Zhu and Zhu *EURASIP Journal on Wireless Communications and Networking* (2018) 2018:43
https://doi.org/10.1186/s13638-018-1043-3

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Real-time positioning of a specific object in the big data environment

Hejun Zhu[1,2*] and Liehuang Zhu[1]

Abstract

Real-time positioning of a specific object in the big data environment can improve the monitoring and management capacity for network data. For the real-time positioning of the specific object, it is necessary to quickly search the network data representing a specific object and match its pattern strings and compare the corresponding Internet protocol (IP) address of the matched network data with the IP address library in real time, so as to determine the position of the specific object. When a traditional method is used for pattern string matching, it will occupy a lot of memories and network resources, thereby reducing the positioning effect of the specific object in the big data environment. A positioning method for a specific object of high performance and multi-pattern matching based on three indexes in the big data network environment is proposed in this paper. Firstly, the initialization of Modified Wu-Manber (MWM) algorithm was carried out, and the algorithm was used to match the network data continuously. Secondly, the three indexes were used to improve the MWM algorithm, and the real-time and fast positioning of a specific object in the big data environment was completed by the Third Index Modified Wu-Manber (TMWM). The experimental results show that compared with the traditional method, the proposed algorithm reduces the pattern string matching scope of network data representing the specific object, improves the search speed of the specific object, and locates the specific object in the big data environment in an effective and rapid manner.

Keywords: Big data, Specific object, TMWM, Positioning

1 Introduction

With the popularization of network application, the network scale has been expanded, the services carried in the network have also become more and more abundant, and network users enjoy much more conveniences. However, the enlarged network application scope and the increasingly serious information disclosure problem have negative impacts on the social and personal security. As a result, it is crucial to monitor the spread of illegal and harmful information, mine the hidden dangers behind big data, improve the information processing technology, and maintain the network security. Under such a background, the accurate and fast positioning of a specific object in the big data environment can improve the monitoring and management capabilities of network data, and this is one of the most difficult problems in the field of network security [1–3].

The typical single-pattern matching algorithm mainly included the Knuth-Morris-Pratt (KMP) algorithm [4, 5] and Boyer-Moore (BM) algorithm [6, 7]. The two most popular typical pattern-matching algorithms were Aho-Corasick (AC) algorithm [8, 9] and Wu-Manber (WM) algorithm [10–13], which were exactly an extension of these two single-pattern matching ideas. AC algorithm is an automation algorithm, in which firstly a search tree based on pattern string sets was established, and then, the text is progressively scanned from front to back; since the size of search tree was related to the size of the character set, AC algorithm needed larger memory resources. Later, some researchers' improved algorithms also focused on the compression of the storage space and shifting to accelerate the comparison speed. For Wu-Manber algorithm, the jumping thought and HASH hashing method in BM algorithm were applied to firstly jump by virtue of bad character table or bad block table

* Correspondence: xjd_hjzhu@163.com
[1]School of Computer Science, Beijing Institute of Technology, Beijing 100081, China
[2]Esafenet Corp, Building A, No.39, Xi'erqi Street, Haidian District, Beijing 100085, China

and then utilize HASH table to screen the starting position that should to be matched, and finally compare the pattern strings one by one.

For multi-pattern string matching, AC algorithm was much more efficient than KMP, BM, and other single-pattern matching algorithms, but AC algorithm fails to jump the character string in searching and the character is inputted by order, and the algorithm cannot skip unnecessary comparison; the major disadvantages of dependence on character set and large memory occupation resulted in the poor performance of AC algorithm. While the execution time of Wu-Manber algorithm did not increase proportionally with the increase of pattern string, it is much less than the sum of the time of matching each text with each keyword and BM algorithm. In addition, the time complexity of such algorithm can reach $O(B\,n/m)$ under the best condition (n is the size of text, m is the shortest pattern length, and B is the length of block character). By contrast, Wu-Manber, with its insensitive characteristics to the pattern quantity and character set and lower time complexity, is more suitable for the requirement of clue keyword search.

A fast positioning method for network data with Modified Wu-Manber (MWM) high performance and multi-pattern matching based on three indexes in the big data network environment is proposed in this paper. Firstly, the initialization of the MWM algorithm was carried out, and the algorithm was used to match the network data continuously. Secondly, the three indexes were used to improve the MWM algorithm, and the real-time and fast positioning of a specific object in the big data environment was completed by TMWM. The experimental results show that compared with the traditional method, this improved algorithm reduces the pattern string matching scope of network data representing the specific object and improves the search speed of the specific object and can effectively and rapidly locate the specific object in the big data environment.

