# Cell Division: Better Parameter Settings for Sketches without Extra Costs

Paper #101, 6 pages

***Abstract*—Sketch is a probabilistic data structure used to record frequencies of items in a multiset. It can work with small memory usage and achieve high speed and accuracy. Therefore, sketches have been widely used in many applications, such as data stream processing, distributed datasets, natural language processing, and network traffic analysis. However, conventional sketches are not accurate enough for non-uniform datasets, which are common in practice. In this paper, we propose a novel technique, named Cell Division (CDV), which can improve the accuracy of sketches for non-uniform datasets without extra costs. The key idea of our technique is to use different parameters, including width and counter size, for different arrays in a sketch. We applied our novel technique to several well-known sketches, including CM sketches and CU sketches. We also made mathematical analysis to prove that our technique is able to improve the accuracy of sketches. Extensive experimental results show that our CDV has improved the accuracy of existing sketches by up to ... The source codes of our new technique are available on our homepage.**

## I. Introduction

### A. Background and Motivation

In today's computer science, estimating the frequency of items in a multiset is a very important problem. A multiset is a set in which items can appear more than once. This problem has been studied in various of fields, such as data stream processing [1]–[4], network traffic analysis [5], [6], natural language processing (NLP) [7], [8], data graph [9], distributed databases [10], social network [11], *etc.* As the speed of data streams becoming faster and faster, it becomes more difficult to achieve accurate estimation for item frequencies, and thus many approximate approaches using probabilistic data structures are being widely used and studied. Among all these approximate approaches, sketches [12] are considered the most successful one due to their high accuracy and fast speed.

A sketch is a probabilistic data structure that can be used to store the frequencies of items in a multiset. Compared to other methods for frequency query, sketches are more memory efficient and can achieve faster speed and higher accuracy. Therefore, sketches are widely used for the problem of estimating item frequencies. Typical sketches include CM sketches [13] and CU sketches [14]. A CM sketch consists of $d$ arrays, $A_1, A_2 \dots A_d$, and each array consists of $w$ counters. Furthermore, each array is associated a hash function $h_i(.)(1 \leqslant i \leqslant d)$. When inserting an item $e$, the CM sketch first computes these $d$ hash functions, $h_1(e), h_2(e) \dots h_d(e)$, and maps the item to $d$ counters, $A_1[h_1(e)], A_2[h_2(e)] \dots A_d[h_d(e)]$. Then it increments these $d$ mapped counters by 1. When querying an item $e$, the CM sketch just reports the minimum value from these $d$ mapped counters. The CU sketch has the same structure and query process as the CM sketch, but it only increments the mapped counters with minimum values instead of incrementing all of them when performing insertion processes. According to the analysis of real datasets, we found that most real datasets are non-uniform, which means that in most datasets, most items have a small frequency, while only a few items have a high frequency. For convenience, we call items with a small frequency `cold items`, and those with a high frequency `hot items`. However, all conventional sketches are unable to deal with this issue, because they are difficult to find a proper counter size to fit these non-uniform datasets. Therefore, there are many counters containing a small number (typically less than 10), and it is definitely a waste of memory. With a fixed memory size, the accuracy of sketches could be poor due to the big waste of memory.

To address this issue, there are many improved approaches, such as Augmented sketch (ASketch) [15], Counter Braids (CB) [16], *etc.* The ASketch adds a filter to a CM sketch, which contains frequencies for a certain number of `hot items`. Therefore, the ASketch achieves higher accuracy than conventional sketches in terms of `hot items`. However, the average accuracy of all items is not improved. The Counter Braids can dynamically allocate more counter size for `hot items`, and can achieve 100% accuracy in most cases. However, the Counter Braids does not support instant query, which means we can only query the frequencies of items after inserting all items. Note that prior approaches still have some shortcomings as mentioned above. To address this issue, we propose a novel technique, which can enable sketches perform better on these non-uniform datasets without extra costs like more memory accesses or more computations.

