Discussion on Fast and Accurate Sketches for Skewed Data Streams: A Case Study

Shuhao Sun[1] and Dagang Li[1,2](✉)

[1] School of ECE, Peking University Shenzhen Graduate School, Shenzhen 518055, China
shuhaosun@pku.edu.cn, dgli@pkusz.edu.cn
[2] Institute of Big Data Technologies, Peking University, Shenzhen 518055, China

Abstract. Sketch is a probabilistic data structure designed for the estimation of item frequencies in a multiset, which is extensively used in data stream processing. The key metrics of sketches for data streams are accuracy, speed, and memory usage. There are various sketches in the literature, but most of them cannot achieve high accuracy, high speed and using limited memory at the same time for skewed datasets. Recently, two new sketches, the Pyramid sketch [1] and the OM sketch [2], have been proposed to tackle the problem. In this paper, we look closely at five different but important aspects of these two solutions and discuss the details on conditions and limits of their methods. Three of them, memory utilization, isolation and neutralization are related to accuracy; the other two: memory access and hash calculation are related to speed. We found that the new techniques proposed: automatic enlargement and hierarchy for accuracy, word acceleration and hash bit technique for speed play the central role in the improvement, but they also have limitations and side-effects. Other properties of working sketches such as deletion and generality are also discussed. Our discussions are supported by extensive experimental results, and we believe they can help in future development for better sketches.

Keywords: Sketch · Skewed data · Data structure

1 Introduction

Estimating the frequency of each item in a multiset is one of the most classic tasks in data stream applications. In many networking scenarios such as real-time IP traffic, IP phone calls, videos, sensor measurements, web clicks and crawls, massive amount of data are often generated as high-speed streams [3, 4], requiring servers to process such stream in a single-pass [5]. Calculating exact statistics (e.g., using hash tables) is often impractical, because the time and space overhead of storing the whole data stream is too high. Therefore, it is popular and widely accepted to estimate the frequencies of each item by the probabilistic data structure [6–8].

Sketches are a family of probabilistic data structure designed for the estimation of item frequencies in data streams [9, 10], which is extensively used in data stream processing. They use counters to store frequencies and have two primary operations:

insertion and query. By using multiple hash functions, sketches summarize massive data streams within a limited space, which means there might be two or more items sharing the same counter(s). Sketches can also be applied to other fields, such as compressed sensing [11], natural language processing [12], and data graph [13].

Conventional sketches (CM sketch [7], CU sketch [14], Count sketch [8], and Augmented sketch [6]) use a number of counters of fixed size. The size needs to be large enough to accommodate the highest frequency. However, according to the literatures [6] and confirmed by our experiments on real datasets, the items in real data streams often have unbalanced distribution, such as Zipf [15] or Power-law [16]. This means that most items have low frequency (called cold items), while a few items have high frequency (called hot items). Such data streams are often called skewed data streams. Therefore, the high-order bits in most counters of conventional sketches are wasted, as hot items are much fewer than cold items in real data streams. This kind of memory inefficiency reduces the number of counters, causing the accuracy of the conventional sketches to drop drastically. Besides, conventional sketches cannot perfectly catch up with the high speed of data streams because they need three or more hash computations and memory accesses for each insertion or query. Overall, conventional sketches fall short handling skewed data streams, and the goal of this paper is to discuss how to design better sketches for this matter.

Two novel sketches have been proposed recently, the Pyramid sketch [1] and the OM sketch [2], which can achieve both high accuracy and high speed using limited memory, especially for skewed data streams. These two sketches bring new ideas that are specifically designed for skewed data. For example, automatic enlargement and hierarchy can greatly improve the accuracy when summarizing skewed datasets, and word acceleration and hash bit technique can significantly improve the speed for each insertion or query operation. However, we found that many aspects need to be further considered when using these techniques, therefore in this paper we will discuss the strategies of automatic enlargement, the side-effect of hierarchy, the use conditions of word acceleration and hash bit technique. Furthermore, we found that there are two other aspects to improve accuracy, which are barely scratched in the original papers [1, 2]. We name these two methods as isolation and neutralization. The usage of them depends on the specific target application scenario. Moreover, when designing the sketch, other requirements and constraints brought by the target application scenario should also be considered, such as deletion and generality [17]. These are also discussed in this paper.

Our contributions can be summarized as follows.

- We sort out five important aspects to design an accurate and fast sketch for skewed data streams. Three of them, memory utilization, isolation, and neutralization are to help improve accuracy, and the other two: memory access and hash computation are important for speed. Their role in an effective and efficient solution are analyzed.
- The specific methods proposed from the latest work [1, 2] are discussed in details, including the strategies of automatic enlargement, the side-effect of hierarchy, the usages of isolation and neutralization, the use conditions of word acceleration and hash bit technique. We also discuss the deletion and generality of the sketch. These discussions will help better understanding and further utilization of these new ideas.