In view of the similarity with the network content security industry, the MWM algorithm from the snort feature matching module is extracted in this paper, which is an improved Wu-Manber algorithm [14–18].

2 Principle

In order to match the multi-keyword list in the big data environment more rapidly, it is necessary to apply efficient multi-pattern matching algorithm but not simple single-pattern matching. Through many practices and researches, it is found that the improved multi-pattern matching algorithm can solve the keyword list matching at the level over ten thousands under the big data traffic environment. The multi-keyword matching actually is to search all positions of all pattern strings in the pattern string set P in the text T.

As shown in Fig. 1, based on the keyword matching model of multi-pattern matching, firstly, the network traffic is collected in real time and the data contents of the network are analyzed by the big data traffic. Afterwards, based on the multi-pattern matching search algorithm, a quick and real-time comparison is made between the pattern string sets P formed by the multi-keyword list and the real-time analytical big data network traffic; if matched, the researcher shall output a list of hit keywords and their corresponding texts T, otherwise continue to the next big-data network traffic analyzed in real time and compare with the pattern string sets P formed by the keyword list.

In this paper, a multi-pattern matching algorithm is used to quickly find and position the network data in the big data environment. Generally, it is an extension of single-mode matching algorithm in the multi-pattern environment. The basic idea is as follows:

$$\Sigma = \{E_1, E_2, \cdots, E_n\} \tag{1}$$

$$P = \{P_1, P_2, \cdots, P_k\} \tag{2}$$

$$T = T[1 \cdots n] \tag{3}$$

The character set Σ in Eq. (1) is composed of character strings. P in Eq. (2) is a set of pattern sets, and each pattern P consists of character strings coming from the

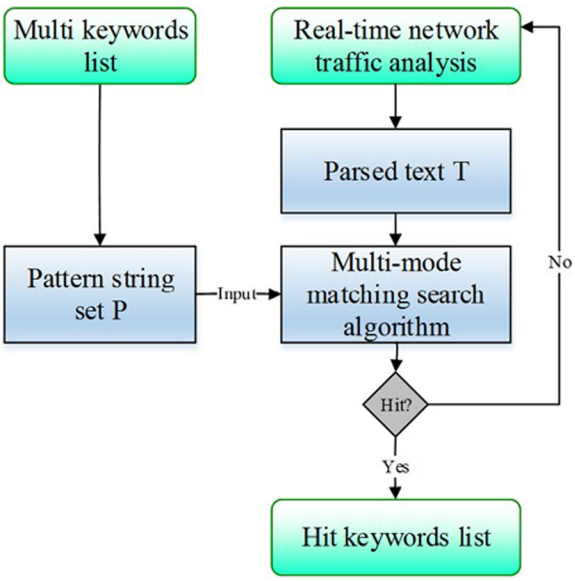

Fig. 1 Multi-keyword matching model based on multi-pattern matching. The network flow is real-timely collected and analyzed, and then, the pattern string set P formed by the multi-keyword list and the network flow real-timely analyzed are rapidly and real-timely compared and searched

character set Σ. In Eq. (3), the text string T with a given length of n is indicated, and each character of the text string also comes from the character set Σ. The multi-pattern matching is to find the emergence of each pattern in the pattern set P:

$$X = \{x_1, x_2, \cdots, x_B\} \tag{4}$$

$$\text{index} = \text{hash}(X) \tag{5}$$

$$\text{SHIFT}[\text{index}] = \begin{cases} m-B+1, & \text{if } X \text{ doesn}'\text{t appear in any pattern} \\ \min\left\{m-j | X[k] = P_i[j-B+k], \forall k, 1 \le k \le B\right\}, & \text{if } X \text{ appears in some patterns} \end{cases} \tag{6}$$

where, in Eq. (4), X is a substring with a length of B to be compared in T, and an index value is obtained through mapping by a hash function. The index value is taken as an offset to obtain the value in the SHIFT table, and the value is decided to read the number of bits that can be skipped after the current substring X, suppose that X is mapped to the entry whose index is the entry of the SHIFT table, i.e., Eq. (5). In Eq. (6), it is necessary to map the substring with a length of B in each pattern to the SHIFT table, in which the mapping function is the same hash function as Eq. (5).

