Parallel Scalable Approximate Matching Algorithm for Network Intrusion Detection Systems

Adnan Hnaif[1], Khalid Jaber[1], Mohammad Alia[1], and Mohammed Daghbosheh[2]
[1]Faculty of Science and Information Technology, Al Zaytoonah University of Jordan, Jordan
[2]Faculty of Science and Information Technology, Irbid National University of Jordan, Jordan

Abstract: *Matching algorithms are working to find the exact or the approximate matching between text "T" and pattern "P", due to the development of a computer processor, which currently contains a set of multi-cores, multitasks can be performed simultaneously. This technology makes these algorithms work in parallel to improve their speed matching performance. Several exact string matching and approximate matching algorithms have been developed to work in parallel to find the correspondence between text "T" and pattern "P". This paper proposed two models: First, parallelized the Direct Matching Algorithm (PDMA) in multi-cores architecture using OpenMP technology. Second, the PDMA implemented in Network Intrusion Detection Systems (NIDS) to enhance the speed of the NIDS detection engine. The PDMA can be achieved more than 19.7% in parallel processing time compared with sequential matching processing. In addition, the performance of the NIDS detection engine improved for more than 8% compared to the current SNORT-NIDS detection engine.*

Keywords: *Exact matching algorithms, approximate matching algorithms, parallel processing, network intrusion detection systems.*

Received February 13, 2020; accepted June 17, 2020
https://doi.org/10.34028/iajit/18/1/9

1. Introduction

Matching algorithms (searching algorithms) are one of the main topics in the fields of computer science applications, where the aim is to find the pattern "P" in a text "T;" "T" is usually longer than "P" [1, 17]. The searching algorithm can be used to find the exact matching or any pattern "P" close to the text "T" (partially matched). Hence, matching algorithms divided into two types: exact matching algorithms and approximate matching algorithms. The exact matching algorithms are used to find the pattern "P" in the text "T." While the approximate matching algorithms are concerned with finding the similarity percentage between pattern "P" and text "T" [3, 19].

Many experiments were carried out using multi-core technology to accelerate the sequential matching process [18, 22]. This paper introduces a parallel scalable approximate matching algorithm based on Direct Matching Algorithm (DMA) called Parallelized the Direct Matching Algorithm (PDMA), which can be used to find an exact or an approximate matching between pattern "P" and text "T" in parallel. Thus, the PDMA consists of two steps: First, create a two-dimensional array called array-index or matrix "M," to arrange all of the text characters in the "M" based on their positions; second, run the PDMA on multi-cores to find all the occurrences of the pattern "P" in the text "T."

Many computer applications, such as the Intrusion Detection System (IDS), use an exact or approximate matching algorithm to detect the intruders who try to access the network. IDS is one of the network security applications that are responsible for protecting the interior network from intruders [15]. Hence, IDS can be defined as software or hardware to monitor the private network activity from any suspicious behavior and then applying a specific action based on the system security administrator [21]. Therefore, IDS can be used to detect the intrusions on how to try to steal the information or affect the network [16].

Besides, IDS classified into two categories: misuse detection and anomaly detection [10]. Misuse detection is also known as signature-based detection, which defined a pattern that similar common attacks. This technique is efficient in finding known intruders by using any exact matching algorithm but suffers from a slow speed of its detection engine in case of using sequential matching processing [14]. As well, anomaly detection, which also called anomaly-based detection, works based on network behavior [20].

Further, misuse detection used any known exact matching algorithm in the Network Intrusion Detection Systems (NIDS) detection engine by applying the matching process between the incoming packet content and the pre-defined ruleset. Using exact matching algorithm in the NIDS detection engine will increase the possibility of increasing the false positive or false negative alarms, because it is often possible that a portion of the pattern "P" matched with a text "T" in the ruleset, so, the system passes it to the network where it should raise an alert and vice versa.

