Global Journals LATEX JournalKaleidoscope™
Artificial Intelligence formulated this projection for compatibility purposes from the original article published at Global Journals. However, this technology is currently in beta. *Therefore, kindly ignore odd layouts, missed formulae, text, tables, or figures.*

String Matching Problems with Parallel Approaches -An Evaluation for the Most Recent Studies

Chinta Someswara Rao[1]

[1] SRKR Engineering College affiliated to Andhra University

Received: 6 December 2012 Accepted: 5 January 2013 Published: 15 January 2013

Abstract

In recent years string matching plays a functional role in many application like information retrieval, gene analysis, pattern recognition, linguistics, bioinformatics etc. For understanding the functional requirements of string matching algorithms, we surveyed the real time parallel string matching patterns to handle the current trends. Primarily, in this paper, we focus on present developments of parallel string matching, and the central ideas of the algorithms and their complexities. We present the performance of the different algorithms and their effectiveness. Finally this analysis helps the researchers to develop the better techniques.

Index terms— text processing, irs, string matching, parallel algorithms.

1 INTRODUCTION

he problem of string matching has been studied from several decades. String matching problem is all about searching a given pattern of interesting length in a large text. The problem is very practical in its nature: it occurs in many real-worlds applications such as web search engines, linguistics, bioinformatics etc. This is the reason why algorithms should be efficient even if the speed and capacity of storage of computers increase regularly. String matching performs important tasks in many applications including information retrieval; library systems, artificial intelligence, pattern recognition, molecular biology, and text search and edit systems. The challenge is that for the string matching to be accurate, it needs to be able to search every byte of every input data streaming for a potential match from a large set of strings [1][2][3][4][5][6][7].

The main contributions of this work are summarized as follows. This work offers a comprehensive study as well as the results of typical parallel string matching algorithms at various aspects and their application on computing models. This work suggests the most efficient algorithmic models and demonstrates the performance gain for both synthetic and real data. The rest of this work is organized as, review typical algorithms, algorithmic models and finally conclude the study.

2 II.

3 OUR CONTRIBUTION

Thousands of papers, literally, have been published about string matching, exploring the multitude of theoretical and practical facets of this fascinating fundamental problem. For example let us consider text(T)length of n and pattern(P) length of m. suppose there is an occurrence of P in T, it means the text string ti,ti+1..ti+m-1 equal to P, so that H(ti,ti+1..ti+m-1,P)=0. Many other algorithms have been published; some are faster on the average, use only constant auxiliary space, operate in real-time, or have other interesting benefits. This work categorizes the algorithms into some categories to emphasize the data structure that drives the matching. These categories are discussed here. a) Intrusion Detection Systems (Ids) Yongin-si et.al [8] proposed an algorithm that maps target patterns onto parallel string matching architectures in intrusion detection systems(IDS). In this iterative pattern mapping, the sets of patterns that are mapped onto string matchers are stored in ascending order of the average pattern length in each turn. By mapping a set of patterns for a string matcher onto the string matchers repeatedly, the required number of string matchers is reduced. Therefore, the proposed iterative pattern mapping minimizes the total memory requirement for parallel string matching architecture.

4 i. DFA Based Approaches

Issues in accelerating DFA-based multi-pattern matching have received much attention in recent years by several researchers. Here we discuss some of them. Hongbin Lu et al., [9] propose a memory-efficient multiple-character-approaching architecture consisting of multiple parallel deterministic finite automata (DFAs), called TDP-DFA. By employing efficient representations for the transition rules in each DFA, TDP-DFA significantly reduces the complexity. They also present a novel scheme to share the storage of transition rules among multiple DFAs, substantially decreasing the total storage cost, and avoiding the cost increase being proportional to the number of DFAs. They evaluate this design through theoretical analysis and comprehensive experiments.1 (D D D D D D D D D)

Results show that TDP-DFA is able to meet the critical requirement of OC-768 wire speed processing, as well as constituting a promising way for scaling up to cope with throughput over 100 Gb/s in the future.