2.1 Algorithm initialization

Before TMWM matching algorithm is used, it is necessary to carry out initialization processing for the rapid research algorithm of network data, and the realization process is described as follows:

Step 1: Input all pattern strings of network data representing the specific objects into a pattern string array and quickly sort them from small to large according to the size of ASCII code of character (American Standard Code for Information Interchange).
Step 2: Take the length m of the shortest pattern string in the network data representing the specific object as the size of the matching window (window matching having the character string to be matched) and, at the same time, as the maximum distance of jumping to initialize the jump table.
Step 3: Utilize the information about the first n character of pattern string of all network data to rewrite the jump table (see the following algorithm for detailed construction process of jump table), and take the jump block (length of B) as the basic unit of searching the jump table to obtain one or more characters (in general, two or three characters).
Step 4: Build up HASH table according to the prefix (the first one to two characters) information about the pattern string of each network data, and take subscript of the first pattern string started with such prefix as the table item in HASH table. While building up HASH table, create a prefix array to store the number of pattern strings having the same prefix.

2.2 Matching process of algorithm

Based on the initialization processing of network data representing specific object above, the pattern strings of the network data is matched, and the detailed implementation process is described as follows:

Step 1: Firstly, take the initial address of the network data as the initial address of the matching window; search for the jump table according to the suffix of the matching window in Chinese text (length of the jump block) to obtain the maximum distance of the jump searching and jump until the jump fails.
Step 2: When the jump fails (that is, the table item of jump table is zero), obtain the prefix of the matching window to search for HASH table, in order to obtain the initial positioning of network data customer pattern with the same prefix in the pattern string array.
Step 3: According to the initial starting positioning information obtained in the previous step and the number of network data service client mode strings of the prefix in the prefix array, the network data service client in the matching window is compared with the mode strings on a byte-by-byte basis to move the matching window one byte backwards for re-matching, whether finding out or not, until the end of the text.
The detailed matching code is described as follows:
Step 1: Use m − B + 1 to fill in Shift [ ] table.
Step 2: For each pattern string

```
{//calculate the jumping distance for each block character in current pattern string
        for (i=m-1;i>=(B-1);i--)
        {
        HASH = hashBlock ([PATTERN [i],…,PATTERN[i-B+1]);
        if (SHIFT[hash]>=m-1-i)
        SHIFT[hash]=m-1-i;
        }
}
```

3 Results and discussion

3.1 Results and discussion of MWM algorithm

In order to better test the performance of MWM algorithm and the current existing STRFINDUNI algorithm, these two algorithms are separately extracted to put in the same Linux environment for testing.

The hardware environment of the test: Server Intel (R) Xeon (R) CPU 5120@ 1.86 GHz 4 kernel 4 G memory.

Test samples: SOGOU Chinese Thesaurus, the test samples are taken from the thesaurus.

Test method: The network data matching process is simulated in real environment, without any text change, and the number of network data pattern strings progressively increases in a linear form, resulting in the proportional change of the hit times. The initialization time consumption and time consumption of network data matching of these two algorithms are respectively measured as in Figs. 2 and 3, respectively.

According to the comparison results made between Figs. 2 and 3, it is obvious that the MWM algorithm spends much more time than the STRFINDUNI algorithm both in the initialization process and in the pattern matching process. The time consumption of MWM algorithm is far more than that of STRFINDUNI algorithm regardless of initialization process or pattern matching process. However, in terms of the extensive application of such algorithm in snort application, the main factors of researches in this paper cause the above differences to be concluded and summarized as follows:

Influence factors concerning the network data service client matching:

(a) Different HASH methods: HASH in STRFINDUNI applies the 1-stage HASH followed by two stages of indexing to construct. During the search process, the first 6 bytes of the head address of chain list of the network data service client pattern string list are found through the screening process and then matched to the network data service client pattern string in the linked list one by one. However, the HASH process in MWM uses only 1-stage HASH level to accelerate the finding, matching them one by one in the network data service client pattern string array. This difference in HASH approach is magnified by testing with dictionary files because the number of phrases that begin with the same first 2 bytes is too large, often up to more than 100, and the number of words beginning with the same first 6 bytes is greatly reduced, usually less than three.

(b) Different basic units of matching of network data: Because STRFINDUNI is used to match the text with Unicode coding method, the basic unit of matching is the word, and the shifting, HASH, and other operations in the algorithm are carried out in the word unit. However, MWM is carried out in byte, and for MWM algorithm known for reducing the number of comparison times by jumping, the differences in the basic unit of pattern string matching for such network data will at least cause a half distance reduction of movement in jumping.

Influence factors concerning initialization:

(a) Influence of sorting operation. After the pattern string of network data is read in the MWM, all the network data service client mode strings are quickly sorted. This operation is not obvious when the number of pattern strings is small, but when the number of pattern strings of network data is very large, this operation will seriously affect the initialization speed. The most important thing is that such a large overhead operation is not well used later.