In order to obtain processing results rapidly, the simultaneous execution of the same task on multiple processors is known as parallel computing. Hence, more than one task can be achieved simultaneously, which produces a significant reduction in the result response time. However, parallel computing plays a very critical role in big data applications such as weather forecasting, data visualization, Biology, engineering, underwater routing, etc., [2, 12].

In parallel computing architecture, all involved nodes (memory or distributed memory) should be connected to set up a parallel environment. In general, Communications are also implementing processing in parallel computation. Therefore, to initiate and configure the messaging environment in parallel computing communications, many message-passing libraries have been developed to send and receive packets of data between processors. The most popular message-passing libraries are Parallel Virtual Machine (PVM) and Message Passing Interface (MPI), whereby POSIX Thread and OpenMP which are considered as the most popular routines in shared address space paradigms [12].

On the other hand, the elapsed time between the beginning and the ending of execution processing on a sequential machine is defined as a serial runtime of a program (Ts). Whereby, the elapses time that specified from the moment of a parallel computation starts to the moment of finishing the last processing is defined as the Parallel run time (Tp).

The rest of the paper is organized as follows: section 2 discusses the related works, and section 3 will present the proposed models of the PDMA and its implementation. Sections 4, 5, and 6 will discuss the Benchmark, test data set, and System Requirements, respectively. Finally, the evaluation results and discussion will be presented in section 7.

2. Related Works

In the following section, we present some of the known algorithms that are used in exact string matching algorithms and approximate matching algorithms. Some of these algorithms are also used in NIDS.

2.1. Exact String Matching Algorithms

The Boyer-Moore algorithm, Quick search algorithm, and Weighted Exact Matching Algorithm (WEMA) are examples of the best-known exact matching algorithms. They were designed for matching between text “T” and pattern “P.” The Boyer-Moore algorithm consists of two phases: The heuristic phase and the good suffix shift phase. The heuristic phase used to create a bad character shift table, which used to determine the number of characters shifted based on pattern length. While the good suffix shift used to look for the occurrences of the substring that matched before when a mismatch occurs. The time complexity of the boyer-moore algorithm is O (mn) [5].

Moreover, the Quick searching algorithm considered as simplicity of the Boyer-Moore algorithm; it has two phases: the first phase is to create a quick search bad character table, which used to determine the number of character shifted in case of mismatch occurs, and the second phase is the searching phase, which used to make a comparison between the pattern “P” and the corresponding characters in the text “T.” The Boyer-Moore algorithm is faster than the quick search algorithm in a long text [5]. While WEMA Hlayel and Hnaif [7] is different from Boyer-Moore and Quick search algorithms, in WEMA, the matching process starts from the minimum character weight that exists in the pattern “P.” If the minimum character weight is equal to zero, then no need to continue the matching process because the pattern “P” does not exist in the text “T.” The limitation of WEMA is unable to find the similarity percentage between the pattern “P” and the text “T.”

Furthermore, Jaber *et al.* [13] presented a framework for parallel Boyer-Moore and Quick search algorithms. The proposed hybrid algorithm implemented using Threads technology or shared memory architecture. On the other hand, Hnaif *et al.* [11] presented a parallel Quick search algorithm by using OpenMP and Pthread to speed up the matching process in the NIDS detection engine between the incoming packet payload and snort ruleset.

Hnaif [8] Parallelized WEMA by using multi-processors with multi-cores and then implemented it to the NIDS detection engine. The author defined a platform to enhance the speed of the detection engine based on WEMA in both sequential and in parallel mode.

2.2. Approximate Matching Algorithms

One of the best approximate matching algorithms is the edit distance algorithm, also known as the Levenshtein Distance algorithm. The algorithm defined the minimum number of insertions, deletions, or substitutions needed to convert the first string into the second string. If the distance between the two strings is zero, that means the two strings are identical [4, 6]. In addition, Hlayel and Hnaif [8] introduced an efficient approximate matching algorithm called DMA, which can be used to find the exact or the similarity percentage between two strings. DMA has an advantage of Levenshtein Distance algorithm in case of the distance between the first string and the second string is zero, because the Levenshtein Distance algorithm must complete all the algorithm steps until it reaches the end of the text, while in DMA, the algorithm goes directly to the all possible locations which can find a match and start from those locations.

