International Journal of Applied Engineering Research
ISSN 0973-4562 Volume 10, Number 6 (2015) pp. 15545-15559

http://www.ripublication.com

A Memory-Efficient In Scalable String Pattern Matching Method For Integrated Deterministic Finite Deep Packet Inspection

[1]Pon.Arivanantham, [2]Dr. M. Ramakrishnan
[1]Research Scholar, Sathyabama University, Chennai. & Associate Professor, Dhanalakshmi Srinivasan College of Engineering & Technology, Mamallapuram.
arivanantham.1972@gmail.com
[2]Professor & Head, Department of Information Technology, Vellammal Engineering College, Chennai
ramkrishod@gmail.com

Abstract

In this paper, we show that the use of traditional methods require prohibitive as scanning application memory first package you use. Examines each packet header and payload to determine the ability to attack signing makes the network intrusion detection system (NIDS), a promising way to protect Internet systems. Because most of the attack on the chain known or can be a plurality of sub-assemblies, and a corresponding number of critical factors, and the bottleneck in the manuscript to meet the growing demands for capacity. We propose an efficient memory architecture, a multi character by automatic and inevitable limitations of several parallel called TDP-DFA. DPI classical parallel algorithm for learning, and the establishment of a memory model for different methods in parallel. Based on this model, we found that the state machinery, which requires a minimum of space on the memory chip, and guide our performance DPI NP ideal organizational structure.

Keywords: Deep Packet Inspection, Network Processor, Pattern Matching, Parallel Processing, Computer network security.

Introduction

Failed traditional security devices such as network-based firewalls, packet filtering on the performance of packet headers only, to identify attacks that use capital is suspicious. Inspection by both packet headers and payloads to determine the signatures of the attack, Network Intrusion Detection System (NIDS) is able to detect whether the hackers / crackers are trying to break or launch denial of service (DOS) attack. Because most known attacks can be represented with chains or multiple subsets, and the corresponding series is one of the main ingredients in the paper.

Matching string in the manuscripts is computationally intensive in that, unlike the classifications simple package, manuscripts need to clear all of the headers and payloads of each incoming packet for thousands of sentences suspicious. Worse than that, a series of variable lengths. Manuscripts in popularity, such as snoring [1], and 70% of the total execution time and 80% of the instructions are for a series matching routine [2]. Another challenge for string matching mechanism exposed to attacks of the worst cases intended. Understanding and al-Qaeda, an attacker can easily reload matching operation, resulting in a series of worst-case input [2]. Thus, dealing successfully with the worst case scenarios of great importance to matching approach in a series of manuscripts. Multi-pattern matching on trodden requires a two-stage time-consuming, and two traffic compression and pattern matching. So, the most current safety tools either ignore the scan compressed traffic, which may be the cause of security vulnerabilities or disable the option for the movement of the compact.

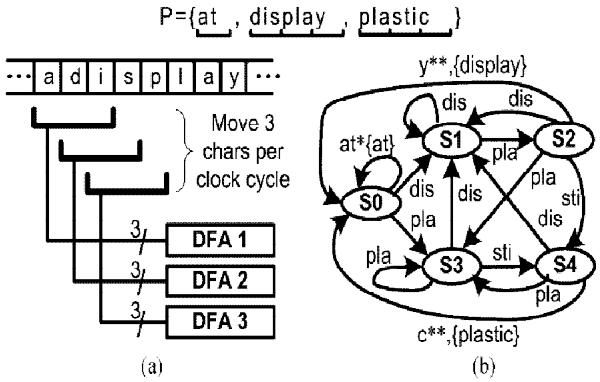

Figure 1: The multiple-character-approaching model

Survey Package Content (also known as 7-Layer scanning or filtering load) is crucial for the continued security applications and network monitoring. In these applications, the load on a particular set of data packets that matches the pattern of the traffic stream to determine the application of certain types of viruses and protocol definitions. Currently, the regular expression is a pattern through a series of explicit language that matches the model it replaces package selection application scanning. It intends to performance and flexibility to describe a useful model used widely to them. For example, in the book Linux Application Protocol (L7 filtering) [1], and that all of the protocol identifiers and regular expressions. Similarly, the development of snoring from 4867 [2] Intrusion Detection in April 2003, and the instruction set of any system of regular expression 1131 using regular expressions for the month of February 2006, another system, intrusion detection and the rules of brotherhood [3] and is also used as a pattern of regular expression. We use in the pattern of performance settings download scanning the real world through a detailed analysis proved the effectiveness of our re-writing and assembly solutions. As shown in these results, according to our implementation of an embodiment DFA- Linux can be based on the filtration system and L7 in snoring using the NFA regex increase 50-700 times the size of the speed game. Can be up to 12-42 times more common acceleration-based analyst DFA.