### B. Proposed Approach

In this paper, we propose a novel technique for sketches, named Cell Division (CDV), which enables sketches to perform better accuracy on non-uniform datasets without extra costs. *One of the key principles of our CDV technique is to use different parameters, including width and counter size, for different arrays in a sketch.* To be specific, we use geometric progression to generate counter sizes for arrays in the sketch. For example, we use a counter size of 2 bits for the first array, 4 bits for the second array, 8 bits for the third array, *etc.* A sketch usually has 4 arrays, and a counter size of 16 bits is enough to contain the maximum frequency among all items. Furthermore, we also use different width for each array, and typically we set a larger width for the array with smaller counter size. Note that in most real datasets, most items have a small frequency (about half of items have a frequency no larger than 2). The larger width for the array with smaller counter size has two following advantages: 1) it saves memory usage because the counter size is small. 2) it reduces the probability of hash collisions, and thus enhances the accuracy significantly.

*The second principle of our CDV technique is to treat counter overflows as flags that indicates the counter is mapped by items with larger frequencies than the capacity of the counter.* As stated above, we may use a counter size of only 2 bits for the first array, and thus counter overflows will happen frequently in this array. If we simply treat these overflowed counters as usual, then we will get illogical results. For example, if we apply our CDV technique to a CM sketch, when querying an arbitrary item, we can only get an estimated result smaller than 4, and then errors of `hot items` will be very huge. Therefore, we should treat counter overflows differently: we simply ignore these overflowed values, and do not take these values into account when performing query processes.

There are two important advantages of our CDV technique: 1) our CDV technique can significantly increase the accuracy for non-uniform datasets. 2) our CDV technique will introduce few extra costs, such as more computational cost, more memory accesses, *etc.*

### C. Key Contributions

1) First, we propose a novel technique for sketches, named Cell Division (CDV), which can significantly increase the accuracy of sketches on non-uniform datasets with few extra costs.
2) Second, we carry out comprehensive mathematical analysis and extensize experiments, and the results prove that our CDV has a great advantage over conventional sketches and the state-of-the-art in terms of accuracy.
3) Third, we place all related source codes of our CD-V technique at our homepage.

## II. Related Work

As data stream processing becomes more and more important these days, it becomes more difficult for conventional approaches to work at a high speed. Therefore, many approaches have been proposed in recent years, including sketches and counter variants.

The CM and CU sketch have been introduced in the previous section. To overcome the limitations of these conventional sketches, several improvements are proposed. For example, the Augmented Sketch (ASketch) [15] is proposed by Roy *et al.* An ASketch consists of a conventional CM sketch and a filter that contains frequencies for `hot items`. When inserting an item $e$, it first checks whether $e$ is in the filter. If so, it directly increments the corresponding counter in the filter by 1. Otherwise, it inserts it into the CM sketch. After each insertion, it will reorganize the filter to guarantee the items with highest frequencies are in the filter. However, the ASketch only achieves higher accuracy for a certain number of `hot items`, but does not enhance the overall accuracy of all items.

There are also several counter-based approaches aiming to solve the problem faced by conventional sketches. The most typical one is the Counter Braids [16]. The Counter Braids uses a layered structure. When inserting `cold items`, it only increments the counters in the first layer. While when inserting `hot items`, counters in the first layer may overflow and those in deeper layers will be incremented. The Counter Braids can achieve 100% accuracy in most cases, and is also memory efficient. However, it does not support instant query, which means that we can only query the item frequencies after inserting all items.

As mentioned above, all these approaches have obvious shortcomings, including low overall accuracy, not supporting certain functions, *etc.* Therefore, these approaches may have poor performance in their applications. The design goal of our proposed technique is to enhance the overall accuracy for sketches with few extra costs.

## III. The Cell Division Technique

In this section, we present the details of our Cell Division technique which can be applied to all exisiting sketches. We mainly present two key techniques of our Cell Division: *unbalanced parameter setting and overflow skip*. By applying the unbalanced parameter setting technique, we can increase the width of arrays with a small counter size, which can significantly reduce the probability of hash collisions. Therefore, the overall accuracy can be improved. By applying the overflow skip technique, we can prevent the illogical results for some items. Table I summarizes the symbols and abbreviations used in this paper.