2 Related Work

2.1 Conventional Sketches

Typical sketches include CM sketch [7], CU sketch [14], Count sketch [8], and Augmented sketch [6]. A CM sketch consists of d arrays: $A_1 \ldots A_d$, and each array consists of w counters. There are d hash functions, $h_1 \ldots h_d$, in the CM sketch. When inserting an item e, the CM sketch first computes the d hash functions and locates the d counters: $A_1[h_1(e)] \ldots A_d[h_d(e)]$. Then it increases all the d hashed counters. When querying an item e, the CM sketch reports the minimum of the d hashed counters as the estimated frequency of this item. The CU sketch has a slight but effective modification to the CM sketch, that is, conservative update. It only increases the smallest one(s) among the d hashed counters during insertions while the query process keeps unchanged. The Count sketch is similar to the CM sketch except that each array uses an additional hash function to smooth the accidental errors. The Augmented sketch aims to improve the accuracy by using one additional filter to dynamically capture hot items, suffering from complexities, slow insertion and query speed. Among these sketches, the CU sketch achieves the best performance in terms of both accuracy and speed. More sketches are detailed in the survey [18].

Unfortunately, the sketches above have two shortcomings for skewed data streams: (1) the accuracy is poor when using limited memory; (2) requiring multiple memory accesses and hash computations for each insertion or query thus slow the speed.

2.2 The OM Sketch

The key techniques of OM sketch are hierarchical counter-sharing, word acceleration and fingerprint check.

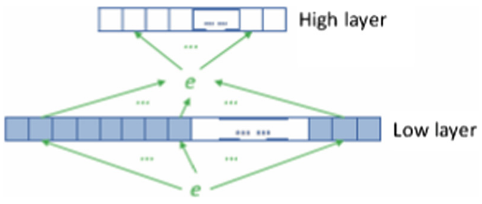

Fig. 1. Basic structure of OM sketch.

As shown in Fig. 1, the OM sketch is organized as a two-layer structure in which the high layer possesses less memory. The low layer with small counter sizes mainly records the information of cold items, while the high layer with relatively large counter sizes mainly records the information of hot items. When one or more counters overflow at the low layer, the OM sketch uses the high layer to record its number of overflows. Based on this structure, the OM sketch significantly improves the memory efficiency, thus improving accuracy. Moreover, the OM sketch constrains the hashed counters within one or several machine words by using the word acceleration technique. It also leverages the hash bit technique [19] to locate multiple hashed counters within one or several machine words at each layer through a 64-bit hash value by one hash function. Therefore, the OM sketch achieves close to one memory access and one hash computation for each insertion or query. Besides, the OM sketch records the fingerprints of the overflowed items in their corresponding machine words at the low layer in order to distinguish them from non-overflowed items during queries.

Insertion: When inserting an item, the OM sketch first computes the low layer hash function to locate the low layer hashed counters, and then increases the smallest counter(s). This method makes the low layer counters of each item always overflow concurrently. If an item overflows, the OM sketch first sets all its low layer hashed counters to zero, and then uses the fingerprint technique to distinguish it from non-overflowed items. Finally, the OM sketch computes the high layer hash function to locate the high layer hashed counters and increases the smallest counter(s).

Query: When querying an item, the OM sketch first gets the value of the smallest hashed counter(s) at the low layer, denoted by V_l. Then it checks if the item overflows. If it is, the OM sketch queries the high layer and gets the value of the smallest hashed counter(s) at the high layer, denoted by V_h. The OM sketch returns $V_l + V_h \times 2^{\delta_l}$ as the estimated size of the item, and δ_l is the counter size at the low layer.

2.3 The Pyramid Sketch

The key techniques of the Pyramid sketch are counter-pair sharing, word acceleration and Ostrich policy.

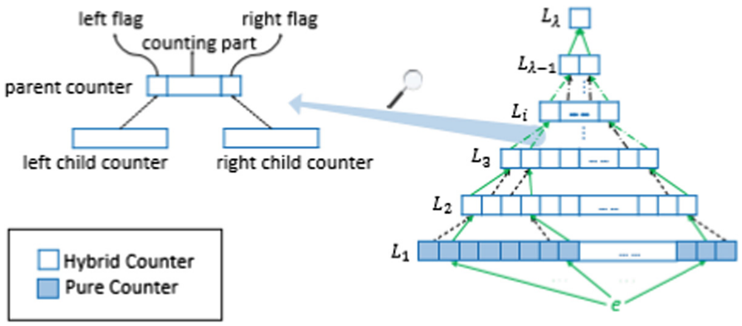

Fig. 2. Basic structure of Pyramid sketch.

As shown in Fig. 2, the Pyramid sketch employs a pyramid-shaped data structure. The i^{th} layer L_i is associated with the $i+1^{th}$ layer L_{i+1} in the following way: the left child counter and the right child counter at L_i are associated with the parent counter at L_{i+1}. When the child counter overflows, the Pyramid sketch uses its parent counter to record its number of overflows. In Pyramid sketch, the first layer is composed of pure counters, only used for recording frequencies. The other layers are composed of hybrid counters, which can be split into three parts: the left flag, the counting part and the right flag. The flag parts indicate whether its child counters are overflowed. Based on this counter-pair sharing technique, the Pyramid sketch dynamically assigns the appropriate number of bits for different items with different frequencies, thus improving the memory efficiency. Like OM sketch, the Pyramid sketch uses word acceleration and hash bit technique to improve its speed.