High-speed pattern matching pattern for certain groups can achieve gigabit rates. This is important for the implementation of a fast regular expression matching of load packet processors using a network or processors for general purposes, and the ability to classify more quickly and efficiently enables many new technologies such as real-time to detect the worm, content and search networks overlay, fine- load balancing granules, etc.

Literature Survey

While many based on a series of [5] program matching algorithm - [9] has been developed in the past few years, and is generally believed that it cannot achieve multi-gigabit throughput. Recently, it was suggested many of the hardware-based algorithms, which are based on many of the solutions on a Field Programmable Gate (FPGA) [10] - [17].

Because of the abundance of parallel programming in FPGA, researchers can instance a large number of parallel processing units, with each unit being in charge of one or several patterns, which greatly enhances productivity. However, these methods require re-compilation time consuming and reconfiguration of FPGA to any change of the rule set.

Dharmapurikar and others. Submitted two plans detection [24], [25] which can handle multiple character per clock cycle and achieve an average production capacity of up to multigigabit with moderate memory consumption. The proposed schemes are vulnerable to malicious attacks because, in the worst cases, they must often access to a relatively slow off-chip SRAMs to launch a series of comparisons exactly. Tripp also offer a solution approaching multi-character [26], which divides the entire DFA in many small DFAS, where all the DFA only looking for one or several signatures to reduce the growing complexity of DFA strongly resulting from the expansion of the input display. Solve the problem of alignment, but most of the signatures must be decomposed into multiple sub-overlap, leading to a positive relationship between the cost of storage and the number of characters processed per clock cycle. On the other hand, we have proposed TDP-DFA reduces the cost of storage for the use of all DFA without affecting the performance, as well as to maintain the sub-line public storage cost in productivity.

The network processors (NPS) components of the basic treatment for various routers and switches Internet programming because of the high processing capability and optimization package. Typical NP employs multiple processing elements of programming (PES), which can be organized in a manner parallel or pipelined. Is tuned in the instruction set architectures (ISAs) NPS for efficient manipulation package, such as finding a set-bit wide memory access [9]. These innovations make use of large-scale nuclear power sources in applications ranging from traditional IP routing and firewall to switch content aware and deep packet inspection, which involved complex processing. However, on the basis of DPI is one of the important issues of the NP in the design of the system, you need to deal with a large number of features and unit efficiency and memory resources. Especially in architecture NP public, on-chip memory has become a valuable resource to store all of the PE program

(instructions) machines DPI state (data), as is the case in [8]. The best use of on-chip memory space is critical to the performance NP design and application of DPI.

Methodology

In this paper, we strive to resolve the fast and efficient memory usage for regular expression matching to scan package payload. Define the scope of the problem as follows:

- We consider an approach based on the DFA in this paper, the approach based on NFA- ineffective treatments serial or parallel processors with limited (such as CPUs, compared with the multi-core chip (FPGA)). Our goal is to achieve O (1) calculate the cost involved for each character, which cannot be achieved by any solutions based on DFA, due to the excessive use their own memory. Therefore, the focus of the study is to reduce the memory overhead DFA, while close to the optimum processing speed of O(1) each of the characters.
- We focus on general-purpose processor-based architecture, and explore regular expressions in this environment restrictions match. Where appropriate, we take full advantage of multi-core processors has become a trend in these structures prevailed. However, our results in FPGA and ASIC-based methods can be used, and based on [20].
- It is worth mentioning that there are two sources for storing DFAS: states and transitions. With respect to the non-linear digital transformation in some countries, as each country can have up to 28 (ASCII all characters) links to the United States for next year. Thus, we see that some countries (minimized DFA) and the main factors that determine the memory usage of the rest of this article. Further, since a high level of performance, and we do not believe that the use of any compression technique DFAS table.