TABLE I
Symbols & abbreviations used in the paper

| **Symbol** | **Description** |
|---|---|
| $d$ | # of arrays of a sketch |
| $w_i$ | # of counters in the $i^{th}$ array |
| $\overline{w}_i$ | # of bits in the $i^{th}$ array |
| $b_i$ | the counter size in the $i^{th}$ array |
| $e$ | an arbitrary element |
| $\mathrm{A}_i$ | the $i^{th}$ array in a sketch |
| $\mathrm{A}_i[j]$ | the counter in the $j^{th}$ position of the $i^{th}$ array |
| $h_i(.)$ | the $i^{th}$ hash function |
| CR | the correct rate of a sketch |
| RE | the relative error of an item |
| AE | the absolute error of an item |
| ARE | the average relative error of a sketch |
| AAE | the average absolute error of a sketch |
| Mips | mega-instructions per second |

### A. Technique I: Unbalanced Parameter Setting

*1) Rationale:* As stated above, in most real datasets, most items have a small frequency (typically less than 10), and only a few items have a large frequency. Conventional sketches use the same parameters for each array, but this parameter setting could wastes a lot of memory since most items have a small frequency, while the counter size should be large enough to contain the largest frequency among all items in the data stream. Therefore, with fixed memory size, the width of each array can be quite small, and thus more hash collisions will happen, leading to a low overall accuracy. So using different parameters for different arrays is a wise solution. On the one hand, we can set a smaller counter size for some arrays, and thus we can also set a larger width for these arrays. It can significantly reduce the probability of hash collisions for these arrays, and therefore improve the arruacy. On the other hand, we should also set a big counter size for some other arrays, since we should have counters whose size is big enough to contain large frequencies. Furthermore, the width of these arrays do not need to be large, since the percentage of items with large frequencies is small. Therefore, it can also guarantee the accuracy of `hot items`. Sketches using our unbalanced parameter setting technique may take on different structures. Here we mainly introduce two versions of our technique.

*2) Version 1:* Based on conventional sketches like CM sketches, we first set different counter sizes for different arrays in the first version.

**Technical content:** As the name of our technique indicates, the first version of our technique uses the idea of cell division. There are two main points to present this version: 1) it uses the same amount of memory (same number of bits) for each array, like conventional sketches; 2) it uses different counter size for each array. For example, there are 4 arrays in a sketch, and it uses a counter size of 2 bits, 4 bits, 8 bits and 16 bits for the 4 arrays. Therefore, if the width of the fourth array is $w$, then the first three arrays has a width of $8w$, $4w$, and $2w$, respectively. Assume that each counter is a cell, and each division will transform one single bigger cell into two smaller cells. Then counters in the first array have experienced cell division for three rounds, and those in the second and the third array have experienced for two rounds and one round, respectively. Furthermore, the total number of bits keeps constant after each round of division, which resembles the substance conservation phenomenon in cell divisions. The structure of this example is shown in Figure 1.

**Analysis:** As stated above, arrays with a small counter size reduces the probability of hash collisions, and thus increase the accuracy significantly. Arrays with a large counter size is able to contain the large item frequencies, and can prevent large errors for items with large frequencies. Therefore, the overall accuracy of the data stream can be enhanced greatly. However, this parameter setting for sketches may not be the best in many cases. Assume that there are only about 10% `cold items` whose frequency is less than 3, then most counters in the array whose counter size is 2 bits will overflow. It will waste a large amount of memory usage, and the performance is not good enough. In the next version, we will solve this issue by differentiate another parameter among different arrays.

*3) Version 2:* Based on the first version of our unbalanced parameter setting technique, we can make further improvements on other parameters. As mentioned above, the distribution of item frequencies will affect the performance of the first version of unbalanced parameter setting technique

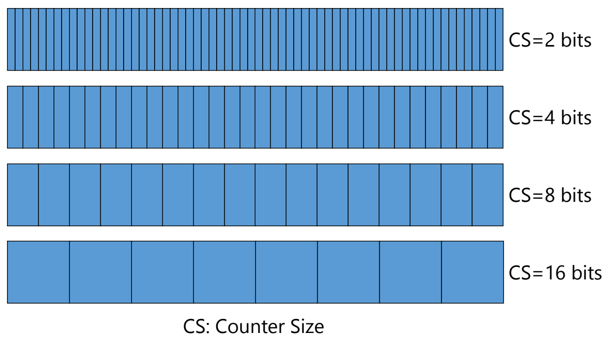

Fig. 1. The structure of a sketch using the first version of Cell Division.

greatly. Therefore, we should also take the parameter width into account.

