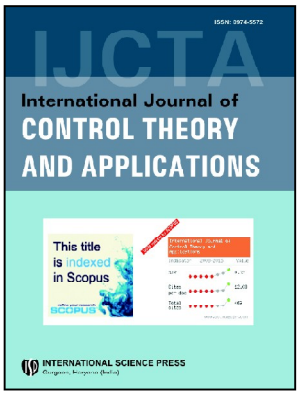

International Journal of Control Theory and Applications

ISSN : 0974-5572

Volume 10 • Number 39 • 2016

A New Multi Pattern Multi Processor Parallel String Matching Algorithm with While Shift

[1]K. Butchi Raju and [2]Chinta Someswara Rao

[1] Department of CSE, GRIET, Hyderabad, India, E-mail: butchiraju.katari@gmail.co
[2] Department of CSE, S R K R Engineering College, Bhimavaram, W. G. District, A.P. India
E-mail: chinta.someswararao@gmail.com

***Abstract*:** In present days a huge volume of digital data is observed because of increasing usage of internet by so many people. Extraction of relevant information from these huge digital data is a difficult task, therefore information retrieval systems (IRS) need the high performance algorithms for searching. The information retrieval systems basically use the string matching algorithms, once the efficiency of the string matching algorithms is improved; the relevant retrieval efficiency of the information retrieval systems automatically increases. For this purpose in this paper, we propose multi pattern multi-processor parallel string matching algorithms for sequential and concurrent environments.

***Index Terms*:** Digital data, IRS, String matching, Multi pattern, parallel string matching

1. INTRODUCTION

String matching is one of the dominant concepts in computer science areas like text editors, text mining, data mining, forensics, intrusion detection systems, plagiarism, DNA sequence analysis and many more. Due to its wide range of application areas a number of algorithms were developed since 1970 [1, 2, 3, 4, 5, 6, 7]. String matching algorithms can be divided into single pattern and multi pattern algorithms. Based on the searching nature single pattern string matching can be classified as prefix, suffix and substring matching algorithms. In single pattern matching various string matching algorithms are proposed by different researchers. Brute force[1] is the first string matching algorithm, in which characters of the pattern are compared with text, if mismatch/ complete match occurs then it shifts exactly one position. Later Knuth-Moms-Pratt (KMP) [8] proposed another prefix based string matching algorithm, it also compares the first character of the pattern with first character of the text, if mismatch/complete match occurs then shift position decision is taken based on correctly matched characters. Boyer-Moore(BM)[9] is another single pattern matching algorithm, in which the right most character of the pattern is compared with the corresponding character of the text, if mismatch/ complete match occurs then it shifts the pattern based on good suffix and bad character rule. The BM has advantages and disadvantages, with these advantages and disadvantages many variants of BM have been developed.

Multi pattern string matching algorithms are somewhat different from single pattern string matching algorithms. In single pattern string matching algorithms searching is performed for single pattern whereas in multi pattern string matching algorithms the searching is performed for multiple patterns concurrently. Aho-corasick (AC)[10] and We Manber (WM)[11]are the very starting multiple pattern string matching algorithms. Aho-corasick[10] has two phases called preprocessing and searching. In preprocessing it uses TRIE (text retrieval) for pre-treatment of the patterns whereas it uses the automaton concept in searching. WM [11] is the invariant of BM[9]and Rabin Karp[12] algorithms, in searching it compares the pattern's right most character with corresponding character of the text, if mismatch/complete match occurs then it shifts the position based on good suffix and bad character rules.

Many improved WM[11] are found in the literature from Quick Wu Manber to B-LAyered bad character Shift Tables (BLAST).

Yang *et al.* proposed Quick Wu Manber (QWM)[13] which is invariant of quick search[14] algorithm. It consists of preprocessing and searching. In preprocessing the head table is constructed with the first two characters of the pattern. In searching the position from head table is read, the pattern and text are fixed then the right most character of the pattern is compared with the corresponding character of the text, if complete match/mismatch occurs then the next position from head table is read.

Chen Zhen *et.al.,* proposed Improved WM algorithm[15], it also invariant of basic WM[11]. It also has two phases called preprocessing and searching. In preprocessing two shift tables good suffix table and hash table are built. In searching, the entry from hash table is read, the pattern and text is fixed then the right most character of the pattern is compared with the corresponding character of the text, if complete match/mismatch then the entries from hash and good suffix tables is read.