(b) The overhead for pattern string copy of network data. The pattern string of STRFINDUNI is read during initialization, and the pattern string of network data is stored in the form of memory copy. However, the pattern string of network data in MWM is only carried out by the pointer passing, reducing the overhead of application for memory and copy.

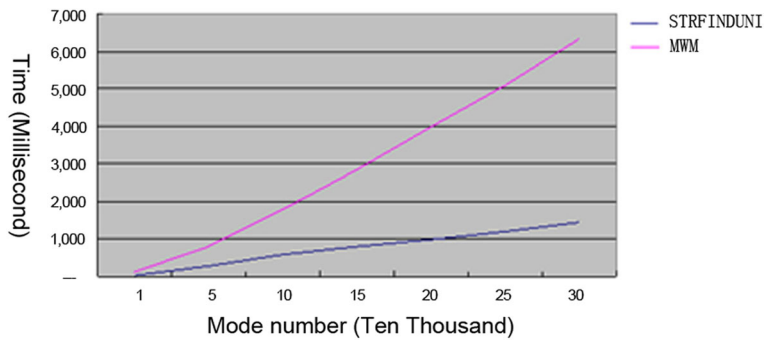

Fig. 2 Test results of the initialization process in real environment. The abscissa indicates the mode number, and the unit is ten thousand. The ordinate indicates the time, and the unit is millisecond

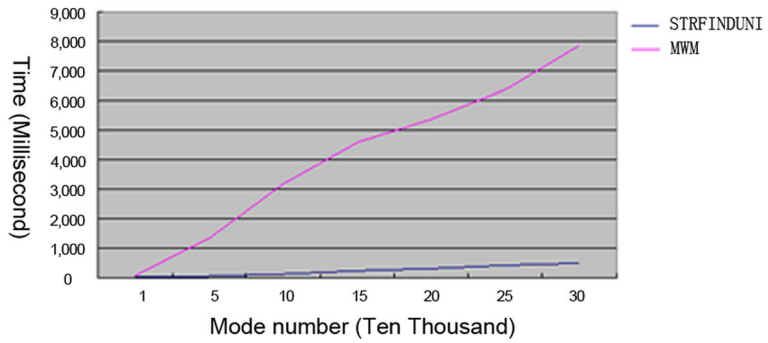

Fig. 3 Test results of matching process in real environment. The abscissa indicates the mode number, and the unit is ten thousand. The ordinate indicates the time, and the unit is millisecond

3.2 Results and discussion of TMWM algorithm

Considering the above defects is not a serious problem in the idea of MWM algorithm itself, but rather, the idea of MWM algorithm is not fully utilized in a specific environment and under specific conditions. Therefore, under the premise of retaining the core idea of the algorithm, this paper proposes an improved algorithm TMWM, and the main modified items are as follows:

(a) Modification of basic unit of matching of network data. The jump table of MWM algorithm is carried out with word as the unit, and the jump operation is completely adopted in the form of word jumping.
(b) Modification of HASH method. The pattern string array of network data upon sorting is fully utilized and two-stage index is added behind HASH table with the minimum cost in combination with two index arrays, and the retrieval of each stage index is carried out by means of binary search.
(c) Modification of sorting operation. Because the index operation and pattern string comparison operation of specific network data only use the first 6 bytes of the pattern string array, it is rather redundant for the quick sorting of the whole pattern string array, and in terms of later application, only the first 6 bytes need to be sorted.

In order to more clearly display the main retrieval process, all table items in Fig. 4 only display two data items, including the number required to be matched at the next stage and the initial positioning of matching at the next stage.

In order to verify the effectiveness of the positioning method of network data with high performance and multi-pattern matching based on pattern string TMWM proposed in this paper, the simulation experiments require to be carried out to compare with the traditional STRFINDUNI algorithm, and the main influence factors of these two algorithms include: number of pattern strings of network data, the size of text of network data, the mean length of pattern string of network data, and the hit times of network data.

In the following, four parameters, including the number of pattern strings, text size, mean length of pattern string, and the hit times, are utilized for testing and corresponding analysis.

