: the set of flow F, hash functions $H()$, sketch C
Output: sketch C after recording all flows in F

```
for(f in F):
    for( j = 0 to k − 1):
        C[j][H_j(f)] += n_f
    end for
end for
```

Algorithm 2 Query

Input: the set of flow F, hash functions $H()$, sketch C
Output: estimated size $\hat{n_f}$ of each flow f in F

```
for(f in F):
    for( j from 0 to k − 1):
        n̂_f = min(C[j][H_j(f)])
    end for
end for
```

3.2 Count-Min Sketch with Sampling

After understanding the algorithms of CountMin sketch, we can further discuss about adding sampling module on it. Note that now we have four steps:

- Sampling probability generation. In order to sample all the flows, we first need to generate a sampling probability p, so that for each flow, the probability of recording it in sketch is p. This also means only about a $1 / p$ packets of F are recorded in sketch.
- Recording. The recording procedure is quite similar with the original one. But instead of recording every flow in F, we record each flow with the sampling probability p. The algorithm is shown in Algorithm 3.
- Query. The different part of query procedure is when calculating estimation size of flow f, the acquired minimum value need to be scaled by multiplying $1 / p$. This algorithm is shown in Algorithm 4.
- Error calculation. Same as original method. The formular is also:

$$\frac{\sum(n\hat{}_f - n_f)}{n}. \tag{2}$$

Algorithm 3 Recording with Sampling

Input: the set of flow F, hash functions $H()$, sketch C, sampling probabilistic p
Output: sketch C after recording flows in F with a sampling probabilistic p

```
for(f in F):
    for(i from 0 to n_f − 1):
        if(random number ∈ [0,1) < p) :
            for( j from 0 to k − 1):
                C[j][H_j(f)] + +
            end for
        end if
    end for
end for
```

Algorithm 4 Query with Sampling

Input: the set of flow F, hash functions $H()$, sketch C, sampling probabilistic p
Output: estimated size $n\hat{}_f$ of each flow f in F

```
for(f in F) :
    for( j from 0 to k − 1):
        min_value = min(C[j][H_j(f)])
    end for
    n̂_f = min_value/p
end for
```

Although we have specific algorithms, there are still some technique problems that should be figured out before conducting the experiments. How do we implement k different hash functions? In this paper, we simply generate an array of k different large random numbers and perform XOR function on these numbers and flow-ids. Because the result might be larger than the column size of the sketch, it should modulo w to get the column index of the sketch. Despite of this, since flow id is a string, we cannot directly perform XOR operation on it. Hence, we need to use native hash function to hash flow id into an integer.

4 Analysis

In this section, error bound of CM sketch with sampling will be calculated. Indicator $I_{i,f,f'}$ denotes if a flow f and a different flow f' will be mapped into the same counter or not. If they have a collision, then $I_{i,f,f'}$ equals to 1, otherwise 0. This definition is shown below.

$$I_{i,f,f'} = \begin{cases} 1, & if\ H_i(f) = H_i(f')\ and\ f \neq f' \\ 0, & otherwise \end{cases} \tag{3}$$

We define $E(I_{i,f,f'})$, which is the probability of f colliding with f'. Since the sketch has w columns, the probability of collision within two distinct flows is $1/w$.

$$E(I_{i,f,f'}) = Probability\{H_i(f) = H_i(f')\} = \frac{1}{w} \tag{4}$$

Then we define the total noise caused by other flows to flow f as:

$$X_{i,f} = p * \sum_{f' \in F} I_{i,f,f'}\, n_{f'} \tag{5}$$

Because after sampling, we do not know the exact number of how many packets of a flow is sampled. We can roughly regard the average flow size as actual flow size of each flow, which is $p * n_f$. Hence, estimated flow size $C[i][H_i(f)]$ is approximately represented as:

$$C[i][H_i(f)] \approx (p * n_f + X_{i,f})/p \geq n_f \tag{6}$$

Expected total noise is:

$$E(X_{i,f}) = E\left(p * \sum_{f' \in F} I_{i,f,f'}\, n_{f'}\right) = p * \sum_{f' \in F} n_{f'}\, E(I_{i,f,f'}) = p * \frac{N - n_f}{w} \tag{7}$$

When N is much larger than the size of a single flow, n_f can be ignored. In this case, we get:

$$E(X_{i,f}) \approx \frac{p * N}{w} \tag{8}$$

Then we can calculate the error bound using Markov's inequality:

$$\begin{aligned} Probability\{\hat{n_f} - n_f > \varepsilon N\} &= Prob\{\forall i \in [1,k], C[i][H_i(f)] > n_f + \varepsilon N\} \\ &= Prob\left\{\forall i \in [1,k], \frac{X_{i,f}}{p} > \varepsilon N\right\} < e^{-k} \end{aligned} \tag{9}$$

5 Experiments

In this section, our experiments will be discussed thoroughly and carefully and results will be analyzed. These experiments are all conducted on macOS Catalina Version 10.15.7, with

2.3GHz Dual-Core Intel Core i5 processor and 8 GB 2133 MHz LPDDR3 memory. Codes are written in Java via IntelliJ IDEA Ultimate 2019.1.1 Edition.