Liuling Dai *et.al.* proposed Quick Multiple Matching (QMM)[16] string matching algorithm, which also consists of preprocessing and searching. In preprocessing shift table and hash table are computed. Shift table consists of two grams of given pattern and hash table consists of hash value each of two gram. In searching the entry from hash table is read, the pattern and text is fixed then the right most character of the pattern is compared with the corresponding character of the text, if complete match/mismatch then the entries from hash and shift tables are read.

Xiaoping Chen *et al.,* proposed High Concurrence Wu Manber (HCWM) [17] string matching algorithm, it also one variant of the WM. HCWM reads multiple patterns, groups them basing on their length and also splits input text and assigns it to different threads. In searching each thread compares the right most character of the pattern from group, if complete match/mismatch occurs then bad character and good suffix rules are used to shift.

Baojun Zhang *et al.* proposed Addressing Filtering Wu Manber (AFWM)[18] Algorithm, in which pre-fix table is built for multiple patterns with pointer values. The pre-fix table values are sorted in ascending order and a hash table is prepared with them. In searching the entry from hash table is read, the pattern and the text are fixed then the right most character of the pattern is compared with the corresponding character of the text, if complete match/mismatch occurs then the other entries from hash tables are read.

Yoon-Ho *et al.* proposed B-LAyered bad character Shift Table (BLAST)[19]string matching algorithm. It has preprocessing and searching phases. In preprocessing bad character shift table is build that consists of B(length of the search window)-layered characters. In searching, the entry from shift table is read, the pattern and the text are fixed then the right most character of the pattern is compared with the corresponding character of the text, if complete match/mismatch occurs then the other entries from shift tables is read.

2. MULTI PATTERN MULTI-PROCESSOR PARALLEL STRING MATCHING ALGORITHM

The multi pattern multi-processor parallel string matching algorithm reads the directory and pattern set. Reads one file from the directory, opens the file, reads the file line by line, appends the line to string buffer. Reads one

pattern from the multi patterns, builds the group with right most character of the pattern. Reads the string from the string buffer, splits it into multiple parts based on the overlapping principle (overlapping principle is sizeof(string buffer)/p+m-1 where p is number of processors and m is length of the pattern). Splitting is done basing on the available processors in the interconnected computers.

On completion of reading, grouping and splitting processes the search process is called for searching the patterns in text. The same reading, splitting and searching processes are continued for all files in the directory. This algorithm searches the multi patterns in different directions in multiple parts of the text concurrently. The actual process is shown in algorithm 1.

Algorithm 1: Multi pattern Multi-processors parallel string matching algorithm

Input : *Directory that contain files (T) and pattern_set (P_1 , P_2, …)*

Output : *The number of occurrence and the positions of each pattern*

/* Initialization */

pattern_set={P_1,P_2, …}, n ← T.length, m1 ← P_1.length, m2 ← P_2.length,…, i ← 0, i1 ← 0, j ← m1-1, T= “”, count←–0, match_position←–0, shift_value←–0;

```
/* main function */
1       while pattern_set != NULL do
2       begin
3               if P1[m1-1]==P2[m2-1] then
4               pattern_set1={P1,P2}
5               else if P1[m1-1]==P3[m2-1] then
6               pattern_set1={P1,P3}
7               else if …..
8               end if
9       end while
10      for File F : Directory do
11      begin
12              while T = F.ReadLine() != NULL
13              T.append(T);
14              parts = T/p+m-1
15      for i2←0 to number of parts do
16      begin
17              search(part, pattern_set, match_position,count)
18      end for;
19      end for;
```

```
/* search function */
20      for i ←m-1 to part.length do
21      begin
22      for i1←0 to pattern_set.length() do
23      begin
24              while T[i]!=P_i1[m1-1]
25              i++;
26              shift_value=i;
27              while j >=0 AND T[shift_value] == P_i1[j]
28              begin
29                      j—;
```

```
30                          shift_value—;
31                    end while
32        if j == -1 then
33        begin
34           match_postion= i-(m1-2);
35           count=count+1;
36        return match_position& count
37        end if;
38        end for;
39        end for;
```

3. CASE STUDY

For testing the proposed algorithms, we will take all the chromosomes (23 pairs) of gorilla gorilla sequence from NCBI website in FASTA(Fast All) format[20]. The gorilla gorilla chromosomes contain 10 complex DNA index strictures (CODI)[21] namely TAGA, AGAA, GATA, TCTA, TCAT, GAAT, AGAT,CTTT, TATC, TCTG. These 10 CODIs are considered as search patterns.