The Pyramid sketch can be applied to conventional sketches (CM, A, C and CU), and the results are denoted as Pcm, Pa, Pc and Pcu. It uses a novel strategy, Ostrich policy, to improve the insertion speed of sketches that need to know the values of the d mapped counters during each insertion. Here, we take Pcu as an example. The key idea of Ostrich policy is ignoring the second and higher layers when getting the values of the d mapped counters, only increases the smallest first layer counter(s).

Insertion: When inserting an item, the Pyramid sketch first computes the hash function to locate the hashed counters at layer L_1. Different sketches will perform different increase operations on these counters. If any of the counters overflows, the Pyramid sketch sets the counter to zero, and assigns its parent counter according to its index. Then, the left/right flag of its parent counter will be set to 1. These operations are called carryin. The Pyramid sketch repeats the carryin operation at layer L_2, and the operation will be performed layer by layer until there is no overflow.

Query: When querying an item, the Pyramid sketch first locates the hashed counters at the first layer, and then gets the values of the d mapped counters by accumulating the values of corresponding counters of each layer. Finally, the Pyramid sketch produces the query output based on the specific sketch under use.

3 Analysis and Discussion

In this section, we will discuss from five different aspects on how to design an accurate and fast sketch for skewed data streams. We use the OM sketch and the Pyramid sketch as latest examples, discussing their methods handling these important aspects. At the end of this section, we will discuss two more aspects, namely the support for deletion and generality. Depending on the target scenarios they might also become as important as the former ones.

3.1 Accuracy Improvement of Sketch for Skewed Datasets

Accuracy is one of the most important indicators of the sketch. We can try tackle the problem from three different aspects: (1) higher memory utilization, (2) isolation, and (3) neutralization to improve the accuracy of the sketch. In the following we will discuss the solutions from the literatures and our findings.

Improvement of Memory Utilization
Improvement of memory utilization means increasing the number of counters in the same memory, so as to reduce the probability of collision. Automatic enlargement and hierarchy are techniques that can be used to improve memory utilization.

Automatic Enlargement Technique
In the process of the automatic enlargement, it's unnecessary to allocate enough bits to each counter in advance. When the counter overflows, the sketch enlarges the initial counter space automatically. In view of the characteristics of skewed data streams, this technique can greatly improve memory utilization of the sketch.

When using the automatic enlargement technique, the enlargement strategy depends on the type of counter overflows. There are two types of overflows, one is called simultaneous overflow and the other is called non-simultaneous overflow. Simultaneous overflow means that all counters of an item overflow at the same time, and needs only one enlargement, while non-simultaneous overflow means that not all counters of an item overflow at the same time, so multiple enlargements are often necessary. What's more, if the initial counter space is separated from its enlarged space, during the automatic enlargement, the non-simultaneous overflow needs to establish the corresponding relationship between each initial counter space and its enlarged space, otherwise the sketch cannot be queried. However, it is unnecessary to do so for the simultaneous overflow. When querying an item, the sketch only needs to query the two spaces separately. Compared to the non-simultaneous overflow, the simultaneous overflow has fewer enlargements and relatively simpler enlargement strategy. However, realizing simultaneous overflow needs to design specific insert operations, which may require extra cost, such as some trade-offs of performances.

Both the OM sketch and the Pyramid sketch adopt automatic enlargement technique. For the OM sketch, it only increases the smallest counter(s) during insertion, thus achieving simultaneous overflow. For the Pyramid sketch, the insert operations depend on the sketch under use. Counters of an item cannot be guaranteed to overflow at the same time. Therefore, the Pyramid sketch is a non-simultaneous overflow. Besides, the initial counter space of the OM sketch and Pyramid sketch is separated from its enlarged space. The automatic enlargement strategy of OM sketch is to use another hash function to enlarge the space for all the overflowed counters. When querying an item, the OM sketch locates the counters of each space by two hash functions. The automatic enlargement strategy used in the Pyramid sketch is index calculation, which is to establish the relationship between the initial counter space and its enlarged space. When querying an item, the parent counters are located through the indices of their child counters. Specific insert operation lets the OM sketch achieve simultaneous overflow, simplifying the automatic enlargement strategy. Meanwhile, it also makes the OM sketch isolate items, which further improve the accuracy (will be described later). However, the cost is that the OM sketch cannot support deletion.

Hierarchical Structure
Considering the characteristics of skewed datasets, the higher frequency of an item is, the less proportion it occupies. Thus, we can design the sketch as a hierarchical structure. For example, lower layers have smaller size but a larger number of counters, while higher layers have larger size but a smaller number of counters. The hierarchical structure increases the number of counters, thus improving the memory utilization. The more the number of layers, the higher the accuracy of the sketch. However, hierarchy has a side-effect, which cannot be ignored. With the increase of the number of layers, the number of memory accesses will be increased, which can slow insertion and query

speed. Therefore, when using hierarchy technique, the number of layers of the sketch is selected based on a tradeoff between accuracy and speed.