The input data we use is a file containing 10,000 distinct flows and each flow has a certain packets number. This first line of the input file is the number of flows n, which is followed by n lines, where each line contains a unique flow id and the number of packets in the flow. All our experiments are based on this file. Part of the input file is provided in Appendix.

We conduct two experiments in this paper.

5.1 Experiment One

The first experiment uses fixed sketch size and only changes the value of p, which is the sampling probability, to see whether it has a negative or positive influence on estimation accuracy. The values of parameters k, w, p are displayed in Chart 2. We set k as 3 and w as 3000 is because these values look reasonable considering the number of distinct flow ids is 10,000. Note that when p is equal to 1.0, it means we are not sampling at all. We use this situation as reference to see whether sampling module is useful. Hash functions remain the same with all different p values.

Parameters	Values
k	3
w	3000
p	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}

Chart 2: Parameters and values of the first experiment

The result of experiment one is shown in Figure 1. For each sampling probability p, we run the program 50 times to obtain an average estimation error. The horizontal axis is sampling probability p and the vertical axis is average estimation error of corresponding p after query. The dotted line in blue is the trend line. This figure clearly denotes that sampling module does not help increase the estimation accuracy of CM sketch. To be more specific, it even negatively affects original CM sketch method.

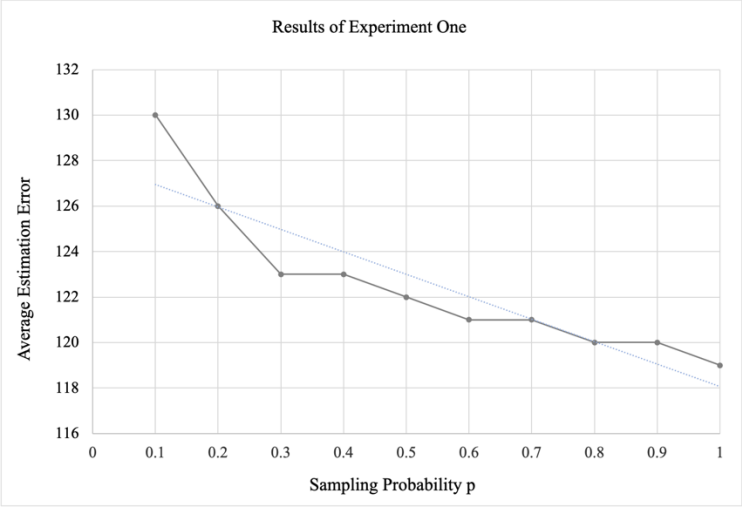

Figure 1: Results of experiment 1

5.2 Experiment Two

In the second experiment, to better find out whether sampling module will work in CM sketch, we further alter the size of sketch by increasing or decreasing w and k. The values of parameters k, w, p of experiment two are displayed in Chart 3.

Parameter	Values
k	{2, 3, 4, 5}
w	{2000, 2500, 3000, 3500, 4000}
p	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}

Chart 3: Parameters and values of the first experiment

The results of experiment two are demonstrated in Figure 2. In order to reduce random error in our results, we run each value combination of (k, w, p) 50 times to obtain an average estimation error. The horizontal axis is the sampling probability p and the vertical axis is average estimation error. Each line corresponds to a specific sketch size and each dot on the line denotes the average estimation error under certain p and certain sketch size. These figures show that although we alter the sketch size by changing the value of k and w, the sample module still does not work well. Or what we can say is, sampling probability p even has a slightly negative effect on the estimation accuracy of CM sketch.