**Technical content:** For different distributions of item frequencies, there are different suitable parameter settings. If the percentage of `cold items` decreases, the width of arrays with small counter size should decrease, and accordingly the width of arrays with larger counter size should increase. If that percentage increases, then the width of arrays with small counter size should increase, and that of arrays with larger counter size should decrease. Therefore, we should allocate different widths for different arrays based on the distribution. For example, as shown in Figure 2, there are 4 arrays in a sketch, and these 4 arrays have a counter size of 2 bits, 4 bits, 8 bits and 16 bits, respectively. When the degree of nonuniformity of the dataset is high, then the total number of bits in the first array increases while that in the last array becomes smaller. When the degree of nonuniformity is small, then the total number of bits in the last array increases. It resembles cell proliferation and cell apoptosis: when there are enough nutrition, cells will proliferate; and when lack of nutrition, some cells will die. *The key point of this technique is to use the nonuniformity of past data streams to determine the parameter settings.*

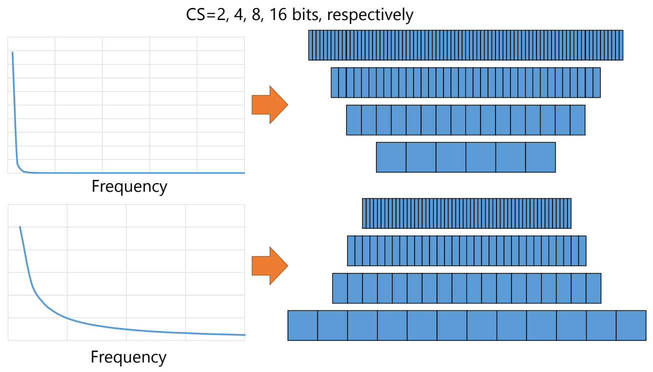

Fig. 2. The structure of a sketch using the second version of Cell Division.

**Analysis:** This version of unbalanced parameter setting can fit the distribution of data streams, so that it can almost achieve the best accuracy as long as there are no drastically changes in the distribution. However, there is still another problem unsolved. This version cannot handle counter overflows properly which can lead to low accuracy, especially for `hot items`. To address this problem, we propose our second technique in the next part.

### B. Technique II: Overflow Skip

In this part, we present our second technique, named *Overflow Skip*. This technique is aimed to handle counter overflows properly in order to prevent low accuracy for `hot items`. If we simply use the same query algorithms as conventional sketches like CM sketches, we will get illogical results. For example, if we are using a CM sketch with 4 arrays as mentioned above, and there is an item $e$ has a frequency of 10000. The item $e$ is mapped to 4 counters in the sketch, *i.e.*, $A_1[h_1(e)] \dots A_4[h_4(e)]$. When querying the item $e$, we should report the minimum value of these 4 counters. However, the counter $A_1[h_1(e)]$ has a size of only 2 bits, and thus the value in this counter is only 3 due to the counter overflow. Therefore, we will get an estimated frequency of 3. Obviously, it is illogical and leads to huge errors. To address this problem, we use Overflow Skip technique to handle overflows. *When querying an item, we will get $d$ mapped counters, where $d$ represents the number of arrays in a sketch. If a counter overflows, we will treat the counter as a flag instead of a meaningful frequency, and omit its value when reporting the minimum value among all these $d$ counters.* For example, as shown in Figure 3, we are using a CM sketch width 4 arrays as mentioned before. When querying an item $e$, it is mapped to 4 counters with values of 3, 15, 18 and 41. As the first counter and the second counter overflow, their values are omitted. Therefore, we report the estimated frequency of 18 instead of 3. The pseudocode of this technique applying to the CM sketch is presented in Algorithm 1.

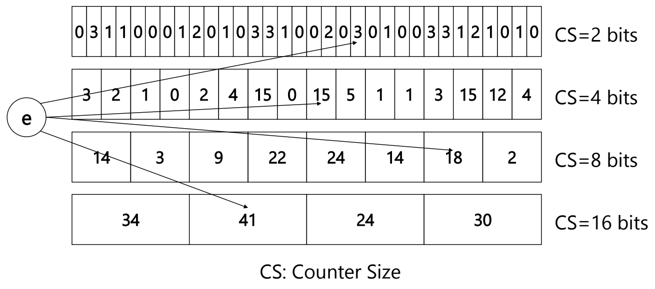

Fig. 3. The overflow skip technique.