(a) Number of pattern strings

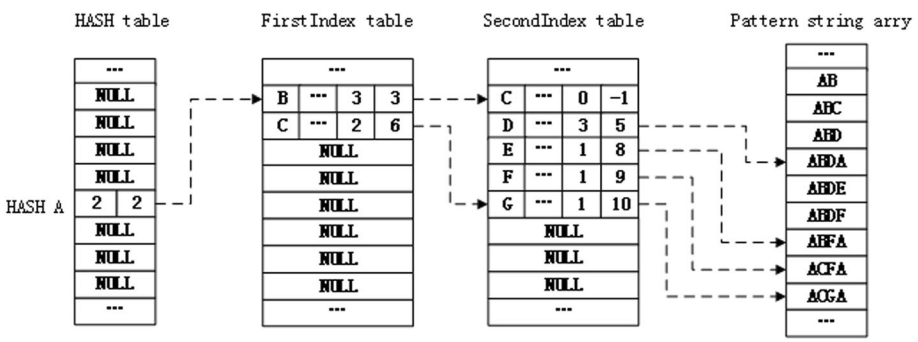

Fig. 4 Part structure of HASH table in TMWM algorithm. All table items only display two data items, including the number required to be match at next stage and the initial location of matching at next stage

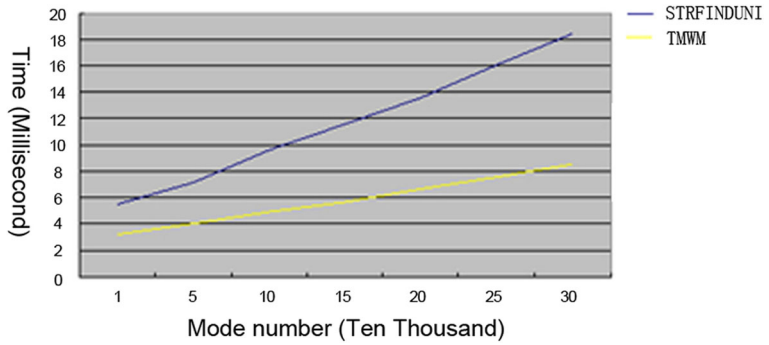

Fig. 5 Test results of the influence of the number of pattern strings on matching process. The abscissa indicates the mode number, and the unit is ten thousand. The ordinate indicates the time, and the unit is millisecond

Test method: The effect of number of the tested pattern string on the matching time and the initialization time is tested under the condition that the mean length of pattern string is unchanged (6 bytes) with unchanged text size and basically constant hit times.

The results are respectively shown in Figs. 5 and 6. As can be seen from the comparison of Figs. 5 and 6, the initialization time of the TMWM algorithm is proportional to the number of pattern strings under other conditions unchanged, whereas the matching time slightly increases with the number of pattern strings.

(b) Text size

Test method: The effect of text size on the matching time and initialization time is tested under the condition that the pattern strings are constant and the hit number of times is basically unchanged.

The results are respectively shown in Figs. 7 and 8. As can be seen from the comparison of Figs. 7 and 8, the matching time of these two algorithms is proportional to the text size under other conditions unchanged, and the initialization time remains constant because the pattern string remains unchanged.

(c) Average length of pattern strings

Test method: The effect of the average length of the pattern string on the matching time and initialization time is tested when the number of pattern strings, the text size, and the number of hit times remain unchanged.

The results are respectively shown in Figs. 9 and 10. As can be seen from the comparison of Figs. 9 and 10, the TMWM algorithm is less affected by the length of the pattern string, whereas the STRFINDUNI algorithm is greatly affected by the length of the pattern string, and it has a big time change especially when matching.

(d) Hit times

Test method: The effect of the hit number of times on the matching time and initialization time is tested when the number of pattern strings, the average length of the pattern strings, and the text size remain unchanged.

The results are respectively shown in Figs. 11 and 12. According to the results of the effects of hit number of times on the initialization process in Figs. 11 and12, the TMWM algorithm has a relatively small effect on the hit

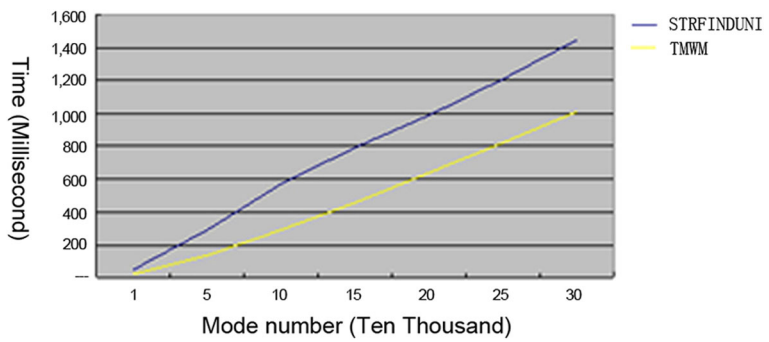

Fig. 6 Test results of the influence of the number of mode strings on initialization process. The abscissa indicates the mode number, and the unit is ten thousand. The ordinate indicates the time, and the unit is millisecond