Both the Pyramid sketch and the OM sketch adopt the hierarchy technique. The Pyramid sketch is a multi-layer pyramid-shaped structure, while the OM sketch is a two-layer trapezoid structure. As mentioned earlier, there is a corresponding relationship between the initial counter space and its enlarged space in the Pyramid sketch. If one child counter monopolizes to one parent counter, massive memory waste will be caused because of the characteristics of skewed datasets. Therefore, the Pyramid sketch lets two child counters share one parent counter in order to improve the memory utilization. With this corresponding relationship, the Pyramid sketch gradually forms a pyramid type. Using a multi-layer structure rather than a two-layer structure like the OM sketch is to further increase the memory utilization. However, this hierarchical structure will slow the insertion and query speed of the Pyramid sketch.

To design the OM sketch as a two-layer structure is a tradeoff between accuracy and speed. For the OM sketch, the characteristics of the low layer counters conform to the characteristics of low frequency items, whose sizes are small and numbers are large, while the characteristics of the high layer counters conform to the characteristics of the intermediate and high frequency items, whose sizes are large and numbers are small. In skewed data streams, the vast majority of items are the low frequency items. Therefore, dividing into two layers can significantly improve the memory utilization of the OM sketch. The more the layers, the lower the accuracy. Thus, the OM sketch is designed as a two-layer structure.

Isolation
Improvement of memory utilization is to improve accuracy by reducing hash collisions between items. In addition to reducing hash collisions, we can limit the range of hash collisions to reduce the collisions between items of different frequency segments. For example, we can isolate the low, intermediate and high frequency items in the sketch, so that the collisions occur only within these frequency segments but not cross. This method reduces the impact of high frequency items on intermediate and low frequency items, and the impact of intermediate frequency items on low frequency items, thus improving the accuracy. Besides, we can design the sketch according to specific requirements and application scenarios. For example, in some scenarios (e.g., NLP), the accuracy of low frequency items is very important. Thus, we can design corresponding sketches to improve the accuracy of low frequency items, and the accuracy of intermediate and high frequency items can be relaxed appropriately.

As mentioned earlier, the simultaneous overflow makes the OM sketch isolate items. This is because it is unnecessary for the simultaneous overflow to establish the relationship between the initial counter space and its enlarged space. The low layer of the OM sketch plays a role of filtering all low frequency items, so only the intermediate and high frequency items can get into the high layer. Therefore, the impact of intermediate and high frequency items on low frequency items is reduced. The accuracy of low frequency items can be greatly improved. The Pyramid sketch does not isolate items. Its child counters are bound to their corresponding parent counter. The low frequency items that collide with the intermediate and high frequency items at the lower layer will also get into the higher layer. Therefore, the estimated frequencies of low frequency items are still affected by the intermediate and high frequency items.

Neutralization

The evaluation indices of accuracy can be divided into under-estimation rate, correction rate and over-estimation rate. There are different requirements for them in different application scenarios. We can improve the accuracy of the sketch by using specific application features. For example, if the application scenario allows the sketch have under-estimation error, a small amount of under-estimation error can be introduced to neutralize part of the over-estimation-error and improve the correction rate. Since the estimated frequency is already larger than the real frequency in many cases, a little under-estimation can improve the overall accuracy. However, if the target application scenario does not allow under-estimation-error, this method will not work. All sketches with under-estimation error will not be applicable to such scenarios. Therefore, this neutralization method is related to the tolerance of under-estimation-error, depending on specific application scenario.

The OM sketch has under-estimation error, which can improve a bit of accuracy. The cause of under-estimation is that when counters overflowed, the OM sketch set all these counters to zero. For the Pyramid sketch, if conventional sketch has under-estimation-error or over-estimation-error, then its corresponding Pyramid sketch will also have it, otherwise it will not. However, there is a special case, Pcu sketch. Since the Pcu sketch uses Ostrich strategy, each insertion does not necessarily increase the smallest hashed counter(s), resulting in a little under-estimation. However, this under-estimation neutralizes part of over-estimation. Thus, in the original paper [2], the experimental results show that Ostrich policy can help improve accuracy. Neither the OM sketch nor the Pcu sketch can be applied to the scenarios that do not allow under-estimation error.

3.2 Insertion and Query Speed Improvement of Sketch

Another important indicator of the sketch is speed. The speed is mainly related to the number of memory accesses and the number of hash calculations required for each insertion and query. Therefore, there are two ways to improve the speed, and in the following discussions we will reference two methods: (1) word acceleration and (2) hash bit technique as representing examples.

Reduction in the Number of Memory Accesses

For conventional sketches, the number of memory accesses for each insertion or query is the same as the number of counters assigned for each item, usually more than three. Since the counter size of conventional sketches is usually large (e.g., 16 bit), it is difficult to reduce the number of memory accesses. However, if we use certain techniques to make the counter size smaller, we can constrain the counters of one item within one or several machine word to reduce the number of memory accesses for each insertion or query. This is called word acceleration, which use condition is that the counter size should be relatively smaller. In modern CPUs, a machine word is usually 64 bits in width. In the GPU architecture, the size of a machine word is much larger. Therefore, one machine word on CPU or GPU can typically contain a reasonably large number of small counters.