Experimental Results: Using the pattern set from Snort, they extract 2234 distinct substrings containing 33 793 characters from the signature database. In their prototype, the space for each state field in a CDLE entry is 2 bytes, allowing the maximum number of states up to 65 536. This is large enough considering the maximum number they measured in real cases is less than 6000. Similarly, the "Action ID" field in an entry of the associated RAM also occupies 2 bytes. It is shown in the Fig. 1. Yi-Hua E. Yang et al., [10] proposed a novel partitioning algorithm which converts an AC-DFA into a "head" and a "body" parts. The head part behaves as a traditional ACDFA that matches the pattern prefixes up to a predefined length; the body part extends any head match to the full pattern length in parallel body-tree traversals. Taking advantage of the SIMD instructions in modern x86-64 multi-core processors, they design compact and efficient data structures packing multi-path and multi-stride pattern segments in the body-tree. Compared with an optimized AC-DFA solution, their head-body matching (HBM) implementation achieves 1.2x to 3x throughput performance when the input match (attack) ratio varies from 2% to 32%, respectively. Their HBM data structure is over 20x smaller than a fullypopulated AC-DFA for both Snort and ClamAV dictionaries. The aggregated throughput of their HBM approach scales almost 7x with 8 threads to over 10 Gbps in a dual-socket quad-core Opteron (Shanghai) server. b) Parallel Processing based Approaches K.L. Chung et al., [11] presents an O(n) time parallel algorithm for finding all initial palindromes and periods of the string matching on an n × n reconfigurable mesh(RM) where n is the length of the string. They provide a partitionable strategy when the RM doesn't offer sufficient processers under the same strategy. This overcomes the hardware limitation and is very suitable for VLSI implementation.

Heikki Hyyro and Gonzalo Navarro **??** average-optimal for m ? w, assuming the alphabet size is constant. In practice, it performs better than the original ABNDM and is the fastest algorithm for several combinations of m, k and alphabet sizes that are useful, for example, in natural language searching and computational biology. To show that the concept of witnesses can be used in further scenarios, they also improve a recent variant of BPM. The use of witnesses greatly improves the running time of this algorithm too.

M. Oguzhan Külekci [13] proposed a new bitparallel algorithm, given name BLIM (bit-parallel length independent matching), and for exact pattern matching that does not restrict the input pattern to be shorter than the word size. The multiple pattern case is also addressed, and it is shown that up to computer word size number of patterns, whatever their lengths are, can be searched simultaneously in a single bit-parallel framework. Similar to other algorithms of this genre, BLIM is also capable of handling fixed-length gaps and character classes in the input strings as well. The proposed algorithm is compared with the other alternatives of its class, mainly the shift-or and BNDM variants. Experimental results indicate that BLIM is compatible with the previous bit-parallel algorithms with an additional gain of overcoming the word size new faster string matching algorithm, but a new approach identifying the use of bits in a different manner. Each bit in the proposed scheme represents an event, and the observations performed during the investigation alter these events according to the pre computed masks. In the exact pattern matching problem examined in this study, the events correspond to the alignments of the patterns in a window, and the observations are actually the characters accessed. Jorg Nolte And Paul Horton [14] discuss an experimental application that exploits TACO's distributed object groups and collective operations for computing the similarity between groups of molecular sequences, a computationally intensive core problem in molecular biology research. In particular they show how TACO's distributed collections can be conveniently combined with well known concepts found in the C++ standard template library (STL) to solve matching and sorting problems effectively on distributed hardware platforms. Figure **??** shows the results of the measurements using both a binary tree (par. red-2) and a 4-ary tree (par. red-4) topology. TACO's implementation is by all means in the competitive range and the reduction on the 4-ary tree topology even outperforms the MPI-based implementation. Kuo-Kun Tseng et al., [15] propose a new Parallel Automaton string matching approach and its hardware architecture for content filtering coprocessor. This new approach can improve the average matching time of the Parallel Automaton with Pre-Hashing and Root-Indexing techniques. The Pre-Hashing technique uses a hashing function to verify quickly the text against the partial patterns in the Automaton, and the Root-Indexing technique matches multiple bytes for the root state in one single matching. A popular Automaton algorithm, Aho-Corasick (AC) is chosen to be implemented by adding the two techniques; they employ these two techniques in a memory efficient version of AC namely Bitmap AC. For the average-case time, their approach improves

Bitmap AC by 494% and 224% speedup for URL and Virus patterns, respectively. Since Pre-Hashing and Root-Indexing techniques can be concurrently executed with Bitmap AC in the limitation. The main contribution of this study is not a or compute others fast. The resulting algorithm is hardware, their proposed approach has the same worstcase time as Bitmap AC.