To assess the efficiency of the proposed algorithms, we consider gorilla gorilla chromosome sequence of size 2.87Gb and also take the existing algorithms WM, QWM, QMM, HCWM, AFWM string matching algorithms and implement them. Table 1 shows the execution results of existing and multi pattern multi-processor parallel string matching algorithms. From these results we draw the graphs which are shown in Fig 1,2 and 3.

Table I
Search times of multi pattern multi-processor parallel string matching algorithm and existing algorithms (WM, QWM, QMM, HCWM, AFWM)

	Single Processor	*Two Processor*	*Four Processor*
WM	966219	473215	240562
QWM	943419	472317	236718
QMM	922314	470453	308761
HCWM	911321	456678	308127
AFWM	891254	445989	297991
Multi pattern Multi-processor	442386	223000	147900

From the fig 3, it is observed that, multi pattern multi processors string matching algorithm takes 442386 milli seconds to search 2.87Gb of data for all ten CODIs, it is two times less compared to existing WM,QWM, QMM, HCWM, AFWM string matching algorithms.

Similar performance from the proposed algorithm is observed from the figs 2 and 3. From the Fig 1,2, and 3, it is also observed that four processors take four times less time than single processor, two processors take two times less time than single processor and so on. From these observations, we conclude that multi pattern multi-processor parallel string matching algorithm can perform very well when compared with existing WM, QWM, QMM, HCWM, AFWM string matching algorithms.

4. CONCLUSIONS

In this paper, we have presented multi pattern multi-processor parallel string matching algorithm involving multi way search with while shift. The proposed parallel string matching algorithm is able to count each occurrence of the patterns. For measuring efficiency of our algorithm, we have conducted experiments by taking gorilla gorilla genome sequence as the data set. We conclude from the results that multi pattern four(multi) processors string