3. The Proposed Models of PDMA and Its Implementation

In this section, we introduce two frameworks: first is the framework of PDMA by using a hybrid distributed-shared memory programming model in order to increase the DMA performance (software solution). Second, the framework of the PDMA to be used in the NIDS detection engine, which aims to improve the speed of the NIDS detection engine. Also, the proposed frameworks can be able to run on a single-processor with a multi-core architecture (software solution).

3.1. The Framework of the PDMA

In the PDMA programming model, PDMA runs on a hybrid distributed-shared memory programming model. The workers simultaneously perform a different task on each core.

The PDMA has two phases: the pre-processing phase and the parallel matching phase. In the pre-processing phase, the "M" will be created in preparation for the parallel matching phase, see Figure 1.

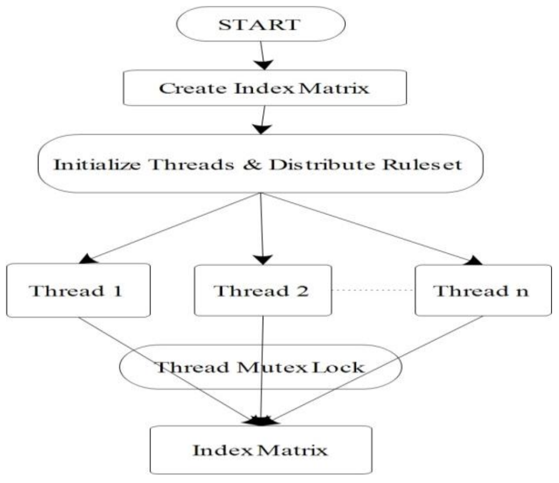

Figure 1. The pre-processing phase of the PDMA.

As shown in Figure 1, the PDMA distributes the ruleset into available cores and based on the number of available cores; several threads will be created, where the optimal number of threads on each core is one. Subsequently, "M" is created. If the ruleset is updated, then "M" will be created automatically in parallel.

The first implemented phase is using OpenMP technology: dynamically OpenMP partitions the iterations of the loop based on the available number of workers (threads), which is made available by the parallel pool. As well as synchronizing tasks is no guarantee anymore. If the number of workers is equal to the number of loop iterations, one loop iteration will be performed by one worker. If their iterations are more than workers, some workers will perform multi-loop iterations to reduce communication time.

As an example, consider "M" ='gcatcgcagaggactcctacgggaggcwgcagagtatacagtacgatg tcgtaataaccccgccccg'. Table 1 depicts the result of the pre-processing phase for "M" (weight "w" is the number of characters repeated).

Table 1. the result of the pre-processing phase for "M".

Pre-processing phase									
M	**Result**								
	Alphabetical character	**Indices of "M" characters**							**Weigh t (w)**
		1	2	3	4	5	…	n	
Gcatcgcagag actcctac ggagcgcag**agt atacag tacg**atgtcg taataacccc gccccgb	a	3	8	10	12	18	…	53	17
	b	64					…		1
	c	2	5	7	13	15	…	62	20
	. .						. .		
	g	1	6	9	11	20	…	63	16
	. .						. .		
	t	4	14	17	31	33	…	51	10

After that, the parallel matching phase will apply to find all the similarities between the text "T" and the pattern "P." Consequently, the parallel matching phase applies as depicted in Figure 2.