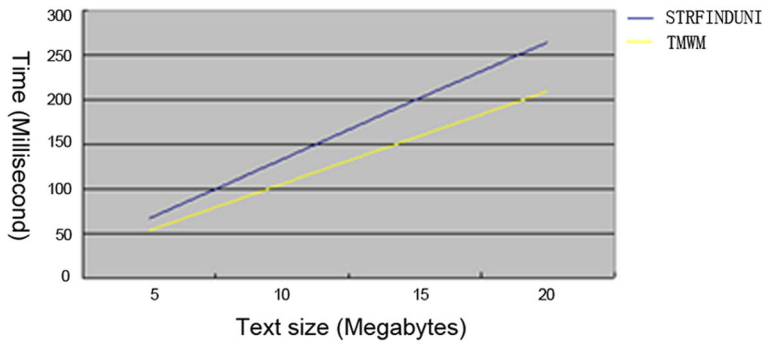

Fig. 7 Test results of the influence of text size on matching process. The abscissa indicates the text size, and the unit is megabytes. The ordinate indicates the time, and the unit is millisecond

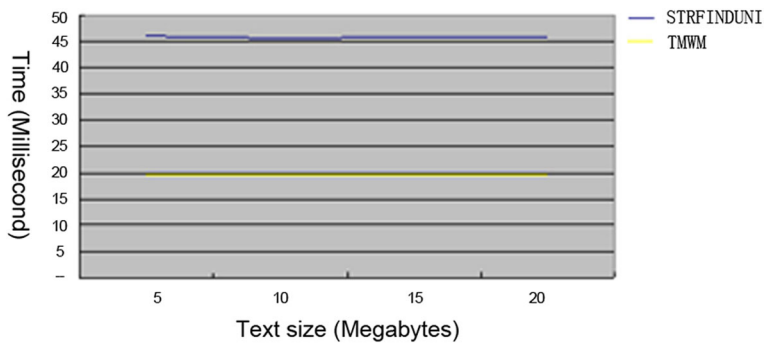

Fig. 8 Test results of the influence of text size on initialization process. The abscissa indicates the text size, and the unit is megabytes. The ordinate indicates the time, and the unit is millisecond

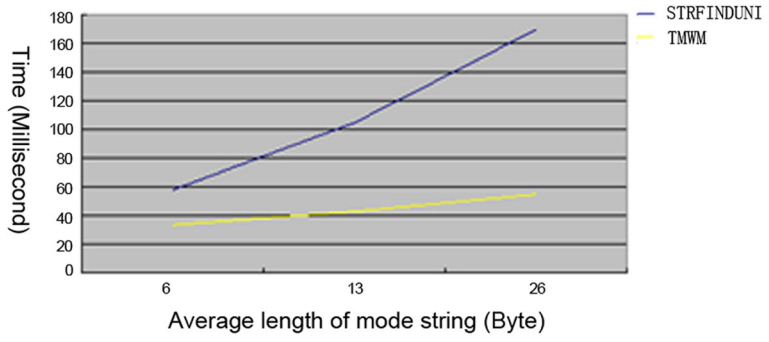

Fig. 9 Test results of influence of pattern string length on matching process. The abscissa indicates the average length of mode string, and the unit is byte. The ordinate indicates the time, and the unit is millisecond

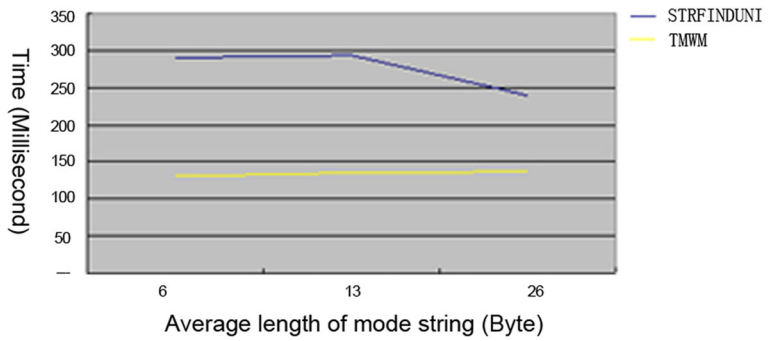

Fig. 10 Test results of influence of pattern string length on initialization process. The abscissa indicates the average length of mode string, and the unit is byte. The ordinate indicates the time, and the unit is millisecond