The hierarchical structure of the Pyramid sketch and the OM sketch increases the number of memory accesses. However, this structure makes their counters size smaller, so that both of them can use word acceleration.

The OM sketch constrains the hashed counters of the low layer within one machine word and the hashed counters of the high layer within two machine words, and scatters these counters over these machine words evenly. As real data streams are skewed, the probability of accessing the high layer for each insertion and query is very small (e.g., 1/20). Therefore, the average number of memory accesses for each insertion and query is close to 1 (e.g., $1 + 1/20 * 2 = 1.1$). The Pyramid sketch constrains the hashed counters of each layer within one machine word. Most of the insertions only access the first layer (the Ostrich strategy also makes Pcu so). Therefore, the average number of memory accesses for each insertion is close to 1, which has been proved in the original paper [1]. However, for queries, the number of memory accesses depends on the frequency of queried item. The higher frequency of queried item, larger number of layers to be accessed and larger number of memory accesses. Therefore, although the Pyramid sketch adopts word acceleration technique, it does not make great improvement in reducing the average number of memory accesses for queries.

Reduction in the Number of Hash Computations
For conventional sketches, the number of hash calculations for each insertion or query is also the same as the number of counters assigned for each item. This is because the size of the counter address is usually large. Thus, one counter can only be positioned by one hash function. However, if we use certain techniques to make the address size smaller, we can use fewer hash functions to locate the counters. For example, if we have adopted the word acceleration to constrain the hashed counters within one or several machine words, the address size of these counters can be shortened. We can leverage the hash bit technique from the literature [19] to reduce the number of hash computations. The key idea is that split one hash value into several bit arrays to locate one or several machine words and offsets of counters in the corresponding machine words. In this way, we can use only one hash computation to handle a sketch which originally required multiple hash computations.

Both OM sketch and Pyramid sketch use hash bit technique. The OM sketch uses hash bit technique at each layer. Supposing the probability of accessing the high layer is 1/20, the average number of hash computations for each insertion or query is close to 1 (e.g., $1 + 1/20 * 1 = 1.05$). The Pyramid sketch only uses hash function and hash bit technique at the first layer. Counters of other layers are located by the index of the first layer counters. Therefore, for the Pyramid sketch, the average number of hash computations for each insertion or query is 1, achieving one hash computation.

3.3 Other Related Aspects

In addition to accuracy and speed, there are also other important properties that need to be considered in the designing of sketch. Here we will discuss a bit on two of them: the support for deletion and the support for generality, which are actually considered in [1, 2].

Deletion: In some application scenarios, the sketch is required to support deletion [18]. If the insert operation is always reversible throughout the use of the sketch, the sketch can support deletion, and the delete operation is the inverse operation of insertion. For example, the insert operation of the CM sketch is to plus 1, then its delete operation is to subtract 1. Therefore, to make the sketch support deletion, we should design a reversible insert operation. The insert operation of the OM sketch is the same as that of the CU sketch, and neither of them supports deletion.

Generality: If the goal of a sketch is to solve a common problem for all sketches, it can be applied to all sketches to improve the target performance of them. We say such a sketch has generality. If the design goal of a sketch is to enhance one or more performances and cannot be applied to all sketches, such sketch has no generality. However, sketches that have no generality are more targeted, thus may be more significant to improve the target performances. The OM sketch sacrifices generality, but brings more significant accuracy and speed.

4 Experimental Result

4.1 Metrics

Average Absolute Error (AAE): AAE is defined as $\frac{1}{|N|}\sum_{i=1}^{N}\left|f_i - \hat{f_i}\right|$ where f_i is the real frequency of the i^{th} item, $\hat{f_i}$ is the estimated frequency of this item, and N is the total number of distinct items in the query set.

Average Relative Error (ARE): ARE is defined as $\frac{1}{|N|}\sum_{i=1}^{N}\left|f_i - \hat{f_i}\right|/f_i$.

Under-Estimation Rate (UER): UER is defined as N_{under}/N where N_{under} is the number of distinct items whose estimated frequency is less than its real frequency.

Correct Rate (CR): CR is defined as N_{acc}/N where N_{acc} is the number of distinct items whose estimated frequency equals to its real frequency.

Over-Estimation Rate (OER): OER is defined as N_{over}/N where N_{over} is the number of distinct items whose estimated frequency is larger than its real frequency.

Throughput: We simulate how sketches actually insert and query on CPU platform and calculate the throughput using mega-instructions per second (Mips).

4.2 Experimental Setup

We use the real IP trace from the main gateway at our campus. The estimation of item frequency corresponds to the estimation of the number of packets in a flow. The number of packets of the trace is 10M and the number of distinct flows is around 1M.

We implement the sketches of CM, CU, C, A, OM sketch and Pyramid sketch in C++. For the four conventional sketches, we set the counter size to 16 bits and the number of arrays to 4. Other experimental settings are the same as the original paper [1, 2]. In all our experiments, unless noted otherwise, the memory size of each sketch is 1 MB by default. We performed all the experiments on a machine with 2-core CPUs (2 threads, Pentium(R) Dual-Core CPU E5800 @3.2 GHz) and 4 GB total DRAM memory.