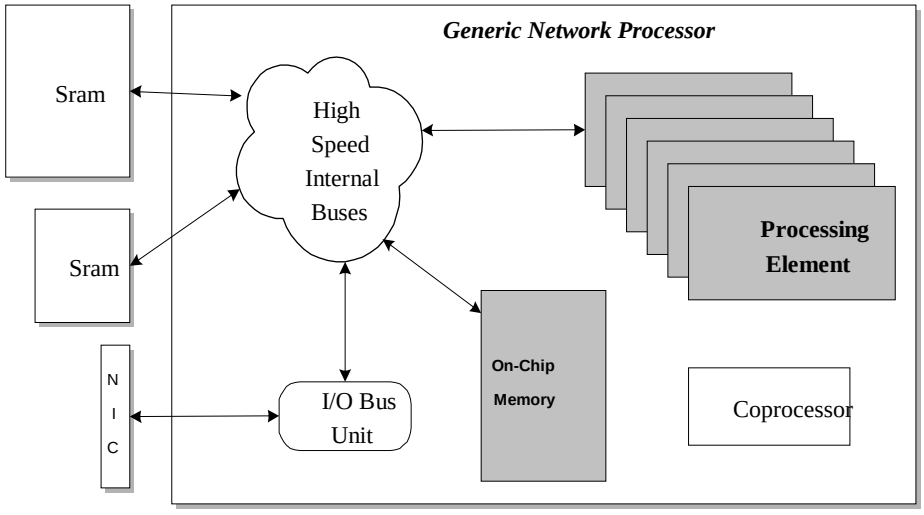

Figure 2: Diagram of a generic network processor.

Network processor typically contain multiple processing cores, processors help, memory components versatile, high-speed network I/O interfaces, as shown in Figure 1. Multiple processing cores, and are often indicators, programmable units for processing package. Treatments include assistance such as hardware accelerators in many of the NPS to create a wizard, and recording events or accelerate a specific task, such as 3DES. NP interfaces with memory modules of different size and speed, and specifically memory on-chip and off-chip SRAM and SDRAMs. The size is limited and therefore its use is critical to reduce the number of expensive off-chip memory accesses. Unit I/O bus controlled receiving and sending a package through a network interface cards (NIC).

When you are receiving packets by bus cards NIC, are stored in SDRAM and then processed by public institutions. For DPI, and processing involves reading bytes of payload package, compare it with the pattern of bytes in the state apparatus and the migration from one country to another. Once processed, then the packets are transferred through the I/O bus and the widening again. One of the design goals is to reduce the number to memory modules outside the chip so that it is bounded by the average memory access time to meet the rates of the line.

Matching of Individual Patterns

In this section, we offer our best solution to match the regular expression patterns of the individual. The main technical challenge is to create a DFAS that can fit in memory, which makes it fast approach based DFA- possible. First, we identified certain basic concepts of the next building DFA packet payload scanning situation. Although the theoretical analysis [12:14] show, DFAS ejected by the index, in this, we determine the specific structure that can lead to the rapid growth of DFAS. Based on this analysis, insights, proposed technical writing style, general Mode of transaction memory efficient matching (which real-world applications often allowed) possibilities. Finally, we offer guidelines for the style book about how to write patterns that can be implemented effectively.

Table 1: Analysis of patterns with *k* characters

Pattern features	**Example**	**# of states**
1. Explicit strings with *k* characters	^*ABCD* .**ABCD*	*k+1*
2. Wildcards	^*AB.*CD* .**AB.*CD*	*k+1*
3. Patterns with ^, a wildcard, and a length restriction *j*	^*AB.{j+}CD* ^*AB.{0, j}CD* ^*AB.{j}CD*	*O(k*j)*
4. Patterns with ^, a class of characters overlaps with the prefix, and a length restriction *j*	^*A+[A-Z]{j}*D	$O(k+j^2)$
5. Patterns with a length restriction *j*, where a wildcard or a class of characters overlaps with the prefix	.**AB.{j}CD* .**A*[*A-Z*]{*j*+}D	$O(k+2^j)$

Design Considerations

Although regular expressions and the theory of automated and can be applied directly on the package scanning load, and there is a clear difference between the two. Most existing studies focus on regular expressions on the particular type of assessment that it is, and check whether the fixed length string belongs to the language that defines the regular expression. More specifically, the series is said to be a fixed length in the language of a regular expression, if not matching the series from start to finish by the DFA in contrast to the regular expression. In contrast, survey data payload in the package, and all of the specific subset of input or regular expression pattern can match the input. Without prior knowledge of the beginning and ending positions of those sub, which was established to recognize the DFAS All games subset can be very complicated.