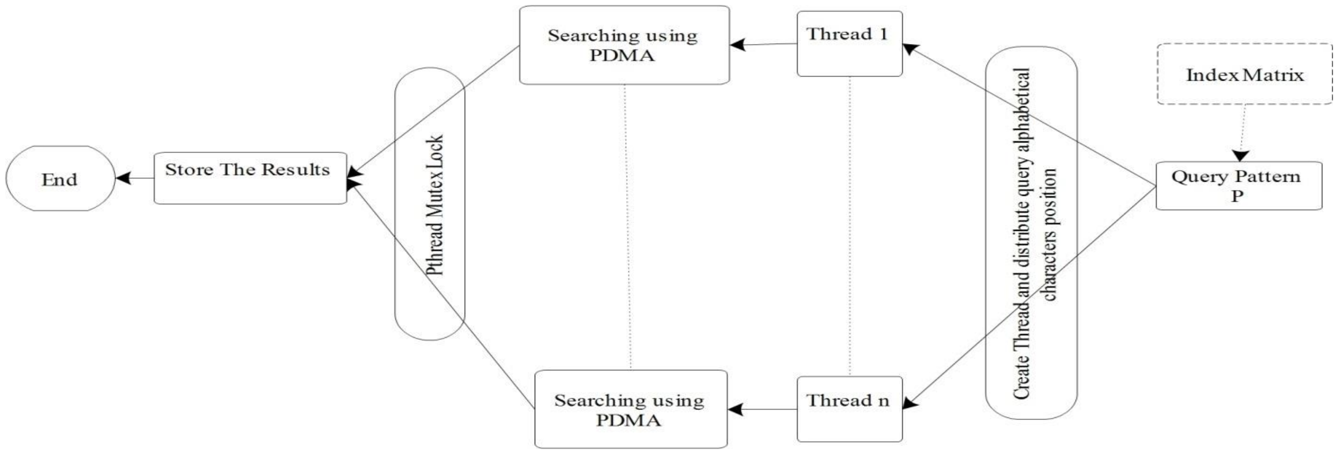

Figure 2. The parallel matching phase of the PDMA.

For instance, "P"='tactgtc,' the searching phase access the character positions directly, and creating the list "L" for each different character. The searching phase starts from the minimum character repeatedly in the pattern "P" to reduce the number of search attempts, in this case, character "a" or "g" (select any one randomly), see Table 2.

Table 2. Create the list "L" for each pattern character.

character repeated	**Positions character in "M."**							
	1	2	3	4	5	6	…	n
t(3)	14	17	31	33	38	43		
a(1)	3	8	10	12	18	22	…	53
c(2)	2	5	7	13	15	16	…	62
g(1)	1	6	9	11	20	21	…	63

As shown in Table 2, the character 'a' has the indices {3, 8, 10, 12, …, 53}, and character 'g' has the indices {1, 6, 9, 11, …, 63} in "M." Regarding the PDMA, we can add the indices of character 'a' or character 'g' to the list 'L' (character 'a' is selected) and create the optimal number of threads under the corresponding position (see Table 3).

Table 3. Determine the minimum character weight.

List "L" for character	**Indices in "M"**						
	1	2	3	4	5	…	n
'a'	3	8	10	12	18		53

Multiple threads are created to search for the character 'a,' where each thread searches at a different position. See Table 4.

As shown in Table 4, thread number 1 starts searching from position 3, and thread number 2 starts searching from position 8, and so on. All created threads search in parallel. See Figure 2.

Table 4. creating multiple threads for "L".

List "L" for character	**Threads and positions**							
	Threads number	1	2	3	4	5	…	Max number of threads
'a'	positions	3	8	10	12	18	…	53