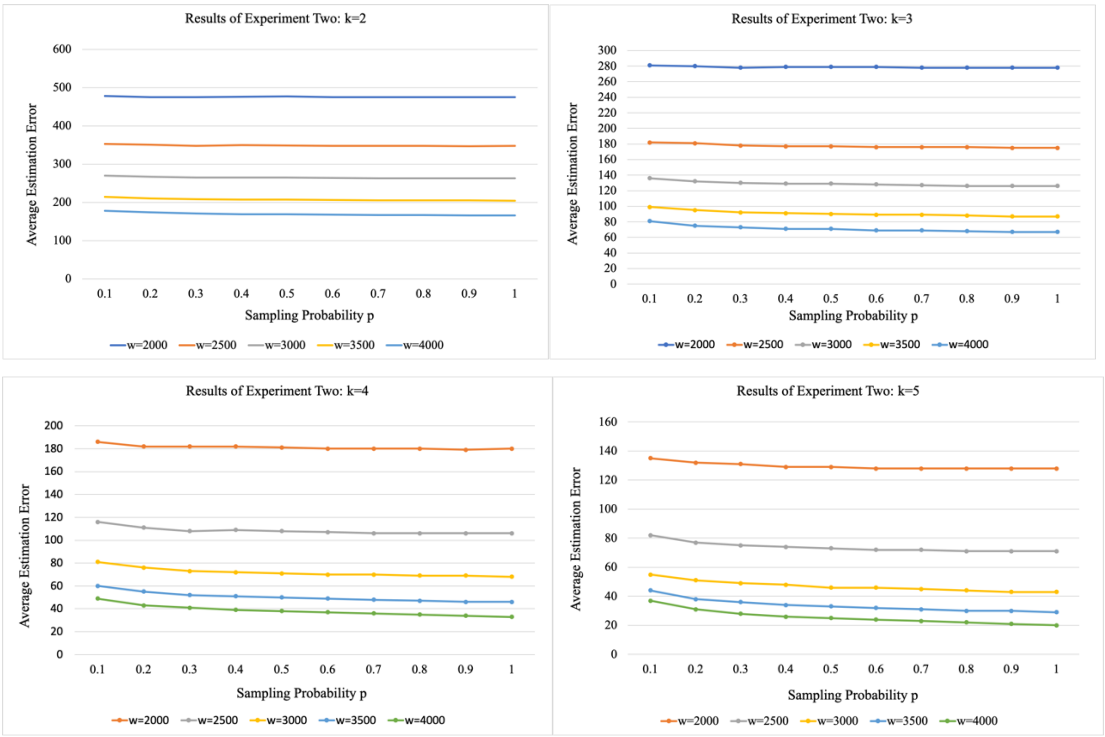

Figure 2:Results of experiment two

6 Related Work

After the proposal of CM sketch in 2005, a lot of researches related to CM sketch has been done. In 2007, Deng and Rafiei propose Count-mean-min (CMM) sketch in [6], which provides an unbiased estimator for CM sketch especially when the data is less skewed. The recording procedure of CMM sketch is similar to CM sketch, but the query process is different. CMM does not regard the minimum value of k counters as estimated size. Instead, it deducts the value of estimated noise from each of k counters and use the median value of the results as estimated

size. In 2011, Count-Min (CM) sketched with consecutive update (CU) is proposed by Goyal and Daumé III. The recording procedure of CM-CU differs from that of CM sketch. To record a flow f with flow size n_f, the minimum estimated flow size e_f based on the existing sketch need to be computed. Then record the larger value between $n_f + e_f$ and $C[i][H_i(f)]$ into each counter. Because existing solutions of estimating flow size only provide probabilistic error bound, the work of [5] in 2021 utilize false positive free zone of Bloom filter to provide guaranteed exact counting of a flow size in the Count-Min sketch. The deficient part of this work is that it does not prove there are more sketches where Bloom filter free zone can guarantee exact counting.

7 Conclusion

We modified Count-Min (CM) sketch by adding a sampling probability p to the recording procedure and hope this sample module can reduce the overestimation of CM sketch. However, after running some experiments with different sampling probability p and different sketch size, we conclude that sampling module does not behave well in reducing estimation error of CM sketch.

References

[1] G. Cormode and S. Muthukrishnan. 2005. An improved data stream summary: The Count-Min sketch and its applications. *J. Algorithms* (2005).

[2] G. Cormode, S. Muthukrishnan. 2004. What's new: finding significant differences in network data streams. *Proceedings of IEEE INFOCOM* (2004).

[3] C. Estan, G. Varghese and M. Fisk. 2006. Bitmap Algorithms for Counting Active Flows on High-Speed Links. *IEEE/ACM Transactions on Networking* (October 2006).

[4] N. Brownlee, C. Mills, and G. Ruth. 1999. RFC2722: Traffic Flow Measurement: Architecture. RFC Editor, USA.

[5] O. Rottenstreich, P. Reviriego, E. Porat, and S. Muthukrishnan. 2021. Avoiding Flow Size Overestimation in Count-Min Sketch with Bloom Filter Constructions. *IEEE Transactions on Network and Service Management* (2021).

[6] F. Deng, D. Rafiei. 2007. New estimation algorithms for streaming data: Count-min can do more[J]. Webdocs (2007).

[7] A. Goyal and H. Daume´ III. 2011. Approximate scalable bounded space sketch for large data NLP. *Empirical Methods in Natural Language Processing (EMNLP)* (2011).

[8] D. Emanuele, Favaro. Stefano and Peluchetti. Stefano. 2021. A Bayesian nonparametric approach to count-min sketch under power-law data streams. *AISTATS* (2021).

[9] D. Barman, P. Satapathy and G. Ciardo. 2007. Detecting Attacks in Routers Using Sketches. *2007 Workshop on High Performance Switching and Routing* (2007).

Appendix: Part of input data

23.96.219.115	777
49.77.20.20	9
185.35.62.200	205
88.100.184.82	20
115.227.123.18	547
200.158.26.238	5
185.35.62.56	192
78.169.15.116	15
173.245.59.76	175
151.250.238.173	9
95.5.171.118	554
52.17.133.62	42
218.61.30.235	358
112.205.116.232	51
201.35.228.66	12
129.16.138.161	200
203.192.248.229	590
170.238.78.34	103
92.45.195.186	13
123.52.207.175	109
114.222.42.174	129
188.166.26.84	970
49.206.175.126	622
103.210.47.203	123
63.148.206.76	941
70.55.161.199	20
71.197.181.79	2
141.212.122.136	123
90.148.27.28	10
201.190.103.236	106