In order to understand better, offer next to some of the concepts related to the completion of the results of matching and execution model DFA subset of matching. Since the expression of a series of investment laws, as well as a full set of results contains all possible subsets of the input pattern possible matches. For example, a given input pattern ab *abbb, can report three possible match, ab, abb, and abbb. We call this type of matching matching comprehensive and known officially as follows:

Exhaustive Matching

Consideration in the matching process M as a function of the pattern P and S series to set the power S, so, M (P, S) = {subset S 'of S | S' and accepted by DFA of P}.

In practice, it is expensive and unnecessary in many cases, to submit a report on the sub-matching, where most of the applications that can be satisfied by a subset of those games. Therefore, we propose a new concept, matching non-overlapping, that relaxes the requirements of matching comprehensive.

Total Memory (KB), ***T***

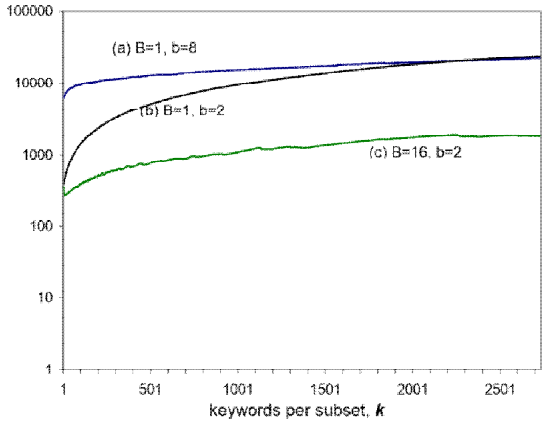

Figure 3: Total memory (*T*) usage of classical, bitlevel and bit-byte level AC state machines (*K* = 2733).

Non-overlapping Matching:
M considered in the matching process as a function of the pattern P and S series to a set of chains, specifically,

$M(P, S) = \{\text{substring } S_i \text{ of } S | \forall S_i, S_j \text{ accepted by the DFA of } P, S_i \cap S_j = \varphi\}$.

If the location you enter more than one, according to the pattern corresponding to the matching process for a subset of all non-overlapping patterns. The pattern AB * and abbb inputs, and three games overlap through the exchange of the prefix ab. For example, will report matching non-overlapping one game instead of three.

For most applications, download scanning, and we expect the interview to have enough non-overlapping, and these applications are mostly wondering if the attacks appeared application layer pattern or a certain group. In fact, most survey-based tools such as grep, kindness and systems such as Snort [2] and Bro [3] implementation of the special cases of non-overlapping matching, such as left-over longer matching or left-over shorter matching. We'll show later in this section, and the memory can be used to create DFA memory usage in higher efficiency of non-overlapping. In the following discussion, we focus on patterns without '^' attached at the beginning. It is noteworthy that for these patterns, there is no prior knowledge of whether / where it may appear substring matching.

Proposed Method

Find a set of matching sub (using either the exclusive or non-matching overlapping), needs to implement the DFA added that with repeated searches of Input: The initial search starts from the beginning of the entry, read the characters until (1) it has been reported that all the games (if was the use of a comprehensive matching) or a single game (if the use of matching non-overlapping), or (2) they reached the end of the entry. In the first case, the new research starts from the next character in the input (if the use of a comprehensive matching) or of the character after the game said (if the