Yunho Oh, Doohwan Oh and Won W. Ro [16] proposed a new parallel genome matching algorithm using graphics processing units (GPUs). Their proposed approach is based on the Aho-Corasick algorithm and it was developed based on a consideration of the architectural features of existing GPUs with a hundred or more cores. Thus, they provide an appropriate task partitioning method that runs on multiple threads and they fully utilize the cache memory and the shared memory structures available in GPUs. Especially, they propose a tiled access method for rapid data transfer from the global memory to the shared memory. They also provide new models for cache-friendly state transition table to improve performance of pattern matching operations on GPUs. The maximum throughput they achieved in various experiments was 15.3Gbps.

For the performance evaluation, they selected five genome sequences and used the EST database provided by the UC Santa Cruz Genome Browser as the pattern set. The details of the input sequences are described in Table **??**(a). In order to analyze the performance of modern GPU architectures, all experiments were performed using an NVIDIA GTX 285 (GT200 architecture) and a GTX 480 (Fermi architecture). J. J. ASTRAIN et al., [17] apply a genetic algorithm to adjust the automaton parameters for selecting the ones best fit to a particular application. They have introduced a genetic algorithm to adjust the parameters of a deformed fuzzy automaton. They propose the use of genetic algorithms for the automatic parameter tuning of a deformed fuzzy automaton and they validate it for the approximate string matching problem .This genetic approach overcomes the difficulty of using common optimizing techniques like gradient descent, due to the presence of non derivable functions in the calculus of the automaton transitions. Experimental results, obtained in a text recognition experience, validate the proposed methodology.

Yoginder S Dandass et al., [18] describes techniques for accelerating the performance of the string set matching problem with particular emphasis on applications in computational proteomics. The process of matching peptide sequences against a genome translated in six reading frames is part of a proteogenomic mapping pipeline that is used as a casestudy. The Aho-Corasick algorithm is adapted for execution in field programmable gate array (FPGA) devices in a manner that optimizes space and performance. In this approach, the traditional Aho-Corasick finite state machine (FSM) is split into smaller FSMs, operating in parallel, each of which matches up to 20 peptides in the input translated genome. Each of the smaller FSMs is further divided into five simpler FSMs such that each simple FSM operates on a single bit position in the input (five bits are sufficient for representing all amino acids and special symbols in protein sequences). This bit-split organization of the Aho-Corasick implementation enables efficient utilization of the limited random access memory (RAM) resources available in typical FPGAs. The use of onchip RAM as opposed to FPGA logic resources for FSM implementation also enables rapid reconfiguration of the FPGA without the place and routing delays associated with complex digital designs. Experimental results show storage efficiencies of over 80% for several data sets. Furthermore, the FPGA implementation executing at 100 MHz is nearly 20 times faster than an implementation of the traditional Aho Corasick algorithm executing on a 2.67 GHz workstation.

5 d) Finite Automata Based Approaches

Gerald Tripp [19] describes a finite state machine approach to string matching for an intrusion detection system. To obtain high performance, they typically need to be able to operate on input data that is several bytes wide. However, finite state machine designs become more complex when operating on large input data words, partly because of needing to match the starts and ends of a string that may occur part way through an input data word. Here they use finite state machines that each operates on only a single byte wide data input. They then provide a separate finite state machine for each byte wide data path from a multibyte wide input data word. By splitting the search strings into multiple interleaved substrings and by combining the outputs from the individual finite state machines in an appropriate way they can perform string matching in parallel across multiple finite state machines. A hardware design for a parallel string matching engine has been generated, built for implementation in a Xilinx Field Programmable Gate Array and tested by simulation. The design is capable of operating at a search rate of 4.7 Gbps with a 32-bit input word size.