### C. Cost Analysis

In this part, we give a brief analysis to prove that our CDV technique is free of extra costs.

**Memory access:** For a CM sketch without using our technique, the number of memory accesses for each insertion process or each query process is $d$, the number of arrays in the sketch. After using our CDV technique, it also requires $d$ memory accesses for each insertion and query process, since each counter can be accessed with one single memory access even if the counter size varies from different arrays. Therefore, there are no extra memory access for CM sketches after using our technique. This conclusion can also be applied to other sketches, whose analysis is omitted due to space limitation.

**Computational complexity:** Except for the amount of computations needed for sketches without using our technique, sketches using our technique requires at most 2 displacement operations for each mapped counter when performing insertion or query processes. This extra amount of computations is far smaller than that required before using our technique, including hash function computations. Therefore, our CDV technique almost introduce no extra computational complexity.

**Others:** Our CDV technique also does not introduce other extra costs. For example, the CM sketch and the CU sketch suffer from no under-estimation errors, and so do these sketches after using our technique. The proof of this will be presented in the next section. Furthermore, the CM sketch after using our technique also supports deletion operations like those without using our technique.

**Algorithm 1:** Query process using the overflow skip technique

```
Input: e: an arbitrary item; A: a CM sketch; b: the
       counter size of each array
Output: The estimated frequency of e
1 $min = MAX\_VALUE$;
2 for $j = 1, j \leqslant d, j++$ do
3     if $A_i[h_i(e)] < 2^{b_i} - 1$ then
4         if $min > A_i[h_i(e)]$ then
5             $min = A_i[h_i(e)]$;
6 return $min$;
```

## IV. Analysis

In this section, we carry out comprehensive mathematical analysis for our CDV technique. We mainly focus on the correct rate and the error bound of the CM sketch after using our CDV technique. We also give a proof of no under-estimation error for the CM sketch after using our technique. We call the CM sketch after using this technique the $\mathrm{C_{CM}}$ sketch.

### A. Proof of No Under-estimation Error

In this subsection, we prove that $\mathrm{C_{CM}}$ sketchhas no under-estimation error. Under-estimation error means that *the querying value is smaller than the real frequency.* As the insertion may make counters overflow and become invalid, the set of valid arrays for the inserted item change over time. For item $e$, let $S_n$ denote the set of indexes of valid arrays after $n^{th}$ insertion, and let $a_i$ denote the index of corresponding counter of item $e$ in $i^{th}$ array. By mathematical induction, we prove that if there is at least one valid array after $n^{th}$ insertion of item $e$, i.e. $S_n \neq \emptyset$, the $\text{C}_{\text{CM}}$ sketch has no under-estimation error.

**Base case:** Suppose $n = 1$. After first insertion of item $e$, all valid counters of $e$ increase by one, so $min_{i \in S_1}\{A_i[a_i]\} \geq 1$. The query result won't be smaller than 1, which means the $\text{C}_{\text{CM}}$ sketch has no under-estimation error in this case.

**Inductive hypothesis:** Assume when $n = k (k \geq 1)$, the $\text{C}_{\text{CM}}$ sketch has no under-estimation error, i.e. $min_{i \in S_k}\{A_i[a_i]\} \geq k$.

**Inductive step:** Suppose $n = k + 1$. As $S_{k+1} \subseteq S_k$, we know before the insertion, $min_{i \in S_{k+1}}\{A_i[a_i]\} \geq k$. After the insertion of item $e$, all valid counters of $e$ increase by one, so $min_{i \in S_{k+1}}\{A_i[a_i]\} \geq k + 1$. The query result won't be smaller than $k+1$, which means the $\text{C}_{\text{CM}}$ sketch has no under-estimation error in this case.

In conclusion, the $\text{C}_{\text{CM}}$ sketch has no under-estimation error.