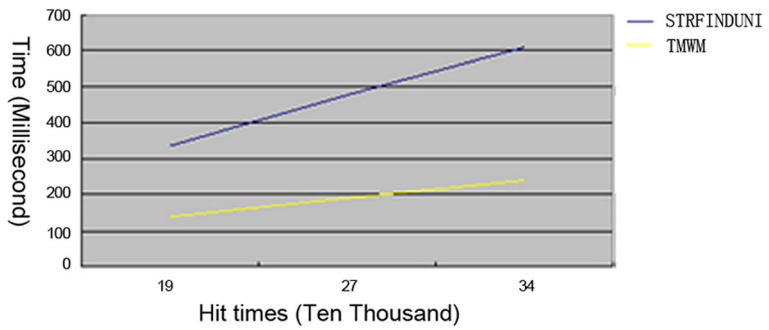

Fig. 11 Test results of the influence of hit times on matching process. The abscissa indicates the hit times, and the unit is ten thousand. The ordinate indicates the time, and the unit is millisecond

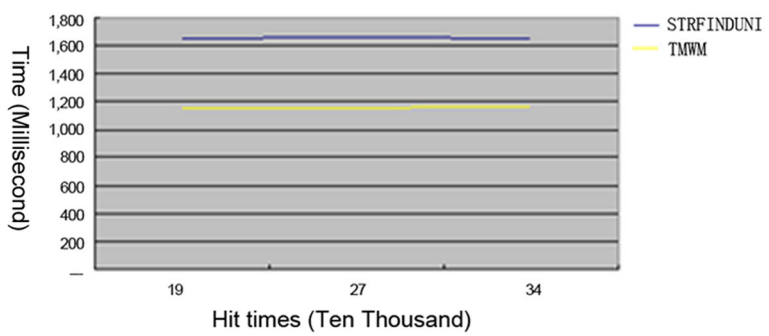

Fig. 12 Test results of the influence of hit times on initialization process. The abscissa indicates the hit times, and the unit is ten thousand. The ordinate indicates the time, and the unit is millisecond

number of times, and the invariant pattern strings lead to no change in the initialization time, so the initialization time of the TMWM algorithm is improved by about 50% as compared with the STRFINDUNI algorithm and the matching time of the TMWM algorithm is increased by about 100% as compared with the STRFINDUNI algorithm.

The above results are obtained through a lot of experiments and analysis. Meanwhile, as the price of hardware resources decreases and more complex regular expressions are used, AC algorithm will have more applications.

4 Conclusions

A real-time positioning method based on the pattern string TMWM with high performance and multi-pattern matching in the network big data environment is proposed. Firstly, the algorithm was initialized, the MWM algorithm was used to continuously match the network data, and the TMWM algorithm was used to improve it. Afterwards, the IP and IP address databases corresponding to the matched network data were compared to complete the positioning of specific objects in the big data environment. Compared with the traditional MWM and STRFINDUNI matching methods, the improved TMWM algorithm may reduce the matching range of the target pattern strings, accelerate the search speed of the pattern strings, and enhance the clue and keyword search speed of about 100%, thus solving the problem of efficiently searching the desired pattern strings in the big data environment. Finally, the effectiveness and feasibility of the proposed algorithm in terms of pattern string matching speed and matching efficiency are verified by comparing and analyzing the number of pattern strings, text size, average length of pattern strings, and the hit number of times.

5 Additional files

Additional file 1: Test samples from the SOGOU Chinese Thesaurus that contains 200 thousand phrases. (TXT 1564 kb)

Additional file 2: Test samples from the SOGOU Chinese Thesaurus that contains 250 thousand phrases. (TXT 1956 kb)

Additional file 3: Test samples from the SOGOU Chinese Thesaurus that contains 300 thousand phrases. (TXT 2350 kb)

Abbreviations

AC: Aho-Corasick; ASCII: American Standard Code for Information Interchange; BM: Boyer-Moore; IP: Internet Protocol; KMP: Knuth-Morris-Pratt; MWM: Modified Wu-Manber; TMWM: Third Index Modified Wu-Manber; WM: Wu-Manber

Acknowledgements

Many thanks to my colleagues, including Zhang C., Zhou H., Wan Y.L., and Dai H.W., who helped me do some experiments, and they gave many suggestions.

Funding

Funding information is not applicable.