use of matching non-interlaced). In the latter case, do not start a new search of the next character in the input. It uses this pattern of frequent scanning with the DFA generally in language parsers. However, it is operative to strong the package payload coincidental to load a package corresponding a particular outline low. The input is checked from start to finish, the DFA can recognize all the games could begin substring in different locations of the input. Search through the use of this method can achieve true O(1) cost is calculated for each character, and thus are suitable for network applications. Achieve high-speed scanning, we used this approach in the rest of the study. The first phase called the goto function builds trie of patterns, where the root of the trie represents the state where it was partially offset without restriction. Chains are expanding all of the root node, adding one state for each character. Chains that share a common prefix also share a similar set of nodes originally in trie. We call the distance from the root in the case of trie goto as the depth of that State. To match series, we start from the root node and traverse the edges of the characters according to the selected input. The second phase transformations involves the introduction of the next. When it does not find a match series, it is possible that a subsequent series of previously offset is the prefix of another string in trie, and therefore we use the next slide transitions to different series (branch) in trie. Automated algorithm and the inevitability limited as follows

```
dfsCompressedNext(sto)
{
parentc = cP arent(sto)
forall (a such that depth(nexto(sto, a)) > m)
{
nxt = nexto(sto, a)
pN xtc = cP arent(nxt)
backneed = 0
for all sN xtc ∈ skick(nxt)
backneed = cFwd(sto, parentc, nxt, sN xtc)
if (backneed == 1)
cBkwd(sto, parentc, nxt, pN xtc)
}
for all a such that gotoo(sto, a) 6 = null
dfsCompressedNext(gotoo(sto, a))
}
/* Forward next transition */
cFwd(sto, parentc, nxt, sN xtc)
{
len = dist(sto , parentc) + 1 + dist(nxt, sN xtc)
inp = (string(sto ) − string(parentc )) + a
inp = inp + (string(sN xtc) − string(nxt))
if (len ≤ T W and saf e(nxt) == true)
{
nextc(parentc, inp, len) = sN xtc
out = cout(parentc, sto, ) ∪ outputo(nxt)
```

```
out = out ∪ cout(nxt, sN xtc, )
outputc(parentc, sN xtc, inp) = out
} else
backneed = 1
return backneed
}
/* Backward next transition */
cBkwd(sto, parentc, nxt, pN xtc)
{
len = dist(sto , parentc) + 1 − dist(pN xtc, nxt)
inp = (string(sto ) − string(parentc )) + a
nextc(parentc, inp, len) = pN xtc
if (len > 0)
{
out = cout(parentc, sto, len)
if (len == depth(parentc) − depth(sto) + 1)
out = out ∪ outputo(nxt)
} else
out = null
outputc(parentc, pN xtc, inp) = out
}
```

When matching more than one byte at a time, and the alignment must be taken into account. Determine the alignment of the character in the payload search begins. For example, if the first character from the input stream "A", which is followed by a sequence of six characters 'M', 'a', 'n', 'on', 'g' and 'e'. State machine will not detect the word "management", as we have is a step more than one byte at a time. Can solve this problem by having a system of additional sub parallel matching, which is one byte for the first sub off. A state machine that examines the byte B at the same time, there is a need to B. Sub-systems in the design of FPGA-based, such as [7], and the works of many organs of the State "virtual" to address this problem. In DPI NP-based, B-programmed public research institutions in a load of different starting positions. It is worth mentioning that the memory needed by the state apparatus is still the same when conducting multiple searches simultaneously.

Wide Memory Access

NPS-level support little state machines with AC logic and shift instructions bit of wisdom. Can be programmed independently of the department of public works and the threads on the various organs of the State level a little bit at the same time. However, one limitation is the memory and wide access to the memory components, including on-chip memory. When you bring the word of memory, is used only a fraction of the bits compared to the characters in the packet payload. This imposes a performance penalty (waste of bandwidth memory) to visit the state organs at the level of something, and if they are not stored in a packed. When packed states in a single word, and there is a need to a large number of transformation processes slightly to extract status information and make comparisons. This makes it possible to take the

information from the state to the next state and the element of memory, and bandwidth memory design of the waste is a very effective method of filling is very important.

Limited Number of Processing Elements

Search is performed on the organs of the State AC available by public institutions. For example, mechanism IXP2855 16 dispensation component. Our model, each processing element occupies each byte little machine at the state level in parallel. However, the increase in B or C, or reduce the demand for public institutions, which is not always possible. If public institutions are not enough to handle all the sub at the same time, we need to schedule them in another iteration of the search to subsets remaining until the exhaustion of all subsets.

$$\frac{N_{BLSM}}{N_{subset}} = \frac{8.B}{b} \tag{1}$$

Thus, bordered by a number of state agencies on the level of a number of elements-bit processing. With the rules as each subset, we have

$$[\frac{K}{k}].\frac{N_{BLSM}}{N_{subset}} \leq N_{PE} \tag{2}$$

For example, when $b = 2$, $B = 1$, $N_{PE} = 128$, $k >$d$K/32$e If we want PEs completion of all sub-groups within a single stage. We can be more accurate if we allow sub-PEs to work in a number of iterations.

Experimental Results

Focus on regular expressions that are commonly used in networking applications, we choose a complex pattern following three groups: the first is the candidate of the Linux layer 7 [1], which contains 70 regular expressions to detect different protocols.