4.3 Performance of Different Sketches

The experiment results of Figs. 3 and 4 show that the AAE and ARE of the OM sketch and the Pyramid sketch are much smaller than those of the conventional sketches. The experiment results of Figs. 5 and 6 show that the insertion and query throughput of the OM sketch and the Pyramid sketch are much higher than those of the conventional sketches. We can see that by using automatic enlargement and hierarchy to improve accuracy and by using word acceleration and hash bit technique to improve speed, the OM sketch and the Pyramid sketch achieve a much better performance than the state-of-the-art in terms of both speed and accuracy.

From the experiment results, we find that in Pyramid versions, Pcu sketch achieves the highest accuracy and speed. Furthermore, the accuracy and speed of Pcu sketch is the closest to that of the OM sketch. Besides, the increase operation of the OM sketch is the same as the Pcu sketch and CU sketch. Therefore, in the following experiments, we use the Pcu sketch as an example of Pyramid sketch, and compare the performances of the CU sketch, Pcu sketch and OM sketch.

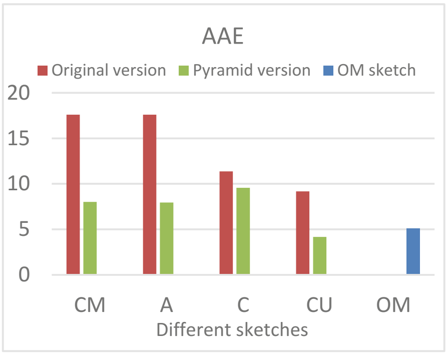

Fig. 3. AAE of sketches

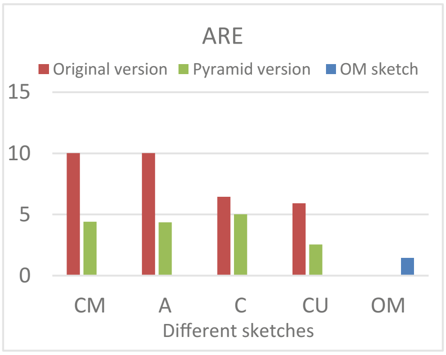

Fig. 4. ARE of sketches

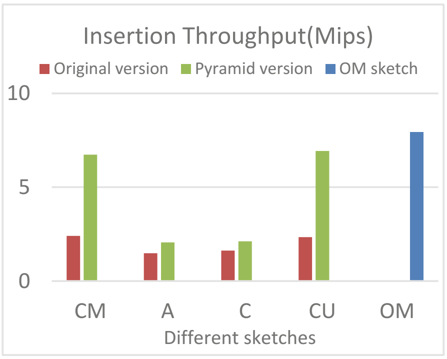

Fig. 5. Insertion throughput of sketches

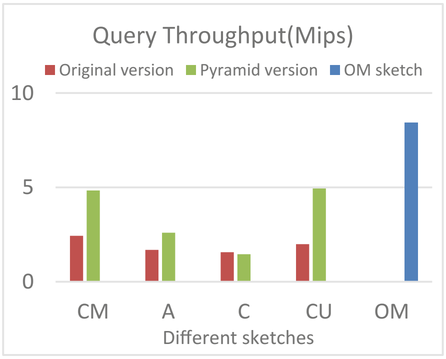

Fig. 6. Query throughput of sketches

4.4 Performances of CU, Pcu and OM Sketch

AAE and ARE

Figures 7 and 8 plots the AAE and ARE of three different sketches on different memory sizes increasing from 0.40 MB to 2.00 MB with a step of 0.20 MB. Our experimental results show that the AAE and ARE of the Pcu sketch and the OM sketch are always lower than those of the CU sketch.

Besides, we find that the ARE of the OM sketch is always lower than that of the Pcu sketch. As mentioned in the Sect. 3.1, the OM sketch uses the isolation method of filtering all low frequency items at the low layer, improving the accuracy of low frequency items. The accuracy of low frequency items has the greatest impact on ARE. Thus, the ARE of the OM sketch can be improved. Meanwhile, the Pyramid sketch does not use the isolation method, and the estimated frequencies of low frequency items are still affected by the intermediate and high frequency items. Therefore, its ARE is always lower than that of the OM sketch. Our experimental results have proved that isolation can help improve the accuracy of the sketch.

Under-Estimation Rate, Correct Rate and Over-Estimation Rate

Figures 9, 10 and 11 plots the under-estimation rate, correct rate and over-estimation rate of three different sketches on different memory sizes increasing from 0.10 MB to 2.00 MB with a step of 0.40 MB. Our experimental results show that expect the CU sketch, both the Pcu sketch and the OM sketch have under-estimation. The under-estimation rate of the OM sketch is about 4.33 times higher than that of the Pcu sketch. The correct rate of the OM sketch is about 1.26 and 8.83 times higher than those of the Pcu and CU sketch. And the over-estimation rate of the OM sketch is about 1.27 and 1.64 times lower than those of the Pcu and CU sketch.