Panagiotis D. Michailidis and Konstantinos G.Margaritis [20] proposed a linear processor architecture for flexible string matching. This architecture is a bit-parallel realization of the non-deterministic finite automation, which minimizes the amount of data flow between adjacent cells. Initially a bit-level algorithm is discussed which consists of two phases, i.e. preprocessing and searching. Then, starting from the data dependence graph of the searching phase processor array architecture is derived. Further, the preprocessing phase is also accommodated onto the same processor array design.

Junchen Jiang et al., [21] proposed a multistring matching acceleration scheme named Synergic Parallel Compact Finite Automata (SPC-FA) Matching System together with its conflict-free dispatching algorithm and the corresponding memory optimization

6 e) Prefix based Approaches

Abdelghani Bellaachia and Iehab Al Rassan [22] proposed a Tagged Sub-optimal code (TSC), a new coding technique to speed up string matching over compressed databases on personal digital assistants (PDA). TSC is a variable-length sub-optimal code that supports minimal prefix property. It always determines its codeword boundary without traversing a tree or lookup table. TSC technique may be beneficial in many types of applications: speeding up string matching over compressed text, and speeding decoding process. This paper also presents two algorithms for string matching over compressed text using TSC (SCTT) and the Byte Pair Encoding (BPE) technique (SCTB). Several experiments were conducted to compare the performance of TSC, Byte Pair Encoding (BPE), and Huffman code. Several PDA databases with different record sizes were used: the well-known Calgary dataset and a set of small-sized PDA databases. Experimental results show that SCTT is almost twice as fast as the Huffman-based algorithm. SCTT has also the same performance in search time as the search in uncompressed databases and is faster than the SCTB algorithm. For frequently updated PDA databases such as phone books, to-do list, and memos, SCTT is the recommended method regardless of the size of the average record length, since the time required to compress the updated records using BPE poses significant delays compared to TSC.

7 Experimental Results: The experimental results

for Searching over Compressed Text using BPE (SCTB) and Searching over Compressed Text using TSC (SCTT) solutions are presented, which was shown in the figure **??**. A library application was developed on a Palm OS handheld device. The application supports basic functions such as adding, deleting, or modifying an article entry, where each entry is a record consisting of author, title, and subject fields. Different sized database records and different record numbers in each database were loaded and implemented for testing purposes. Both searching techniques were implemented in the library application to allow searching while databases were in compressed form. C language with CodeWarrior compiler version 4.01 was the development environment used for designing and implementing the library application. Figure **??** shows the searching time using the SCTT, SCTB, and Huffman-based methods. Results show that SCTT is 88% faster than Huffman-based and 92% slower than SCTB-SO. Moreover, SCTT is 22% slower than the SCTB-Linear solution.

Figure **??** : Searching Time on PDA Figure 4 shows the searching time using the SCTT, SCTB, and Huffman-based methods compared to a linear search over uncompressed databases for small-sized records. Results show that SCTT is 85% faster than Huffman-based and 6% faster than SCTB-SO. In addition, SCTT is 15% faster than the SCTB-Linear solution for small-sized records. Leena Salmela, Jorma Tarhio and Petri Kalsi [23] proposed the improvements for FAAST algorithm, a variation of Boyer-Moore string matching problem for kmismatches. FAAST is specifically tuned for small alphabets. They further improve FAAST algorithm gaining speedups in both preprocessing and searching. They also present three variations of the algorithm for the k-difference problem. They show that the searching time of the algorithms is average-optimal and the preprocessing also has a lower time complexity than FAAST. Their experiments show that their algorithm for the k-mismatch problem is about 30% faster than FAAST and the new algorithms compare well against other state-of-the-art algorithms for approximate string matching.

Mihai Oltean [24] proposed a solution for finding a pattern P of length m in text T of length n. They describe a special device which can do string matching by performing n-m + 1 text-to-pattern comparisons. The proposed device uses light and optical filters for performing computations. Two physical implementations are proposed. One of them uses colored glass and the other one uses polarizing filters. They have made an in-depth analysis of the strengths and of the weaknesses of each method. At first sight they can infer that polarizing filters are more stable than colored glass for the string matching problem. The physical implementation of the proposed devices might be time consuming, so these methods might not bring such a great benefit unless they find some real-world cases where there are no other options for implementation but the ones they have proposed. However, the greatest benefit is that they have shown that string matching can be efficiently done by using the massive parallelism of the light.