We use two sets of effects package in the real world. The first group is the intrusion detection evaluation data from a draft MIT DARPA [24]. It contains more than one million packets. The second data set is the set of a local network of 20 local machine at the University of California at Berkeley, which contains more than 6 million of the total package. MIT discharge characteristics very different from the Berkeley dump. Massachusetts Institute of Technology packages mostly unloading invasion, which is a long time, with a load of data packets is the average length of 507 386 bytes. Dumped in Berkeley, however, most of the parcels to normal traffic, the average load of a package of 67.65 bytes. There is a high proportion of packets are ICMP and ARP packets that are too short. The parameter is the key post let's compression ratio is an indication of the proportion of BCAM condition before to after sharing resources. We can see that a little AC (curve (b)) is superior to the classic AC when k is small, and a bit bytes AC (curve (c)) consistently outperforms the other two. This observation suggests that we have a generalized form of bytes a bit to make it possible to find the optimal setting of the parameters (b, b, k) that are not covered in traditional AC algorithm. We can also see the advantages of using subsets

of the results. Is determined by a number of sub-groups of the number (k) of the keywords in a subset. X-axis shows the different value of k. For the case of non-strategic subsidiary, the results are plotted in the points where it appears as K = 2733. Figure is a clear trend that with the strategy subset, memory usage is reduced significantly T for all three methods.

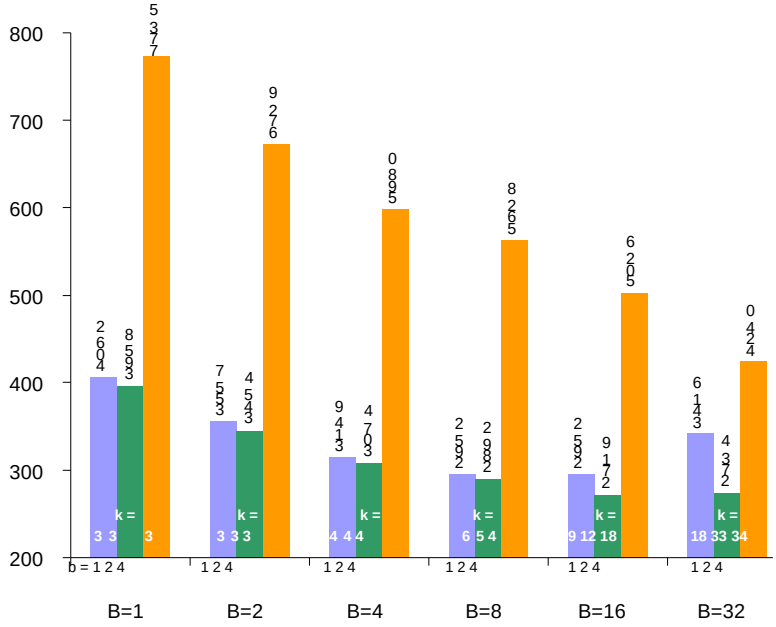

Figure 4: Memory usage results of different *b*.

Observed that the increase in B reduces memory usage significantly until B = 16. behind 16, the largest B does not reduce memory usage. Varying as a different effect on different B: for B <16, the largest K gives worse results. For B≥16, as the largest reduces memory requirements. However, behind the k = 16, there is no significant improvement.

Table 7: Results of grouping algorithms for the multi-core architecture

7 (a) Linux L7-filter (70 Patterns)

Composite DFA state limit	**Groups**	**Total Number of States**	**Compilation Time (s)**
617	10	4267	3.3
2000	5	6181	12.6
4000	4	9307	29.1
16000	3	29986	54.5

7(b) Payload patterns from Bro (222 Patterns)

Composite DFA state limit	Groups	Total Number of States	Compilation Time (s)
540	11	4868	20
1000	7	4656	118
2000	5	5430	780
6000	4	9197	1038

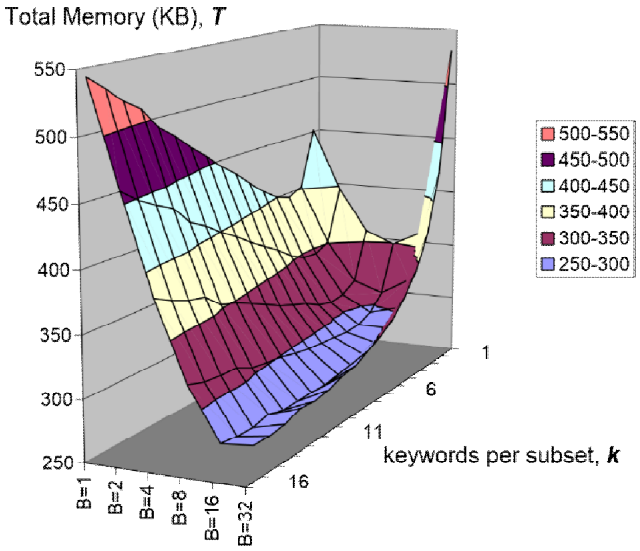

Figure 5: Memory requirement with different *B* and *k* (when *b* = 2).

