Multithreaded Implementation of Hybrid String Matching Algorithm

Akhtar Rasool, Dr. Nilay Khare, Himanshu Arora ,
Amit Varshney , Gaurav Kumar
Maulana Azad National Institute of Technology
Bhopal (M.P) – 462051, India
akki262@yahoo.co.in
nilay_khare@yahoo.co.in
himanshuarora.manit@gmail.com
amit.varshney009@gmail.com
gauravsachan90@gmail.com

Abstract

Reading and taking reference from many books and articles, and then analyzing the Navies algorithm, Boyer Moore algorithm and Knuth Morris Pratt (KMP) algorithm and a variety of improved algorithms, summarizes various advantages and disadvantages of the pattern matching algorithms. And on this basis, a new algorithm – Multithreaded Hybrid algorithm is introduced. The algorithm refers to Boyer Moore algorithm, KMP algorithm and the thinking of improved algorithms. Utilize the last character of the string, the next character and the method to compare from side to side, and then advance a new hybrid pattern matching algorithm. And it adjusted the comparison direction and the order of the comparison to make the maximum moving distance of each time to reduce the pattern matching time. The algorithm reduces the comparison number and greatly reduces the moving number of the pattern and improves the matching efficiency. Multithreaded implementation of hybrid, pattern matching algorithm performs the parallel string searching on different text data by executing a number of threads simultaneously. This approach is advantageous from all other string-pattern matching algorithm in terms of time complexity. This again improves the overall string matching efficiency.

1. Introduction

The Knuth-Morris-Pratt string searching algorithm was conceived by Donald Knuth, Vaughan Pratt and James H. Morris in 1977. The KMP matching algorithm uses degenerating property of the pattern and improves worst case time complexity. Whenever we detect a mismatch, we already know some of the characters in the text (since they matched the pattern characters prior to the mismatch). We take advantage of this information to avoid matching the characters that we know will anyway match. [1,3]

The Boyer-Moore string searching algorithm was developed by Bob Boyer and J Strother Moore in 1977. The algorithm pre-process the pattern that is being searched for, it doesn't need to check every character of the string to be searched but rather skips over some of them. Its efficiency derives from the fact that with each unsuccessful attempt to find a match between the search string and the text it's searching, it use the information gained from that attempt to rule out as many positions of the text as possible where string cannot match. [2,4,5]

Hybrid pattern matching algorithm came in existence after combining KMP & Boyer-Moore string searching algorithms to generate a new algorithm. It also a pattern searching algorithm that searches a pattern from left to right in the string. Reducing the time requirement in the worst/average case it an effort to reduce processing time, the goal is to combine the best/average case advantages of algorithm with the worst case guarantees of KMP. According to the experiments we have conducted, new algorithm is among the fastest in practice for the computation of all occurrences of a pattern $p = p[1..m]$ in a text string $s = s[1..n]$ on an alphabet of size n giving a time complexity of $O(m + n)$.[6,7]

2. Multithreaded Hybrid String-Pattern Matching

Multithreading as a widespread programming an execution model allows multiple threads to exist within the context of a single process. These threads share the process' resources but are able to execute independently. The threaded programming model provides developers with a useful abstraction of concurrent execution. Thus it allows to operate faster on computer systems that have multiple CPUs, CPUs with multiple cores, or across a cluster of machines — because the threads of the program naturally lend themselves to truly concurrent execution.[8]

Benefiting From Multithreading:

Multithreading your code can help in the following areas:

- Improving Application Responsiveness

- Using Multiprocessors Efficiently
- Improving Program Structure
- Using Fewer System Resources

KMP algorithm which is the advanced and modified algorithm among the category of all such types of existing algorithm like Naive searching & Boyer-Moore algorithm but still we want some modification on it. As we desire less time for searching any pattern in the string. This algorithm gives better performance in the smaller as well as in the larger size of strings so it can be used where we need better performance and less time for searching any pattern, but to search a pattern in a very large size string it creates some problem in terms of Time Complexity. This problem is much more in this modern era , as space is not the big deal but time matters a lot here while execution of the pattern in the string.

We would like to introduce the concept in which a very large size string will be divided in parts depending upon the pattern size. The same pattern will be executed in the parts of strings in parallel that will help a lot to reduce the time complexity of the algorithm. As in terms of memory and processors we are much reliable multiple executions can be done simultaneously. The same concept is applied here the KMP matcher for matching the pattern in the strings which are divided in multiple parts and are executed in parallel. Additional one thing which comes in this concept is the case when a part of pattern arises in one string and another part of pattern arises in the consecutive string then we have to make sure about that the pattern matches or not. If the pattern size is "m" then we will store the last (m-1) elements of string1 and the first (m-1) elements of consecutive string2. So total of 2(m-1) elements will be stored in a new derived string. Now the pattern of size "m" will match its elements in this new derived string. It then shows number of attempts and the occurrence of pattern if exist in the same way i.e. KMP matcher algorithm. This process will be done for all joining parts of the strings. Now we will combine all the 4 execution result together. The major advantage of applying parallel processing in KMP Algorithm is its excellent relative time complexity, over the smaller size string.