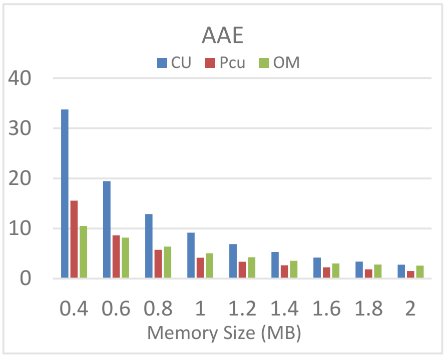

Fig. 7. AAE vs. memory sizes

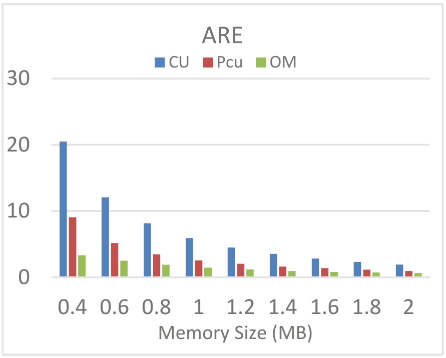

Fig. 8. ARE vs. memory sizes

The experimental results show that the higher under-estimation rate, the lower over-estimation rate and the higher correct rate. As mentioned in Sect. 3.1, a small amount of under-estimation-error can be introduced to neutralize part of the

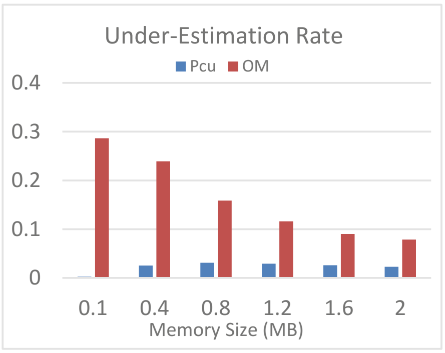

Fig. 9. UER vs. memory sizes

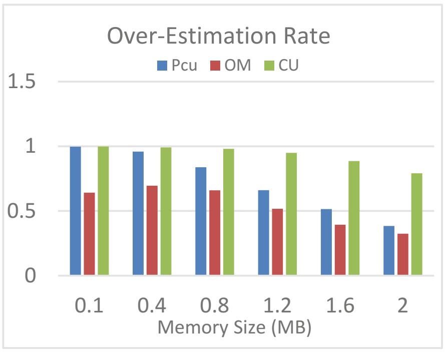

Fig. 10. OER vs. memory sizes

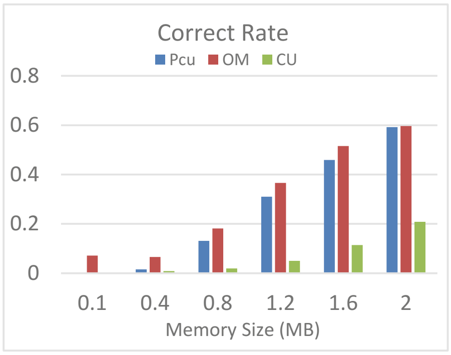

Fig. 11. CR vs. memory sizes

over-estimation-error and improve the correction rate. Our experimental results have well proved the neutralization method indeed improve the overall accuracy.

Speed
Figures 12 and 13 plots the insertion throughput and query throughput of different sketches on different memory sizes increasing from 0.40 MB to 2.00 MB with a step of 0.20 MB. Our experimental results show that the speed of the Pcu sketch and the OM sketch is always higher than that of the CU sketch. The insertion throughput of the OM sketch is about 1.18 and 3.39 times higher than those of the Pcu and CU sketch. The query throughput of the OM sketch is about 1.71 and 4.23 times higher than those of the Pcu and CU sketch.

From the experimental results, we find that the Pcu sketch can significantly improve the insertion throughput but cannot greatly improve the query throughput. As mentioned in the Sect. 3.2, the OM sketch employs two-layer structure and uses word acceleration technique, so that can achieve close to one memory access for each insertion and query. For the Pyramid sketch, although it also uses word acceleration

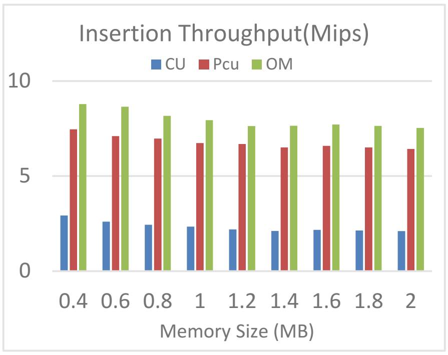

Fig. 12. Insertion throughput vs memory sizes

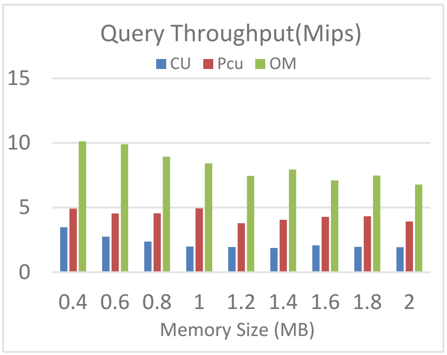

Fig. 13. Query throughput vs memory sizes

technique, the hierarchical structure still highly affects the number of memory accesses for each query. Our throughput experimental results have well proved our viewpoint that the hierarchy technique can slow the insertion and query speed and is more significant to slow the query speed.