## *B. Correct Rate of the $C_{CM}$ sketch*

Given a multiset with N distinct items, let $e_i$ denote the $i^{th}$ item, and $f_i$ denote its frequency. If the frequency exceeds the capacity of some counters, these counters will overflow and become invalid. Without loss of generality, we assume $b_1 \leq b_2 \leq \cdots \leq b_d$, and define

$$E_i = \begin{cases} \{e_j | 1 \leq f_j < 2^{b_1} - 1, 1 \leq j \leq N\} & i = 1 \\ \{e_j | 2^{b_{i-1}} - 1 \leq f_j < 2^{b_i} - 1, 1 \leq j \leq N\} & 1 < i \leq d \end{cases}$$

For items in $E_i$, their corresponding counters located in $1 \sim i - 1$ arrays will overflow and become invalid.

**Theorem 1.** *Let $p_i$ denote the probability that one arbitrary counter in $i^{th}$ array stores the accurate value of its corresponding item. $p_i$ equals to the probability that no collision happens in in one certain counter in $i^{th}$ array :*

$$p_i = (\frac{w_i - 1}{w_i})^{N-1} \tag{1}$$

According to the uniform property of hash function $h_i(.)$, this equation is obvious.

**Theorem 2.** *Let $P_i$ denote the probability that items in $E_i$ can get an accurate result when querying the $C_{CM}$ sketch. Then*

$$P_i = 1 - \prod_{j=i}^{d}(1 - p_j) \tag{2}$$

*Proof.* For item $e \in E_i$, if $1 \leq j < i$, $e$'s corresponding counters in $j^{th}$ array will overflow and the probability that it can't provide the accurate result is 1; if $i \leq j \leq d$, The probability that $e$ couldn't get the accurate result from its corresponding counter in $j^{th}$ array is $1-p_j$. The $\text{C}_{\text{CM}}$ sketch fails to report an accurate result when the corresponding counters in all arrays fail to provide the accurate result, whose probability is

$$P_{fail} = \prod_{j=1}^{i-1} 1 \prod_{j=i}^{d}(1 - p_j) = \prod_{j=i}^{d}(1 - p_j)$$

Therefore, the probability that $\text{C}_{\text{CM}}$ sketch can report the accurate result of $e$ is $1 - P_{failed}$. □

**Theorem 3.** *Let $C$ denote the correct rate of the $C_{CM}$ sketch. Then*

$$C = \sum_{i=1}^{d} \frac{|E_i| P_i}{N} \tag{3}$$

## *C. Error Bound of the $C_{CM}$ sketch*

Without loss of generality, assume $b_1 \leq b_2 \leq \cdots \leq b_d$.

**Theorem 4.** *For an arbituary item $e_i$, assume its corresponding counter in $t^{th}$ array doesn't overflow while ones in array $1 \sim t - 1$ overflow. Without loss of generality, assume all its corresponding counters in array $t \sim d$ don't overflow. Let $\hat{f}_i$ denote its estimated frequency and $f_i$ denote its real frequency. Let $N$ denote the number of distinct items and $V$ denote the sum of all items' real frequency, i.e. $V = \sum_{k=1}^{N} f_i$. Given a small variable $\epsilon$, we have the following guarantee with probability at least $1 - (\epsilon w_d)^{d-t+1}$:*

$$\hat{f}_i \leq f_i + \epsilon V \tag{4}$$

*Proof.* We define an indicator variable $I_{i,j,k}$ that is 1 if $h_j(e_i) = h_j(e_k)$, and 0 otherwise. Due to the independence of hash functions, the expectation of this indicator variable is:

$$E(I_{i,j,k}) = Pr[h_j(i) = h_j(k)] = \frac{1}{w_j} \leq \frac{1}{w_d} \tag{5}$$

We define the variable $X_{i,j}$ as $X_{i,j} = \sum_{k=1}^{N} I_{i,j,k} f_k$. $X_{i,j}$ reflects the expectation of the error caused by the collisions happening in a counter of $j^{th}$ array. Let $R(A_i[j])$ denote the report value of $A_i[j]$, then we have

$$R(A_i[h_j(e_k)]) = f_k + X_{i,j} \tag{6}$$

The expectation of $X_{i,j}$ is:

$$E(X_{i,j}) = E(\sum_{k=1}^{N} I_{i,j,k} f_k) = \sum_{k=1}^{d} f_k E(I_{i,j,k}) \leq \frac{V}{w_d} \tag{7}$$