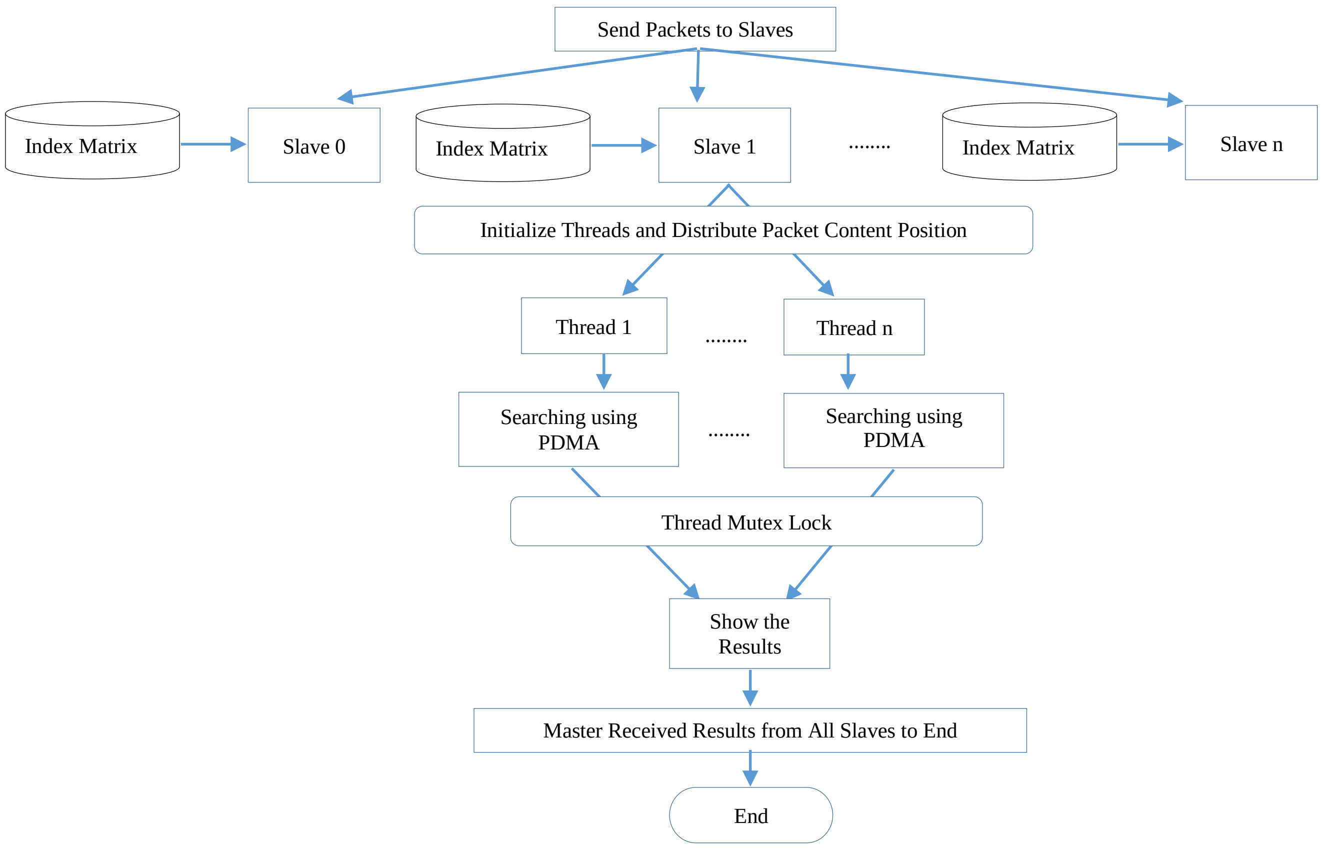

Figure 3. Implemented the PDMA in NIDS.