Guang-Ming Tan et al., [25] proposed an attempt to design efficient multiple pattern searching algorithms on multi-core architectures. They observe an important feature which indicates that the multiple pattern matching time mainly depends on the number and minimal length of patterns. The multi-core algorithm proposed in this paper leverages this feature to decompose pattern set so that the parallel execution time is minimized. They formulate the problem as an optimal decomposition and scheduling of a pattern set, and then propose a heuristic algorithm, which takes advantage of dynamic programming and greedy algorithmic techniques, to solve the optimization problem. Experimental results suggest that their decomposition approach can increase the searching speed by more than 200% on a 4-core AMD Barcelona system.

Experimental Results: The input text and patterns are randomly generated. The length of the input text is 10 million bytes; the lengths of patterns follow a random distribution in a range **??**2; 200]. The number of patterns is set to be {10 000; 20 000; 40 000}. They examine the parallel algorithms on a commercial multicore processor, AMD Barcelona. It is a quad-core processor which features a highly integrated design with all four cores on a single die with shared resources. Each core has its own private 128KB L1 cache and 512KB L2 cache. All four cores share a common L3 cache that is at least 2MB in size. The full system provides an aggregate memory bandwidth of 21.4 GB/s and 54.4 GFlops/s peak performances. The compiler used in the experiment is GCC

4.1. Fig. **??** Figure **??** : Scalability of the parallel programs for different problem sizes, x-axis is the number of cores g) Other Approaches

Hung-Che Shen, and, Chungnan Lee [26] proposed a ”Whistle for Music” system which enables users to retrieve MIDI format music by whistling a melodic fragment. Three essential components are query processing, MIDI preprocessing and an approximate search engine. For query processing, they have achieved a real-time and robust whistle-to-MIDI converter. For feature extraction, the proposed MIDI preprocessing can extract individual, local and global melodic descriptions from MIDI files. In order to match query with target, they extend an existing search engine into a fast approximate melodic matching engine. Based on the integration of those three components, the system can return a list of MIDI files that are ranked by how closely they match the whistling. The systematic evaluation for the query-by-whistling system is finally performed. Although the content is focused on MIDI data, the unified algorithmic framework is suitable for a wide range of applications in music information retrieval. They have demonstrated three essential components: a melody transcription (a query processing), a MIDI preprocessing (feature extraction) and melodic AGREP (a search engine). One major feature in their system implementation is that ”Whistle for music” is fast enough for ”searching while whistling.” The other feature is that(D D D D D D D D D)

Year they provide detailed description of extracting multi-level melodic descriptions from a MIDI database. In order to provide a more sophisticated MIR system, they explore the query representation by individual, local and global descriptions. In addition, they have provided a helpful sight whistling tutor for derive a high-quality query. Finally, they have shown that the issue of scaling with database size can be studied by simulation. Given error distances between queries and targets, they can plot the expected number of queries whose correct targets will be ranked over a specific database size. The results show that careful measurement and objective comparisons can lead us to know the scaling trend about query and target. One encouraging aspect is that the performance can be predicted based on the evaluation methods.

HU Yue et al., [27] proposed a complete automaton and its high-speed construction algorithm for large-scale U-, V-, and U-V-uncertain multiple strings, including two or more uncertain strings interlaced with one another. The maximum number of parallel complete automation of the V-uncertain string is also given. This paper reveals that there are two kinds of pretermissions, i.e., similarly-connected and interlaced string pretermissions, and that mistake may appear in the matching of the regular expressions, or states in the automaton may increase in number, if the intersection of the U-uncertain strings sets and the homologous subsequent special point in the U-uncertain strings sets are not eliminated from the whole system. B. N. Araabi et al., [28] presents a syntactic/semantic string representation scheme as well as a string matching method as part of a computer-assisted system to identify dolphins from photographs of their dorsal fins. A low-level string representation is constructed from the curvature function of a dolphin’s fin trailing edge, consisting of positive and negative curvature primitives. A high-level string representation is then built over the low-level string via merging appropriate groupings of primitives in order to have a less sensitive representation to curvature fluctuations or noise. A family of syntactic/semantic distance measures between two strings is introduced. A composite distance measure is then defined and used as a dissimilarity measure for database search, highlighting both the syntax (structure or sequence) and semantic (attribute or feature) differences. The syntax consists of an ordered sequence of significant protrusions and intrusions on the edge, while the semantics consist of seven attributes extracted from the edge and its curvature function. The matching results are reported for a database of 624 images corresponding to 164 individual dolphins. The identification results indicate that the developed string matching method performs better than the previous matching methods including dorsal ratio, curvature, and curve matching. The developed computer-assisted system can help marine mammalogists in their identification of dolphins, since it allows them to examine only a handful of candidate images instead of the currently used manual searching of the entire database. The figure 7 describes the percentage of test images with first correct match VS number of database individuals examined before catching the correct match.