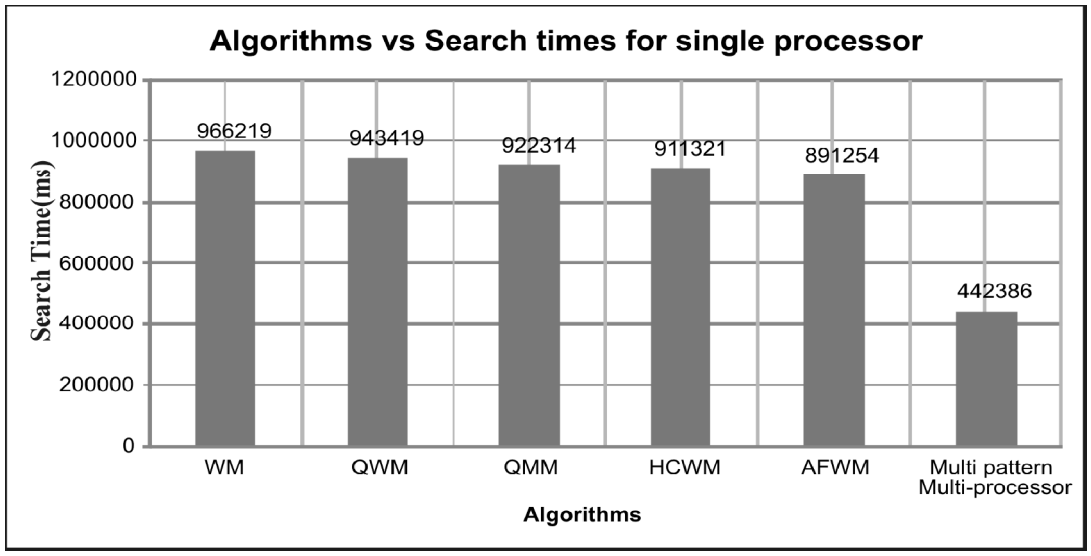

Figure 1: Search times of different algorithms for single processor

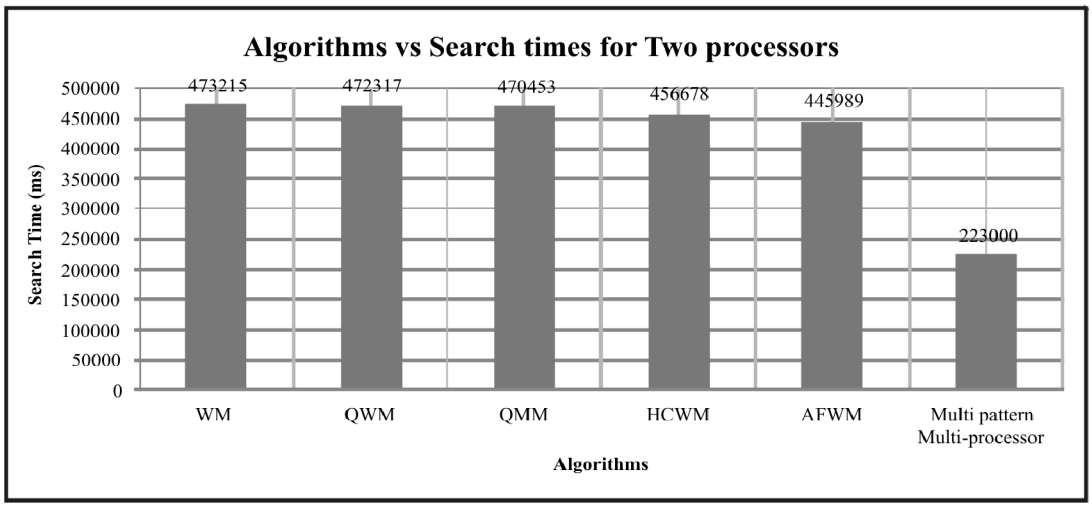

Figure 2: Search times of different algorithms for two processors

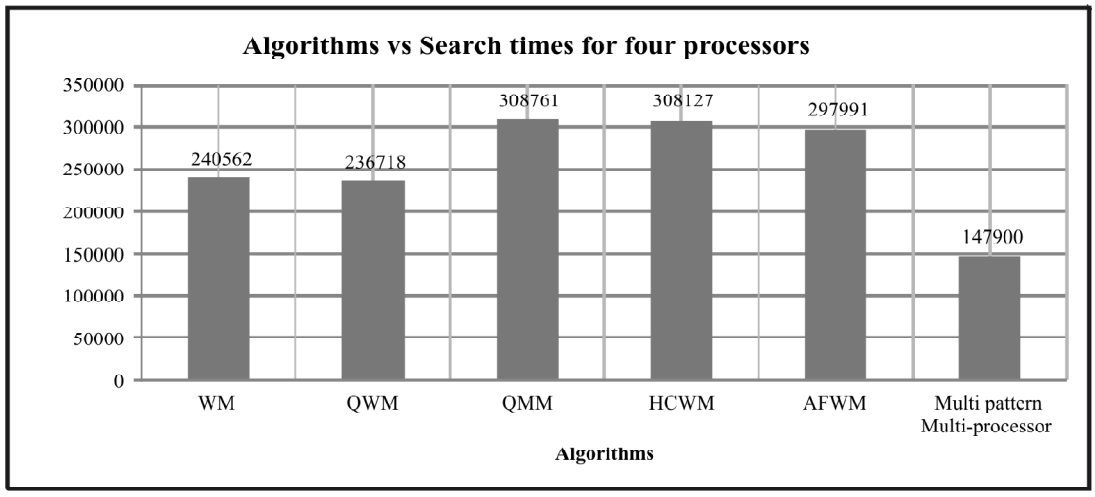

Figure 3: Search times of different algorithms for four processors

matching algorithm reduces the search time by a factor of 6when compared with the WM pattern string matching algorithm running with a single processor.

REFERENCES

[1] Aho, Alfred V., and John E. Hopcroft., "Design & Analysis of Computer Algorithms", Pearson Education India, 1974.

