PARALLEL ALGORITHM FOR ACCELERATING PATTERN MATCHING USING MPI

[1]Rahul Pise, [2]Manik Chavan
CSE Department, Walchand College of Engineering,
Sangli, Maharashtra
Email: [1]rahoolpise@gmail.com, [2]chavan_manik@yahoo.com

Abstract— Pattern matching is a commonly used operation in many applications including image processing, bioinformatics, computer and network security, among many others. Knuth-Morris-Pratt (KMP) algorithm is one of the well-known pattern matching techniques and it is intensively used in computer and network security. Parallel algorithms for accelerating pattern matching have attracted a lot of attention due to their cost effectiveness and enormous power for massive data parallel computing. The proposed work of pattern matching is done using MPI on Intrusion Detection System (IDS) like Snort and the extracted pattern from Snort. Considering the parallel and distributed architecture, we proposed a method on which we perform very efficient parallel algorithm for KMP algorithm. The result shows the performance of parallel algorithm is completely outplayed the sequential performance using MPI.

Index Terms— Message passing Interface (MPI), Graphics processing units (GPU), pattern matching, parallel algorithm.

I. INTRODUCTION

Network Intrusion Detection System (NIDS) is widely used to protect computer systems from network attacks such as malware, port scans, and denial-of-service attacks. The most difficult part of NIDS is to inspect the packet content against the thousands of patterns. Because of that the increasing number of attacks, traditional sequential pattern matching algorithms is inadequate to meet the throughput requirements for high speed networks.

There were many approaches have been proposed to accelerate the pattern matching. The hardware approaches is classified into logic architecture [1], [2], [3], [4] In this architecture attack patterns are synthesized into logic circuits which are typically implemented on Field Programmable Gate Array (FPGA) to match more than one pattern in parallel. In comparison, memory architectures [5], [6], compile attack patterns into a state machine and perform pattern matching.

Recently, GPUs has been accepted to accelerate pattern matching because of their enormous power for massive data parallel computing. Wu-Manber-like multiple-pattern matching algorithm on GPUs achieved speedup two times as fast as the modified Wu-Manber algorithm used in Snort system.

The proposed work of pattern matching is done using MPI on IDS like Snort and the extracted pattern from Snort.

II. LITERATURE SURVEY

GPUs have been adopted to accelerate pattern matching because of their enormous power for massive data parallel computing. Pattern matching is widely used in various applications such as in Network intrusion Detection System (NIDS), cancer pattern, natural language processing, spam filter, word processor etc.

A. Naive Brute Force

It is one of the simplest algorithms which having complexity O (m n). In this, the First character of a pattern P with length m is aligned with the first character of text T with length n and then scanning is done from left to right after that shifting is done at each step it gives less efficiency.

B. Boyer Moore Algorithm

It performs larger shift-increment whenever there is a mismatch. It differs from Naive in the

way of scanning. It scans the string from right to left; unlike Naive approach that is P is aligned with T such that the last character of P will be matched to first character of T. If character is matched, then the pointer is shifted to left to very rest of the characters of the pattern. If a mismatch is detected at say character c, in T which is not in P, then P is shifted right to m positions and P is aligned to the next character after c. If c is part of P, then P is shifted right, so that c is aligned with the rightmost occurrence of c in P. The worst complexity is still O (m + n).

C. Knuth-Morris-Pratt (KMP)

This algorithm is based on automaton theory. Firstly a finite state automata model M is being created for the given pattern.

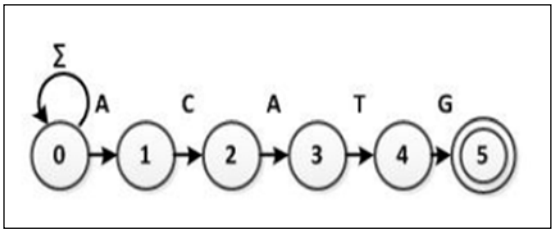

Fig.1 NFA for ACATG

The input string T with ∑ = {A, C, T, G} is processed through the model. If pattern is presented in the text, the text is accepted otherwise rejected. The following figure is a NFA model, created in ACATG string pattern.

The only drawback of the KMP algorithm is that it doesn't tell the number of occurrences of the pattern. The following Figure is the equivalent DFA of the above NFA.

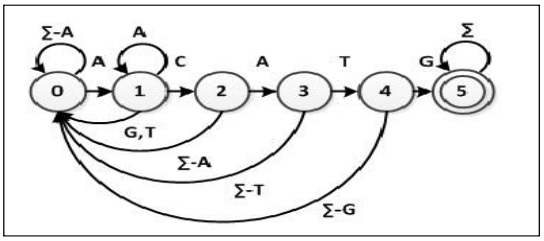

Fig. 2 Deterministic Finite Automata for ACATG

III. MESSAGE PASSING INTERFACE (MPI)

The aim of the MPI is to establish an efficient and flexible standard for message passing which is used for writing message passing programs. As such, MPI is the vendor independent and having message passing library. The benefit of developing message passing software using MPI helps to achieve design goals like portability, efficiency and flexibility. MPI performs their operations by using the methods such as MPI_send, MPI_receive, MPI_COMM_WORLD, MPI_Init and many more by Using these kind of functionality the processes communicate with themselves.

IV. PROPOSED WORK

The proposed work is dealing with the implementation of an efficient parallel algorithm for the pattern matching KMP algorithm [7]. The Naive pattern searching algorithm doesn't work well in the cases where we see matching characters followed by a mismatch.

The KMP matching algorithm uses degenerating property i.e. pattern having common sub-patterns appearing more than once in the pattern and it improves the worst case complexity to O (n). The idea behind KMP's algorithm is after some matches whenever we detect a mismatch, we already know that few of the characters in the text since they matched the patterns before the mismatched. Then by using this information we prevent matching characters that would be anyway going to match.