Then, by the Markov inequality, we have:

$$\begin{aligned} & Pr\left[\hat{f}_i \geq f_i + \epsilon V\right] \\ & = Pr\left[\forall_{j \geq t}.\ R(A_j[h_j(e_i)]) \geq f_i + \epsilon V\right] \\ & = Pr\left[\forall_{j \geq t}.\ f_i + X_{i,j} \geq f_i + \epsilon V\right] \\ & = Pr\left[\forall_{j \geq t}.\ X_{i,j} \geq \epsilon V\right] \\ & = Pr\left[\forall_{j \geq t}.\ \frac{X_{i,j}}{E(X_{i,j})} \geq \epsilon w_d\right] \\ & \leq \left\{E\left[\frac{X_{i,j}}{E(X_{i,j})}\right] / (\epsilon w_d)\right\}^{d-t+1} \\ & = (\epsilon w_d)^{d-t+1} \end{aligned} \tag{8}$$

□

## V. Experimental Results

## VI. Conclusion

Sketches are widely used in many applications due to their high accuracy, memory efficiency and fast speed. However, many real datasets are non-uniform, and sketches are not accurate enough for these datasets because of high probability of hash collisions. To address this issue, in this paper, we propose a novel technique for sketches, named Cell Division (CDV), which enables exisiting sketches to achieve higher accuray with few extra costs. The key idea of our CDV technique is to use different parameters like width and counter size in different arrays in a sketch. We apply our technique to several typical sketches, including CM sketches and CU sketches. We also conduct comprehensive mathematical analysis and extensive experiments to prove that sketches perform better after using our technique. Experimental results show that our CDV technique is ... We believe our CDV technique can be widely applied to many sketches.

## References

[1] Moses Charikar, Kevin Chen, and Martin Farachcolton. Finding frequent items in data streams. *international colloquium on automata, languages and programming*, 312(1):693–703, 2002.

[2] Charu C Aggarwal and Philip S Yu. On classification of high-cardinality data streams. pages 802–813, 2010.

[3] Nishad Manerikar and Themis Palpanas. Frequent items in streaming data: An experimental evaluation of the state-of-the-art. *data and knowledge engineering*, 68(4):415–430, 2009.

[4] Dina Thomas, Rajesh Bordawekar, Charu C Aggarwal, and Philip S Yu. On efficient query processing of stream counts on the cell processor. pages 748–759, 2009.

[5] Zaoxing Liu, Antonis Manousis, Gregory Vorsanger, Vyas Sekar, and Vladimir Braverman. One sketch to rule them all: Rethinking network flow monitoring with univmon. pages 101–114, 2016.

[6] Aiyou Chen, Yu Jin, Jin Cao, and Li Erran Li. Tracking long duration flows in network traffic. pages 1–5, 2010.

[7] Amit Goyal, Hal Daume, and Graham Cormode. Sketch algorithms for estimating point queries in nlp. pages 1093–1103, 2012.

[8] Amit Goyal, Hal Daume, and Suresh Venkatasubramanian. Streaming for large scale nlp: language modeling. pages 512–520, 2009.

[9] Peixiang Zhao, Charu C Aggarwal, and Min Wang. gsketch: on query estimation in graph streams. *Proceedings of The Vldb Endowment*, 5(3):193–204, 2011.

[10] Graham Cormode and Minos Garofalakis. Sketching streams through the net: distributed approximate query tracking. *very large data bases*, pages 13–24, 2005.

[11] Charu C Aggarwal and Karthik Subbian. Event detection in social streams. pages 624–635, 2012.

[12] Graham Cormode. Sketch techniques for approximate query processing. 2011.

[13] 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.

[14] Cristian Estan and George Varghese. New directions in traffic measurement and accounting. *acm special interest group on data communication*, 32(4):323–336, 2001.

[15] Pratanu Roy, Arijit Khan, and Gustavo Alonso. Augmented sketch: Faster and more accurate stream processing. pages 1449–1463, 2016.

[16] Yi Lu, Andrea Montanari, Balaji Prabhakar, Sarang Dharmapurikar, and Abdul Kabbani. Counter braids: a novel counter architecture for per-flow measurement. *measurement and modeling of computer systems*, 36(1):121–132, 2008.