[2] Chinta Someswara Rao, Dr S Viswanadha Raju, "Next Generation Sequencing (NGS) Database for Tandem Repeats with Multiple Pattern 2^0-shaft Multicore String Matching", Genomics Data, Elsevier, Vol.7, 2016, PP.307–317, ISSN:2213-5960.

[3] Chinta Someswara rao, S Viswanadha Raju, "A Novel Multi Pattern String Matching Algorithm with While Shift", Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, ACM, pp.1-5, 2016.

[4] Chinta Someswara Rao, Dr S Viswanadha Raju, "Concurrent Information Retrieval System (IRS) for large volume of data with multiple pattern multiple (2^N) shaft parallel string matching", Annals of Data Science, Springer, Vol.3, Issue.2, 2016, PP.175-203, ISSN: 2198-5804.

[5] Chinta Someswara Rao, Dr S Viswanadha Raju, "A Frame Work for XML Ontology to STEP-PDM from Express Entities: A String Matching Approach", Annals of Data Science, Springer, Vol.3, Issue.4, 2016, PP.469-507, ISSN: 2198-5804.

[6] Chinta Someswara Rao, Dr S Viswanadha Raju, "Recent Advancements in Parallel Algorithms for string matching on computing models-a Survey and Experimental Results", ADCONS, Proceedings in LNCS Springer, 2012, pp. 270-278,ISBN: 978-3-642-29280-4.

[7] Chinta Someswara Rao, Dr S Viswanadha Raju, "Parallel String Matching with Multi Core Processors-A Comparative Study for Gene Sequences", Global Journal of Computer Science and Technology, Vol.13, Issue.1, 2013, PP.27-41, ISSN: 0975-4172.

[8] D. E. Knuth, J. H. Morris and V. R. Pratt, "Fast Pattern Matching in Strings", SIAM Journal on Computer, vol. 6, no. 2, pp. 323-350, 1977.

[9] R. S. Boyer and J. S. Moore, "A fast string searching algorithm", Communications of the ACM, vol. 20, no. 10, pp. 762-772, 1977.

[10] V. Ahoa and J. Corasickm, "Efficient string matching: an aid to bibliographic search", Communications of the ACM, vol. 18, no. 6, pp. 333-340.1975.

[11] S. Wu and U. Manber, "A fast algorithm for multi-pattern searching", TR 94-1 7. Tucson, AZ: Department of Computer Science, 1994.

[12] R. M. Karp and M. O. Rabin, "Efficient randomized pattern-matching algorithms," In: (2nd ed.), Tech. Rept. 31-81, Aiken Computer Lab, Harvard University, 1981.

[13] Yang and Shunli Ding, "An Improved Pattern Matching Algorithm Based on BMHS", In the proc. Of 11th International Symposium on Distributed Computing and Applications to Business, Engineering & Science, 2012.

[14] Lecroq, Thierry, "Experimental results on string matching algorithms", Software: Practice and Experience, pp.727-765,1995

[15] Chen Zhen and Wu Di, "Improving Wu-Manber: A Multi-pattern Matching Algorithm", In the proc. of 2008 IEEE International Conference on Networking, Sensing and control (ICNSC), pp. 812 –817, 2008.

[16] Liuling Dai, "An aggressive algorithm for multiple string matching" Information Processing Letters, Volume 109, pp. 553–559, 2009.

[17] Baojun Zhang, Xiaoping Chen, Xuezeng Pan, and Zhaohui Wu "Highconcurrence Wu-Manber Multiple Patterns Matching Algorithm", Proceedings of the International Symposium on Information Proces, pp. 404, 2009.

[18] Baojun Zhang , XiaoPing Chen , Lingdi Ping , Wu, Zhaohui, "Address Filtering Based Wu-Manber Multiple Patterns Matching Algorithm", International Workshop on Computer Science and Engineering, Qingdao, Vol.1, pp. 408 – 412, 2009.

[19] Yoon-Ho, Seung-Woo, "BLAST: B-LAyered bad-character SHIFT tables for high-speed pattern matching", Journal of Information Security, Institution of Engineering and Technology (IET), Volume 7, pp.195-202, 2013.

[20] *http://www.ncbi.nlm.nih.gov*

[21] Norrgard, Karen, "Forensics, DNA fingerprinting, and CODIS", Nature Education , no. 1, 2008.