Generality
The experimental results above show that the performances of the OM sketch are better than the Pyramid sketch in terms of both speed and accuracy. As mentioned in Sect. 3.3, the sketch that does not have generality is more targeted, and may be more significant to improve the target performances. Thus, our experimental results have well proved our analysis and discussion on Generality.

5 Conclusion

Sketches have been applied to many fields. In this paper, we sort out five important aspects in improving the accuracy and speed of sketch for skewed data streams with limited memory. We provide detailed discussions on the positive and negative effects of typical and latest methods from different aspects on the performances of the sketch. Two other properties of the sketch such as deletion and generality are also discussed. Generally, although the purpose of these aspects are somehow orthogonal to each other, the methods handling them may have effects on more aspects and need more thorough considerations with their limitations and side-effects. Experimental results demonstrate the validity and extendibility of our discussions. We believe our paper can be a good help to the future study of the accurate and fast sketches.

Acknowledgements. This work was supported by Shenzhen Basic Research Program (JCYJ20160525 154348175), the Shenzhen Municipal Development and Reform Commission (Disciplinary Development Program for Data Science and Intelligent Computing) and Shenzhen Key Lab Project (ZDSYS201703031405137O5).

References

1. Yang, T., Zhou, Y., Jin, H., Chen, S., Li, X.: Pyramid sketch: a sketch framework for frequency estimation of data streams. Proc. VLDB Endow. **10**(11), 1442–1453 (2017)
2. Zhou, Y., Liu, P., Jin, H., Yang, T., Dang, S., Li, X.: One memory access sketch: a more accurate and faster sketch for per-flow measurement. In: IEEE GLOBECOM (2017)
3. Manerikar, N., Palpanas, T.: Frequent items in streaming data: an experimental evaluation of the state-of-the-art. Data Knowl. Eng. **68**(4), 415–430 (2009)
4. Cormode, G., Johnson, T., Korn, F., Muthukrishnan, S., Spatscheck, O., Srivastava, D.: Holistic UDAFs at streaming speeds. In: ACM SIGMOD, pp. 35–46. ACM (2004)
5. Cormode, G., Garofalakis, M., Haas, P.J., Jermaine, C.: Synopses for massive data: samples, histograms, wavelets, sketches. Found. Trends Databases **4**(1–3), 1–294 (2012)
6. Roy, P., Khan, A., Alonso, G.: Augmented sketch: faster and more accurate stream processing. In: ACM SIGMOD, pp. 1449–1463. ACM (2016)
7. Cormode, G., Muthukrishnan, S.: An improved data stream summary: the count-min sketch and its applications. J. Algorithms **55**(1), 58–75 (2005)
8. Cormode, G., Hadjieleftheriou, M.: Finding frequent items in data streams. Proc. VLDB Endow. **1**(2), 1530–1541 (2008)
9. Chen, A., Jin, Y., Cao, J., Li, L.E.: Tracking long duration flows in network traffic. In: IEEE INFOCOM, pp. 1–5. IEEE (2010)
10. Liu, Z., Manousis, A., Vorsanger, G., Sekar, V., Braverman, V.: One sketch to rule them all: rethinking network flow monitoring with UnivMon. In: ACM SIGCOMM, pp. 101–114. ACM (2016)
11. Gilbert, A.C., Strauss, M.J., Tropp, J.A., Vershynin, R.: One sketch for all: fast algorithms for compressed sensing. In: ACM STOC, pp. 237–246. ACM (2007)
12. Durme, B.V., Lall, A.: Probabilistic counting with randomized storage. In: IJCAI, pp. 1574–1579. Morgan Kaufmann Publishers Inc. (2009)
13. Polyzotis, N., Garofalakis, M., Ioannidis, Y.: Approximate XML query answers. In: ACM SIGMOD, pp. 263–274. ACM (2004)
14. Estan, C., Varghese, G.: New directions in traffic measurement and accounting. ACM Trans. Comput. Syst. **21**(3), 270–313 (2002)
15. Powers, D.M.W.: Applications and explanations of Zipf's law. Adv. Neural. Inf. Process. Syst. **5**(4), 595–599 (1998)
16. Adamic, L.A., Huberman, B.A., Barabási, A.L., Albert, R., Jeong, H., Bianconi, G.: Power-law distribution of the World Wide Web. Science **287**(5461), 2115 (2000)
17. Yang, T., Liu, L., Yan, Y., Shahzad, M., Shen, Y., Li, X., Cui, B., Xie, G.: SF-sketch: a fast, accurate, and memory efficient data structure to store frequencies of data items. In: IEEE ICDE. IEEE (2017)
18. Graham, C.: Sketch techniques for approximate query processing. Found. Trends Databases (2011)
19. Qiao, Y., Li, T., Chen, S.: One memory access bloom filters and their generalization. Proc. IEEE INFOCOM **28**(6), 1745–1753 (2011)