We study the effect of B on the memory consumption in more detail. Each value of B, and we differ as found using less memory. Results are plotted in Fig. 5, where x is the B axis and the y-axis is standardized memory usage (the key issue is B = 1, T = 395KB). The figure shows that the increase in B can reduce memory usage even B = 16, which exceed the memory usage ramps up. The results show B = 16 can reduce memory requirements by up to 31%. Another observation is interesting for various B, the optimal point of different k. In addition, these increases as the optimal B. Strengthen compared with the DFA on the basis of repeated scanning engine, scanner, and increase our production performance of 1244% to 4238%. We also note that in spite of these dumps have characteristics significantly different, and the scanner provides our productivity similar to these dumps because it scans each character only once.

Discussion and Conclusion

We analyze the model and find the optimal setting, which requires less memory. In this way, limited memory on the chip can be used entirely to improve the overall performance. We propose efficient memory architecture approaching multi-character suit ASIC applications, TDP-DFA. We provide a contribution to the parallel and

overlapping window DFA to implement the treatment in each clock cycle multiple target word. Using a slight modification in the rules of the transition and immediately said it will reduce the complexity of each DFA clear. Experimental results show us that the bytes bit proposed model can be reduced by up to 86% of the memory usage. In addition, we painted a middle path together to further accelerate the matching process. Based on our analyst 1-2 orders of magnitude faster and the speed of 2-3 orders of magnitude of the size of the DFA used widely implemented based on NFA DFA normally used. Plan can be applied in the best combination for a DFA-core processor architecture here, corresponding to the general structure of the processing in the process or thread, which can be placed between the different patterns of parallel processing units.

Reference

[1] Song, T., Zhang, W., Wang, D., & Xue, Y. (2008, April). A memory efficient multiple pattern matching architecture for network security. In *INFOCOM 2008. The 27th Conference on Computer Communications. IEEE*. IEEE.

[2] Vasiliadis, G., Polychronakis, M., & Ioannidis, S. (2011, October). MIDeA: a multi-parallel intrusion detection architecture. In *Proceedings of the 18th ACM conference on Computer and communications security* (pp. 297-308). ACM.

[3] Meiners, C. R., Patel, J., Norige, E., Torng, E., & Liu, A. X. (2010, August). Fast regular expression matching using small TCAMs for network intrusion detection and prevention systems. In *Proceedings of the 19th USENIX conference on Security* (pp. 8-8). USENIX Association.

[4] Le, H., & Prasanna, V. K. (2013). A memory-efficient and modular approach for large-scale string pattern matching. *Computers, IEEE Transactions on*, *62*(5), 844-857.

[5] Lin, C. H., & Chang, S. C. (2011). Efficient pattern matching algorithm for memory architecture. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, *19*(1), 33-41.

[6] Patel, J., Liu, A. X., & Torng, E. (2012). Bypassing space explosion in regular expression matching for network intrusion detection and prevention systems. In*In Proceedings of the 19th Annual Network and Distributed System Security Symposium*.

[7] Kim, H., Kim, H. S., & Kang, S. (2011). A memory-efficient bit-split parallel string matching using pattern dividing for intrusion detection systems. *Parallel and Distributed Systems, IEEE Transactions on*, *22*(11), 1904-1911.

[8] Tran, N. P., Lee, M., Hong, S., & Shin, M. (2012, June). Memory efficient parallelization for aho-corasick algorithm on a gpu. In *High Performance Computing and Communication & 2012 IEEE 9th International*

Conference on Embedded Software and Systems (HPCC-ICESS), 2012 IEEE 14th International Conference on (pp. 432-438). IEEE.

[9] Weng, N., Vespa, L., & Soewito, B. (2011). Deep packet pre-filtering and finite state encoding for adaptive intrusion detection system. *Computer Networks*,*55*(8), 1648-1661.