Availability of data and materials

These test samples were taken from the SOGOU Chinese Thesaurus, mainly including patternstring-20w.txt, patternstring-25w.txt, and patternstring-30w.txt (Additional files 1, 2, and 3).

Authors' contributions

HZ proposed the improvement method, and the method was tested, compared and summarized by HZ. LZ made an improvement on the method. Both authors read and approved the final manuscript.

Authors' information

Zhu Hejun is a Doctor of Network Information Security and studies at Beijing Institute of Technology as an in-service doctor. His research interests include network information security and big data processing and mining.
Zhu Liehuang is a Doctor of Computer Science and Technology and Professor. He graduated from the Beijing Institute of Technology in 2004. He worked in Beijing Institute of Technology. His research interests include cryptographic algorithms and security protocols, Internet of things security, and cloud computing security.

Competing interests

The authors declare that they have no competing interests.

Publisher's Note

Received: 22 November 2017 Accepted: 27 January 2018
Published online: 17 February 2018

References

1. A Dainotti, A Pescape, K Claffy, Issues and future directions in traffic classification. IEEE Netw. **26**(1), 35–40 (2012)
2. L Peng, B Yang, Y Chen, Z Chen, Effectiveness of statistical features for early stage internet traffic identification. Int. J. Parallel Prog. **44**(1), 181–197 (2016)
3. S Faro, T Lecroq, The exact online string matching problem: a review of the most recent results. ACM Comput. Surv. **45**(2), 1–42 (2013)
4. Aygün S, Güneş EO, Kouhalvandi L, Advances in Information, Electronic and Electrical Engineering IEEE, Python based parallel application of Knuth-Morris-Pratt algorithm (IEEE, Vilnius, 2017), pp. 1–5.
5. Tsarev RY, Chernigovskiy AS, Tsareva EA, Brezitskaya VV, Nikiforov AY, Smirnov NA, Combined string searching algorithm based on Knuth-Morris-Pratt and Boyer-Moore algorithms (IOP, Krasnoyarsk). **122**(1), 012034 (2016).
6. TH Tsai, Average case analysis of the Boyer-Moore algorithm. Random Struct. Algoritm. **28**(4), 481–498 (2010)
7. F Bassino, J David, C Nicaud, Average case analysis of Moore's state minimization algorithm. Algorithmica **63**(1-2), 509–531 (2012)
8. Vakili S, Langlois JMP, Boughzala B, Savaria Y, Symposium on Architectures for NETWORKING and Communications Systems, Memory-efficient string matching for intrusion detection systems using a highprecision pattern grouping algorithm (IEEE, Santa Clara, 2016), pp. 37–42.
9. Arudchutha S, Nishanthy T, Ragel RG, IEEE International Conference on Industrial and Information Systems, String matching with multicore CPUs: performing better with the Aho-Corasick algorithm (IEEE, Peradeniya, 2014), pp. 231–236.
10. Li X, Wu L, Proceedings of the 2015 Chinese Intelligent Automation Conference, A multi-modal searching algorithm in computer go based on test (Springer, Heidelberg), **336**, 143-149 (2015).
11. Ng T, Rappaport D, Kai S, Developments in language theory, State complexity of neighborhoods and approximate pattern matching (Springer International Publishing, Switzerland, 2015), pp. 389-400.
12. Aldwairi M, Al-Khamaiseh K, Web Applications and NETWORKING, Exhaust: optimizing Wu-Manber pattern matching for intrusion detection using Bloom filters (IEEE, Sousse, 2015), pp. 1–6.

13. Alkhathami M, Alazzawi L, Elkateeb A, Models and techniques analysis of border intrusion detection systems. Global Journal of Research in Engineering. **15**(7), 35–43 (2015)
14. N Tuck, T Sherwood, B Calder, G Varghese, Deterministic memory efficient string matching algorithms for intrusion detection. IEEE INFOCOM. **4**(11), 2628–2639 (2004)
15. Patel B, Efficient string matching algorithm for intrusion detection, International Journal of Computer Engineering and Science. **1**(1), 9-17 (2015).
16. Kouzinopoulos CS, Michailidis PD, Margaritis KG, Multiple string matching on a GPU using cuda. Scalable Computing. **16**(2), 121-137 (2015).
17. G Navarro, NR-GREP: a fast and flexible pattern-matching tool. Software Pract Experience **31**(13), 1265–1312 (2010)
18. CN Modi, DR Patel, A Patel, R Muttukrishnan, Bayesian classifier and snort based network intrusion detection system in cloud computing. International Conference on Computing Communication & NETWORKING Technologies. IEEE **90**, 1–7 (2012)