This algorithm usually does preprocessing over the pattern pat [] and it constructs an auxiliary array lps[] of size m which is the same as the size of the pattern. In this the lps stands for longest proper prefix which is also suffixed. For each sub-pattern pat[0..i] where i = 0 to m-1, lps[i] stores the length of the maximum matching proper prefix which is a suffix of the sub-pattern pat[0..i].

If the matching pattern is long the computation time gets increases to perform the matching and we are matching the multiple patterns in this paper.

V. PARALLEL IMPLEMENTATION OF KMP ALGORITHM

The steps of parallel implementations using MPI are given below:

1) Read the input file of Snort rules, prepare the input stream of that file, obtain the file size, allocate memory to contain whole file, copy the file into the buffer and divide it across processes using MPI_Bcast and MPI_Barrier.
2) Prepare the input stream of pattern file and get the number of lines.
3) Divide the input file among the processors.
4) Call the KMP search function and compute the longest proper suffix and at last show the matching patterns as output.

VI. RESULTS

The most time consuming part of KMP search is LPS that is Longest Prefix Suffix. The following results show the

Compute time of this algorithm.

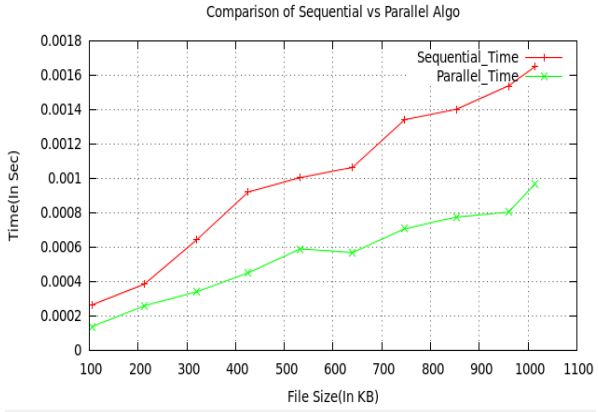

Fig.3 Sequential Vs Parallel Time Comparison graph

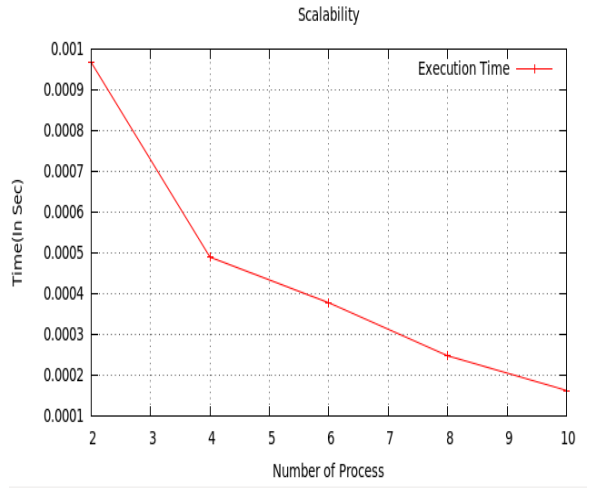

Fig.4 Scalability

The above graph in fig. 3 shows the comparison of sequential and parallel time in which we have taken the input file in KB on X-axis and Time(s) to compute in Y-axis by keeping the pattern file constant. So the above graph shows that the parallel computation time is more efficient than the sequential time and the performance of parallel is completely outplayed the sequential performance.

The graph above in fig. 4 shows the number of processes on X-axis and the time taken by that process on Y-axis. In the above graph we have taken the 10 processes and computed the time taken by them in seconds. It shows that if we go on increasing the number of processes, then the performance gets better. In this way the scalability is achieved.

VII. CONCLUSION

In this project we have implemented the Knuth-Morris-Pratt algorithm to match the Snort pattern and the performance of the parallel algorithm is far better than the sequential algorithm for multiple patterns.

The future work would be to use this technique using CUDA (Compute Unified Device Architecture).

REFERENCES

[1] Reetinder Sidhu and Viktor K Prasanna. Fast regular expression matching using fpgas. In Field-Programmable Custom Computing Machines, 2001. FCCM'01.The 9th Annual IEEE Symposium on, pages 227-238. IEEE, 2001.

[2] Christopher R Clark and David E Schimmel. Scalable pattern matching for high speed networks. In Field-Programmable Custom Computing Machines, 2004.FCCM 2004. 12th Annual IEEE Symposium on, pages 249-257. IEEE, 2004.

[3] Sarang Dharmapurikar and John Lockwood. Fast and scalable pattern matching for content _ltering. In Architecture for networking and communications systems, 2005. ANCS 2005. Symposium on, pages 183-192. IEEE, 2005.

[4] Alfred V Aho and Margaret J Corasick. E_cient string matching: an aid to bibliographic search. Communications of the ACM, 18(6):333-340, 1975.

[5] Nen-Fu Huang,Hsien-Wei Hung, Sheng-Hung Lai, Yen-Ming Chu, and Wen-Yen Tsai. A gpu-based multiple-pattern matching algorithm for network intrusion detection systems. Advanced Information Networking and Applications-Workshops, 2008. AINAW 2008. 22nd International Conference on, pages 62-67. IEEE, 2008.

[6] Christopher V Kopek, Errin W Fulp, and Patrick S Wheeler. Distributed data parallel techniques for content-matching intrusion detection systems. In Military Communications Conference, 2007. MILCOM 2007. IEEE, pages 1-7. IEEE, 2007.

[7] Knuth, Donald E., James H. Morris, Jr, and Vaughan R. Pratt. "Fast pattern matching in strings." SIAM journal on computing 6.2 (1977): 323-350.