3. Multithreaded Hybrid String Matching Algorithm

Given a String S and Pattern P of size m and n respectively.

Step1: We have given a String S of size m , break that string into two parts (i.e. S1 and S2), as shown in figure 1.

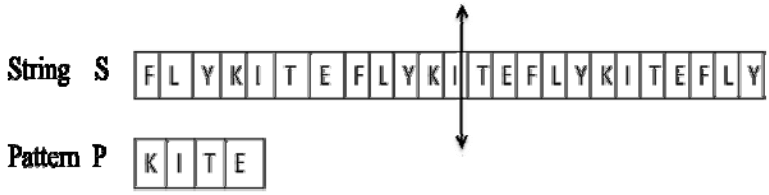

Figure 1: Division part description

Step2: Merge the characters of length (n-1) from the end of the one string and (n-1) characters from beginning of consecutive string.

Step3: Perform string pattern searching in the three strings extracted (S1 , S2 and S3) simultaneously, using Multithreaded Hybrid Algorithm , as shown in figure 2.

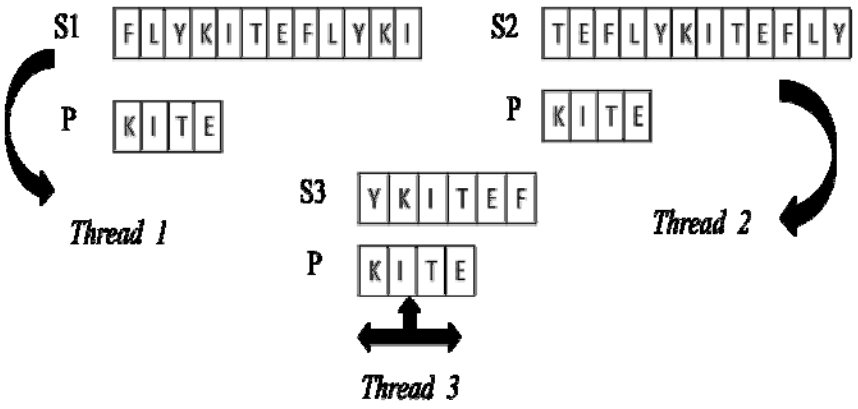

Figure 2 : Division process

Step4: If pattern found, display them else terminate the threads of a program and exit.

4. Results and Evaluation

The algorithm was tested with all four types of String-Pattern matching and four different pattern lengths (in Mbs). The results of String-Pattern Matching evaluated are shown in figure 3.The evaluation was done by taking constant number of threads i.e. 4 in case of Multithreaded Hybrid algorithm. We have implemented the program using a personal computer with 2 GHz Intel processor, and 3 GB of RAM.

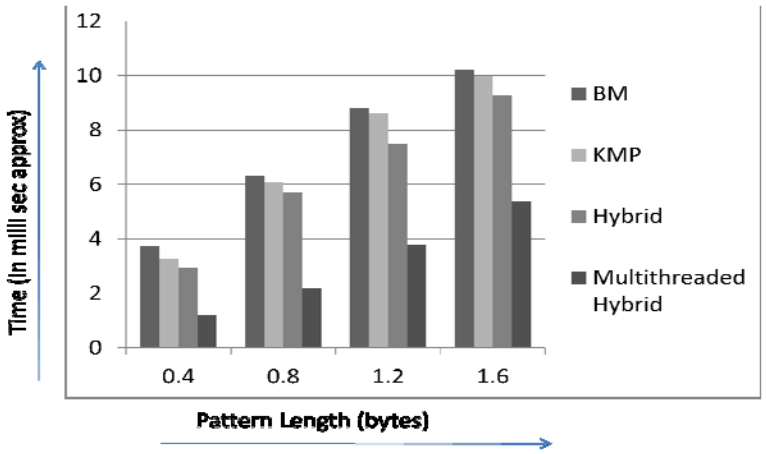

Figure 3: Comparison graph

Results of Multithreaded Hybrid Algorithm

The results of character comparison are collected by comparing characters in the pattern and in the text as shown in figure 7. On increasing the number of threads the performance increases, but upto a limit and beyond the limit it starts degrading.

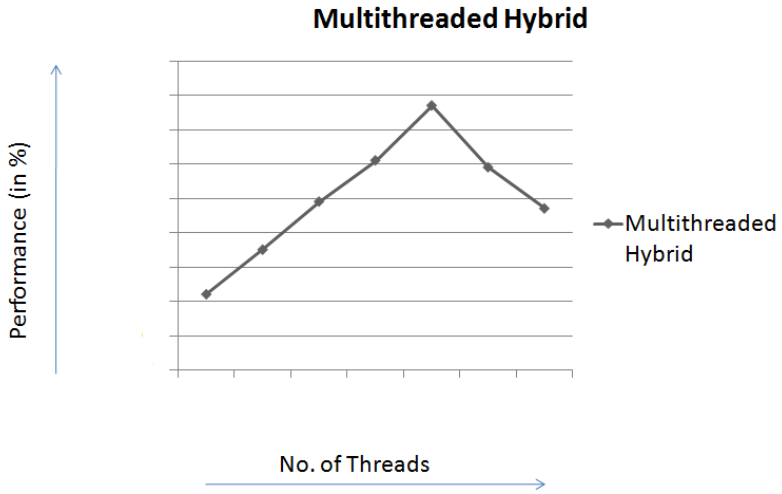

Figure 4: Performance in increasing threads

In the above figure it is clearly visible that the performance increases with the increase in number of threads but upto a certain limit and beyond that limit the performance starts degrading. This test was conducted using a personal computer with 2 GHz Intel processor, and 3 GB of RAM.