8 CONCLUSIONS

In this study, we widely investigate the problem of sequential and parallel approaches in the context of string matching. An outline of string corresponding is made, in which the special forms of parallel string matching problem are also distinguished, and the classifications of parallel string matching problem are discussed.We importantly review different classifications of parallel string matching algorithms. Based on this study, a number of positive suggestions are made which will cooperative to the researchers for developing better techniques. [1] [2]

[1]CString Matching Problems with Parallel Approaches -An Evaluation for the Most Recent Studies

[2]C comparison of the parallel searching time. The String Matching Problems with Parallel Approaches -An Evaluation for the Most Recent Studies

OPEN ASSOCIATION OF RESEARCH SOCIETY, USA

RESEARCH | DIVERSITY | ETHICS

1

Figure 1: Figure 1 :

[Lu et al. ()] 'A memory-efficient parallel string matching architecture for high-speed intrusion detection'. H Lu , K Zheng , B Liu , X Zhang , Y Liu . *IEEE Journal on* 2006. p. . (Selected Areas in Communications)

[Tseng et al. ()] 'A parallel automaton string matching with pre-hashing and root-indexing techniques for content filtering coprocessor'. Tseng , Ying-Dar Kuo-Kun , Tsern-Huei Lin , Yuan-Cheng Lee , Lai . *Architecture Processors, 2005. ASAP 2005. 16th IEEE International Conference on application-Specific Systems*, 2005. IEEE. p. .

[Araabi et al. ()] 'A String Matching Computer-Assisted System for Dolphin Photoidentification'. B N Araabi , N Kehtarnavaz , T Mckinney , G Hillman , B Wu Rsig . *Annals of Biomedical Engineering* 2000. 28 p. .

[Yoginder et al. ()] 'Accelerating String Set Matching in FPGA Hardware for'. S Yoginder , Dandass , C Shane , Mark Burgess , Susan M Lawrence , Bridges . *Bioinformatics Research* 2008. p. . (BMC Bioinformatics)

[Yongin-Si ()] *An iterative Pattern Mapping For parallel string matching Architecture In Intrusion Detection Systems*, Gyeonggi Yongin-Si . 2012. IEICE Electronics Express. 9 p. .

[Salmela et al. ()] *Approximate Boyer-Moore String Matching for Small Alphabets" Algorithmica*, Leena Salmela , Jorma Tarhio , Petri Kalsi . 2010. p. .

[Astrain et al. ()] 'Approximate String Matching Using Deformed Fuzzy Automata: A Learning Experience'. J J Astrain , J R Garitagoitia , J R Gonzalez De Mendivil , J Villadangos , F Farin , A . *Fuzzy Optimization and Decision Making*, 2004. p. .

[Michailidis and Margaritis ()] 'Bit-level processor array architecture for flexible string matching'. Panagiotis D Michailidis , Konstantinos G Margaritis . *proceedings of the 1st Balkan Conference in Informatics*, (the 1st Balkan Conference in Informatics) 2003. p. .

[Hyyro and Navarro ()] 'Bit-Parallel Witnesses and Their Applications to Approximate String Matching'. Heikki Hyyro , Gonzalo Navarro . *Algorithmica* 2004. p. .

[Külekci ()] 'BLIM: A New Bit-Parallel Pattern Matching Algorithm Overcoming Computer Word Size Limitation'. M Oguzhan Külekci . *Mathematics in Computer Science*, 2010. p. .

