Improved big data filtering algorithm based on bloom filter

Yujie Liang¹, Ying Yu², Wenhao Ouyang³
¹School of College of Software, Jiangxi Normal University, Jiangxi 330022, China
²School of College of Software, Jiangxi Normal University, Jiangxi 330022, China
³School of College of Software, Jiangxi Normal University, Jiangxi 330022, China

Corresponding author and e-mail: Yujie Liang, luck6393@163.com

Abstract. This paper focuses on the Bloom Filter deformation algorithm. The storage space occupied by Bloom Filter is independent of elements, which has high space efficiency and low time complexity of insert and query operations. However, as the number of added collections increases, the Bloom Filter’s error rate increases, thus filtering out a lot of non-repeating information. In order to solve this problem, the standard Bloom Filter is improved by introducing an algorithm of the structure of the master-slave Bloom Filter. Only when the master-slave Bloom Filter produces errors, the filter is considered as an error. At the same time, by dynamically increasing the number of filters, the growth of misjudgment rate is delayed. This paper also involves an improved Bloom Filter algorithm to reduce the error rate of the repeated data in the judgment filter.

1. Introduction
The arrival of the era of big data is both an opportunity and a challenge. The amount of big data is huge and the content is complex. How to obtain useful information has become a knowledge. In this process, we need to determine whether the page to be collected has already been collected [1]. This requires us to determine whether the URL is repeated, which requires the design of an efficient storage model with a low misjudgment rate, which ordinary hash functions can no longer meet [2]. The dynamic master-slave Bloom Filter proposed in this paper greatly reduce the storage requirements and the misjudgment rate of standard Bloom Filter.

If we want to determine whether an element exists in a collection, we usually store all the elements and walk through the collection to see if it does. Do this by designing data structures, such as tables, trees, and hash tables. In general, the use of time for space, or the use of space for time, in practical applications, need to find a balance, better use of resources to get the maximum benefit.

Under more stringent reaction-time conditions, if we store all the data in memory, we need more and more storage space and take longer and longer to retrieve, which leads to excessive storage costs and waste of time. The problem now is that as data becomes more abundant, time and space needs are met. We need a data structure and an algorithm that takes less time and space. Bloom Filter is one solution. This is a very long binary vector and a series of random mapping features. Bloom Filter is a kind of random data structure with high spatial efficiency. It USES a bit array to represent a set succinct and can judge whether an element belongs to the set or not. This efficiency of Bloom Filter comes at a cost: when determining whether an element belongs to a set, it is possible to mistake an element that does not belong to the set for a false positive. Therefore, Bloom Filter is not suitable for
applications with "zero errors". In the application of low error tolerance, Bloom Filter can achieve great savings in storage space by making few errors [5-7].

2. Principle and improvement

2.1. The principle of analytic
Bloom Filter principle is that an element through a number of hash function mapping into a number of points in the array to put them into 1, data may be repeated in the judgment, only need to consider the elements of a number of key value is 1, you can judge if for 1 then this will be a large data may already exists, if there is a zero does not exist.

![Figure 1. The principle of Bloom filter.](image)

2.2. Disadvantages
The reason why Bloom Filter can achieve high efficiency in time and space is that it sacrifices the accuracy of judgment, the convenience of deletion, and there is misjudgment. The element to be checked may not be in the container, but the value in k positions obtained after hash is 1. At the same time, another difficulty is that you cannot delete, because if you delete the key value of one data, it may affect other data, that is, you cannot accurately judge whether other data has existed or not. An element that is put into the container maps to the k positions of the array to be 1. When it is deleted, it cannot be simply set to 0 directly, which may affect the judgment of other elements. Counting Bloom Filter can be used.

2.3. Misjudgment rate
Through previous articles, we can clearly understand that a Bloom Filter has the following parameters:
Table 1. Parameters of the Bloom Filter.

| Variables | Description of variables |
|-----------|--------------------------|
| m         | Width of bit array (bit number) |
| n         | The number of elements added to it |
| k         | Number of hash functions used |
| F         | False Positive |

The F (misjudgment rate) of Bloom Filter satisfies the following formula:

\[(1 - (1 - \frac{1}{m})^{kn})^k \approx (1 - e^{-kn/m})^k\]  

When there are certain m and n, the k value that minimizes F (misjudgment rate) is:

\[\frac{m}{n} \ln 2 \approx \frac{9m}{13n} \approx 0.7 \frac{m}{n}\]  

The current F (misjudgment rate) is:

\[\left(\frac{1}{2}\right)^k \approx 0.6185^{m/n}\]  

According to the above formula, for any given F, we have:

\[n = m \ln(0.6185) / \ln(F)\]  

Meanwhile, we need k hash functions:

\[k = -\ln(F) / \ln(2)\]  

Since k is a positive integer, in practical application, the above equation is used to obtain the actual F:

\[F = (1 - e^{-kn/m})^k\]

2.4. Improved algorithm

Based on the analysis of the misjudgment rate, we find that the reason why the Bloom Filter algorithm can generate misjudgment is that the key value after the element passes through the hash function is not a one-to-one mapping relation. As there are more and more data, the probability of misjudgment increases. In extreme cases, when every value in a bit array is 1, any data will be judged as existing. At this time, there is a solution: dynamically increase the length of the bit array, so that the number of array elements with the value of 0 is maintained at a certain percentage, so that the misjudgment rate is greatly reduced, which is also known as the dynamic Bloom Filter model. Based on Bloom Filter model, an improved structure of "master-slave" Bloom Filter is proposed in this paper, which further reduces the misjudgment rate.
2.5. Base algorithm flow

① A data need to judge whether there is duplication;
② The data is obtained through two different hash functions, hash1 and hash2, respectively, to obtain two different hash key values;
③ If one of the arrays of all the corresponding hash key values of the master Bloom Filter is 0, set the array of all the corresponding hash key values of the master Bloom Filter to 1, jump ⑥;
If all arrays of the corresponding hash key values for the master Bloom Filter are 1, jump ④;
④ If one of the arrays of all the corresponding hash key values of the slave Bloom Filter is 0, jump ⑥;
If all arrays of the corresponding hash key values for the slave Bloom Filter are 1, jump ⑤;
⑤ Output data duplication;
⑥ Set the array of all the corresponding hash key values of the slave Bloom Filter to 1, and there is no duplication in the output data.