5. Conclusion

The results of the above comparison show that the Hybrid algorithm greatly improves the matching efficiency as shown in figure 5. The main drawback of the Boyer-Moore type algorithms is the pre-processing time and the space required, which depends on the alphabet size and/or the pattern size. For this reason, if the pattern is small (1 to 3 characters long) it is better to use the naive algorithm. If the pattern size is large, then the Knuth-Morris-Pratt algorithm is a good choice. In all the other cases, in particular for long texts, the Boyer-Moore algorithm is better. Finally, the Hybrid Algorithm which is the combination of the Boyer-Moore algorithm and Knuth-

Morris-Pratt algorithm is the best algorithm, according to execution time, for almost all pattern lengths. The most important characteristic of the Hybrid Algorithm is better use of end-term character and the next bit characters to get the maximum moving distance, as it starts comparing from the right side and then compares the left side of the string. This approach is efficiently improving the time complexity as it is based of multithreading technique. The use of pattern matching is very broad and efficient pattern matching algorithm can improve system performance. Multithreaded implementation improves the cpu utilization and increase the time efficiency.

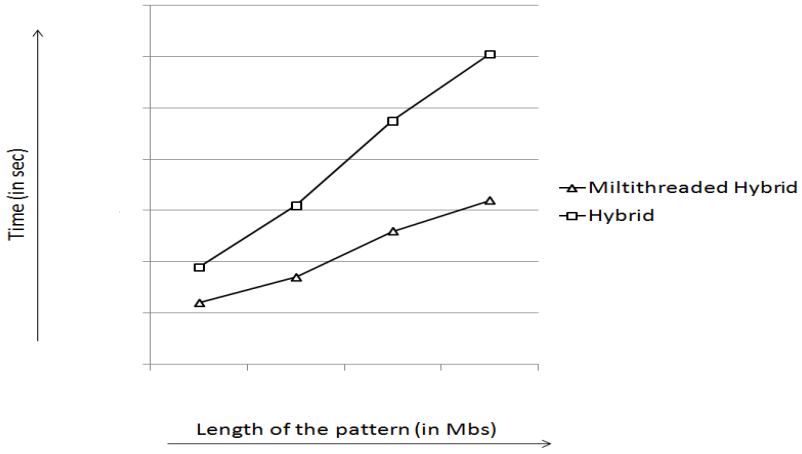

Figure 5: Time comparison graph

6. Future Work

Multithreaded hybrid string matching algorithm can be port on multi-core architecture and the parallel architecture to improve the string matching efficiency performance. The above algorithm can also be implemented on various parallel computation architectures like SIMD (Single Instruction Multiple Data) and MIMD (Multiple Instruction Multiple Data).

References

[1] String Matching: Knuth-Morris-Pratt Algorithm , Greg Plaxton Theory in Programming Practice, Spring 2004, Department of Computer Science, University of Texas at Austin.
[2] R. S. Boyer and J. S. Moore, A fast string searching algorithm, Commun. Assoc. Comput. Mach., 20 (1977), pp. 762–772.
[3] A correctness proof of the Knuth-Morris-Pratt , string-matching algorithm, ASP May 6, 2009.
[4] String Matching: Boyer-Moore Algorithm, Greg Plaxton, Theory in Programming Practice Fall 2005 Department of Computer Science, University of Texas at Austin.
[5] A FAST Pattern Matching Algorithm , S. S. Sheik, Sumit K. Aggarwal, Anindya Poddar, N. Balakrishnan and K. Sekar , Bioinformatics Centre and Supercomputer Education and Research Centre, Indian Institute of Science, Bangalore 560 012, India Received June 18, 2003.
[6] Hou Xian-feng, Yan Yu-bao, XiaLu “Hybrid pattern-matching algorithm based on BM-KMP algorithm” 2010 3rd International Conference on Advanced Computer Theory and Engg.(1CACTE), 978-1-4244-6542-2© 2010 IEEE.
[7] William F. Smyth, ShuWang, Mao Yu: An Adaptive Hybrid Pattern-Matching Algorithm on Indeterminate Strings, pp. 95–107. Proceedings of PSC 2008, Jan Holub and Jan Z˘ d’a´rek (Eds.), ISBN 978-80-01-04145-1 c Czech Technical University in Prague, Czech Republic.
[8] Susan J. Eggers, Joel S. Emer, Henry M. Levy, Jack L. Lo, Rebecca L. Stamm, Dean M. Tullsen Simultaneous Multithreading: A Platform for Next-Generation Processors, IEEE Micro 0272-1732/97© 1997 IEEE.