[10] Cheng, C. J., Wang, C. C., Ku, W. C., Chen, T. F., & Wang, J. S. (2012). A scalable high-performance virus detection processor against a large pattern set for embedded network security. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, *20*(5), 841-854.

[11] Gong, Y., Liu, Q., Shao, X., Pan, C., & Jiao, H. (2014, July). A novel regular expression matching algorithm based on multi-dimensional finite automata. In*High Performance Switching and Routing (HPSR), 2014 IEEE 15th International Conference on* (pp. 90-97). IEEE.

[12] Lin, C. H., Liu, C. H., & Chang, S. C. (2011, December). Accelerating regular expression matching using hierarchical parallel machines on GPU. In *Global Telecommunications Conference (GLOBECOM 2011), 2011 IEEE* (pp. 1-5). IEEE.

[13] Luchaup, D., Smith, R., Estan, C., & Jha, S. (2011). Speculative parallel pattern matching. *Information Forensics and Security, IEEE Transactions on*, *6*(2), 438-451.

[14] Yang, Y. H., & Prasanna, V. K. (2013). Robust and Scalable String Pattern Matching for Deep Packet Inspection on Multicore Processors. *Parallel and Distributed Systems, IEEE Transactions on*, *24*(11), 2283-2292.

[15] Chen, H., Chen, Y., & Summerville, D. H. (2011). A survey on the application of FPGAs for network infrastructure security. *Communications Surveys & Tutorials, IEEE*, *13*(4), 541-561.

[16] Zha, X., Scarpazza, D. P., & Sahni, S. (2011, June). Highly compressed multi-pattern string matching on the Cell Broadband Engine. In *Computers and Communications (ISCC), 2011 IEEE Symposium on* (pp. 257-264). IEEE.

[17] Yang, L., Karim, R., Ganapathy, V., & Smith, R. (2011). Fast, memory-efficient regular expression matching with NFA-OBDDs. *Computer Networks*, *55*(15), 3376-3393.

[18] Vasiliadis, G., Polychronakis, M., & Ioannidis, S. (2011, November). Parallelization and characterization of pattern matching using GPUs. In *Workload Characterization (IISWC), 2011 IEEE International Symposium on*(pp. 216-225). IEEE.

[19] Jiang, L., Tan, J., & Liu, Y. (2012, May). ClusterFA: a memory-efficient DFA structure for network intrusion detection. In *Proceedings of the 7th ACM Symposium on Information, Computer and Communications Security* (pp. 65-66). ACM.

[20] Vespa, L., Weng, N., & Ramaswamy, R. (2011). Ms-dfa: Multiple-stride pattern matching for scalable deep packet inspection. *The Computer Journal*, *54*(2), 285-303.

[21] Hsiao, Y. M., Chen, M. J., Chu, Y. S., & Huang, C. H. (2012). High-throughput intrusion detection system with parallel pattern matching. *IEICE Electronics Express*, *9*(18), 1467-1472.

[22] Lin, C. H., Liu, C. H., Chien, L. S., & Chang, S. C. (2013). Accelerating pattern matching using a novel parallel algorithm on gpus. *Computers, IEEE Transactions on*, *62*(10), 1906-1916.

[23] Ficara, D., Di Pietro, A., Giordano, S., Procissi, G., Vitucci, F., & Antichi, G. (2011). Differential encoding of DFAs for fast regular expression matching. *Networking, IEEE/ACM Transactions on*, *19*(3), 683-694.

[24] Choi, Y. H., Jung, M. Y., & Seo, S. W. (2011). A fast pattern matching algorithm with multi-byte search unit for high-speed network security. *Computer Communications*, *34*(14), 1750-1763.

[25] Zu, Y., Yang, M., Xu, Z., Wang, L., Tian, X., Peng, K., & Dong, Q. (2012, February). GPU-based NFA implementation for memory efficient high speed regular expression matching. In *ACM SIGPLAN Notices* (Vol. 47, No. 8, pp. 129-140). ACM.

[26] Guinde, N. B., & Ziavras, S. G. (2010). Efficient hardware support for pattern matching in network intrusion detection. *computers & security*, *29*(7), 756-769.

[27] Hung, C. L., Lin, C. Y., & Wang, H. H. (2014). An efficient parallel-network packet pattern-matching approach using GPUs. *Journal of Systems Architecture*, *60*(5), 431-439.