3. Main results
In this chapter, the misjudgment rate of the master-slave bloom filter is analyzed. We use F1 and F2 to represent the master-slave Bloom Filter respectively. F1(S) means that when data S enters the master Bloom Filter, for any x∈U, satisfied i∈[1,k], V1(hi(x))=1, Where h(x) represents the i th hash function, the key value of the map when x enters. V(h(x)) represents the value at h(x) is equal to 0 or 1. The misjudgment rate of master Bloom Filter is P(F1(S)-S). Similarly, we know that the misjudgment rate of salve Bloom Filter is P(F2(S)-S).
Therefore, the misjudgment rate of the master-slave Bloom Filter is
\[ F = P((F1(S) - S) \cap (F2(S) - S)) \]  
(7)

Based on the above analysis, the misjudgment rate of the master-slave Bloom Filter is much lower than that of the standard Bloom Filter. If \( h(x) \) in the master-slave filter is independent of each other, the misjudgment of the master-slave filter does not affect each other. When the Bloom Filter \( m, n \) and \( k \) of the master-slave structure are all the same, the misjudgment rate is as follows:
\[ F = (1 - e^{-kn/m})^{2k} \]  
(8)

3.1. Data result
In this chapter, the misjudgment rate of the master-slave structure Bloom Filter and the standard Bloom Filter was tested. After several rounds of tests, we found that the misjudgment rate increased most intuitively when the data range was \( 10^7-10^8 \). Take the mean of multiple groups of data, the specific results are shown in the figure below:
4. The data analysis
Judging from the growth curve of misjudgment rate in the figure above, the improved structure of the master-slave Bloom Filter can slow down the growth of misjudgment rate. Especially when the data volume is less than $5 \times 10^7$, the improved structure of master-slave Bloom Filter can effectively reduce the misjudgment rate by about 20%. However, when the data reaches $10^8$, the bit array from Bloom Filter is almost all 1, and the error rate is similar to that of standard Bloom Filter.

Table 2. Complexity comparison.

|                  | Space complexity | Time complexity |
|------------------|------------------|-----------------|
| Standard structure | O(m)             | O(k)            |
| Master-Slave structure | O(2m)          | O(2k)           |

The space complexity of the standard Bloom Filter is a constant O(m), while the retrieval time complexity is a constant O(k). In contrast, with a 1% false positive rate and an optimal value of k, a Bloom filter requires only 9.6 bits per element (regardless of the size of the element). This advantage comes from the compactness of inherited arrays and from their probabilistic properties. A 1 percent false alarm rate can be reduced 10 times by adding about 4.8 bits per element. The space complexity of the master-slave structure Bloom Filter is a constant O(2m), while the retrieval time complexity is a constant O(2k). Compared with the standard bloom filter, the master-slave bloom filter requires twice as much space and time, but it greatly reduces the misjudgment rate.

However, when the amount of data exceeds the maximum of the master-slave Bloom Filter, the misjudgment rate is similar to that of the standard Bloom Filter. Therefore, according to the Bloom Filter model, we can use another method to dynamically increase the number of master-slave Bloom Filter. Dynamic master-slave Bloom Filter can delay the growth of misjudgment rate, and the more the number of filters, the lower the misjudgment rate.

5. Conclusions
Based on Bloom Filter, this paper improves its structure, adopts the master-slave structure, and dynamically increases the number in order to realize the exchange of a certain amount of space for the judgment accuracy rate and reduce the misjudgment rate of the big data set. In practice, the efficiency of the crawler can be greatly improved by filtering the repeated URL in the crawler.
References

[1] Xu na, Liu siwei, Wang xiang, Ni weiming. Duplicate Web Page Elimination Based on Bloom Filter [J]. Microcomputer application. 2011, (3),27 10.3969/j.issn.1007-757X.2011.03.016.

[2] Hui-jun meng. Research and application of URL deduplication algorithm based on Bloom Filter algorithm[D]. China's out of school education (Basic Education Edition) 2019.

[3] Zhou zhou, Fu wenliang, song tian, liu qingyun. “Fast URL Lookup Using Parallel Bloom Filters”[J]. Electronic journals 2015, (9) doi:10.3969/j.issn.0372-2112.2015.09.023.

[4] Ding zhenguo, Wu precious, Xin youqiang. Research of Large-scale URL Filter Based on Bloom Filter[J]. Knowledge organization and knowledge management, 2008, (3) doi: 10.3969/j.issn.1003-3513.2008.03.008.

[5] Yan hua yun, Ji guan hong. Bloom Filter research progress [J]. Journal of telecom science, 2010, 26 (2) : 31-36 doi: 10.3969 / j.j SSN. 1000-0801.2010.02.008.

[6] Wu xi, Yu Neng hai, Zhang Wei ming. An improved multi-keyword fuzzy search scheme based on BloomFilter over encrypted text [J]. Control and Decision 2019, 34(1) doi: 10.13195/j.kzyjc.2017.0802.

[7] Rao Wen, Chen Xu. The Query Optimization and Application of BloomFilter for Large Dataset [J]. Microcomputer Applications 2018, 34(2) doi:10.3969/j.issn.1007-757X.2018.02.021.