[Chung and Chen ()] K L Chung , Taipei , H N Chen . *Taoyaun" Parallel Finding All Palindromes and Periods of a String on Reconfigurable Meshes*, 1998. p. .

[Shen and Lee ()] *Content-based MIDI Music Retrieval and Computer-aided Composition Based on Musical Whistling*, Hung-Cche Shen , Chung-Nan Lee . 2006.

[Hu Yue et al. ()] 'Giant complete automaton for uncertain multiple string matching and its high speed construction algorithm'. Hu Yue , Gao Qingshi , Wang Li , Peifeng . Information Sciences 2011. p. .

[Oh et al. ()] 'GPU-Friendly Parallel Genome Matching with Tiled Access and Reduced State Transition Table'. Yunho Oh , Doohwan Oh , Won W Ro . *Int J Parallel Prog* 2013. p. .

[Yang et al. ()] 'Head-body partitioned string matching for deep packet inspection with scalable and attackresilient performance'. Y-He Yang , K Viktor , Chenqian Prasanna , Jiang . *Parallel & Distributed Processing*, 2010. IEEE. p. . (IEEE International Symposium on)

[Simon and Inayatullah ()] 'Improving Approximate Matching Capabilities for Meta Map Transfer Applications'. Y Simon , M Inayatullah . *Proceedings of Symposium on Principles and Practice of Programming in Java*, (Symposium on Principles and Practice of Programming in Java) 2004. p. .

[Oltean ()] 'Light-based string matching'. Mihai Oltean . *Nat Comput* 2009. p. .

[Someswararao et al. ()] *Parallel Algorithms for String Matching Problem based on Butterfly Model*, Chinta Someswararao , Butchiraju , Viswanadha Raju . July -Sept, ISSN 2229-4333, 2012. 3 p. . IJCST

[Nolte and Horton ()] 'Parallel Sequence Matching with TACO's Distributed Object Groups -A Case Study from Molecular Biology'. Jorg Nolte , Paul Horton . *Cluster Computing*, 2001. p. .

[Raju ()] 'parallel string matching algorithm using grid", "international journal of distributed and parallel systems'. S Raju . *ijdps* 2012. 3 (3) .

[Someswararao et al. ()] 'PDM data classification from STEP-an object oriented String matching approach'. Chinta Someswararao , Butchiraju , Viswanadha Raju . *IEEE conference on Application of Information and Communication Technologies*, 2011. p. .

[Someswararao et al. ()] 'Recent Advancement is Parallel Algorithms for String matching -A survey and experimental results'. Chinta Someswararao , Butchiraju , Viswanadha Raju . *IJAC* 2012. 4 (4) p. .

[Someswararao et al. ()] 'Recent Advancement is Parallel Algorithms for String matching on computing models -A survey and experimental results'. Chinta Someswararao , Butchiraju , Viswanadha Raju . *LNCS* 2011. Springer. p. .

[Someswararao et al. ()] 'Recent Advancement is String matching algorithms-A survey and experimental results'. Chinta Someswararao , Butchiraju , Viswanadha Raju . *IJCIS* 2013. 6 (3) p. .

[Tan et al. ()] 'Revisiting Multiple Pattern Matching Algorithms for Multi-Core Architecture'. Guang-Ming Tan , Ping Liu , Dong-Bo Bu , Yan-Bing Liu . *Journal of Computer Science and Technology* 2011. p. .

[Tripp ()] 'String Matching Engine for use in high speed network intrusion detection systems'. Gerald Tripp . *J Comput Virol* 2006. p. . (A parallel)

[Bellaachia And Iehab Al Rassan ()] 'String Matching Over Compressed Text on Handheld Devices Using Tagged Sub-Optimal Code (TSC)'. Abdelghani Bellaachia And Iehab Al Rassan . *Real-Time Systems*, 2005. p. .

[Jiang et al.] 'Synergic Parallel Compact Finite Automatons for Accelerating Multi-String Matching'. Junchen Jiang , Yi Tang , Bin Liu , Xiaofei Wang , Yang Xu . *Proceedings of the 5th ACM/IEEE Symposium on*, (the 5th ACM/IEEE Symposium on)