3.2. The Framework of the PDMA in NIDS

As mentioned, the most effective tool for detecting and preventing intruders attempting to steal information is NIDS. The current NIDS depends on finding the exact matching between the incoming packet payload and ruleset. It is possible to have a similarity between the text and the pattern, for instance, if the text is equal to 'coat' and the pattern is equal to 'cot' (a cot is part of coat). Any exact matching algorithms will not be able to detect that 'cot' is part of 'coat.' Thus, intruders can exploit this issue and log onto the network. Hence, we parallelized the PDMA and applied it in the NIDS to enhance the speed of the NIDS detection engine. Figure 3 shows the overall framework of implemented the PDMA in NIDS. The framework of the PDMA in NIDS implemented using the Task Farming model: this phase is called the Master/Slaves model or also known as a distributed memory programming model. Since master node creates one slave for each possible packet based on the available number of slaves, as shown in Figure 3. Then, the assigned packets send to slaves to be processed via sending the values using MPI. As well as, threads are created on each slave to be executable, so slaves' tasks are assigned to threads. Finally, the Synchronization process is needed before the results are combined and terminating all threads. However, this process must wait to complete all threads and then detaches the threads for joining the final results.

4. Benchmark

To evaluate the performance of the PDMA models, different tests will be performed and compared with DMA and Boyer-Moore, Quick search and WEMA, in order to test the processing time, which is the time needed to find the pattern "p" in the text "T." The effects of variant pattern length will also be examined, with changing the number of cores. Also, different tests will be performed in the NIDS detection engine by using PDMA.

5. Test Data Sets

We used the SNORT NIDS ruleset as an adequate dataset that can be used in NIDS. SNORT NIDS ruleset is one of the widest ruleset used in the NIDS because it includes a collection of intrusions signatures in the network environment.

6. System Requirements

This section presents the setting of the experiment of the proposed PDMA framework and its implementation. All experiments were run on the JadHPC cluster 2.30 GHz, available at the Faculty of Science and Information Technology, Al-Zaytoonah University of Jordan. The operating system is Redhat 7, with the development language being Java.

7. Evaluation Results and Discussion

7.1. Evaluation Results of the PDMA

The DMA algorithm works sequentially, but the ever-increasing internet speed of up to 10 GB/S has become necessary to develop the algorithm to cope with these high speeds with data transmission.

The PDMA is expected to be able to find the exact matching or the similarity matching between the incoming packet payload and ruleset. The presence of high speeds link will not affect the matching process. Accordingly, the matching phase will increase to process a large amount of data with a slower time than it takes in sequential processing time.

The PDMA using multi-cores architecture is testes with data set. The results are obtained according to the comparisons between the implementations of the PDMA and the DMA, which measure the enhancements of the PDMA over DMA. A range of 1000 to 10000 patterns read from the file to search for in the data sets. The comparison result between PDMA and DMA and the speedup is shown in Figures 4 and 5, respectively.

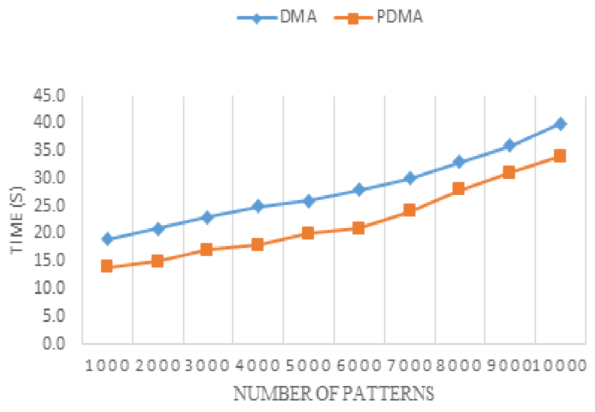

Figure 4. The comparison result between PDMA and DMA.

The relative benefit of solving a problem in parallel is measured by Speedup (*S*). Though, (*S*) is computed for identical processing elements "p" as the ratio of solving a problem time on a single processing element to solving the problem time on a parallel computer; see by Equation (1) [10]. Table 5 shows the time required needed to process the range of 1000-10000 patterns in sequential and in parallel, with several 1-10 threads.

$$S = \frac{Ts}{Tp} \qquad (1)$$

Table 5. Time required to process 1000-10000 patterns with several 1-10 threads.

Number of patterns	Parallel time (ms)	Sequential time (ms)	Speed up
1000	31.0	43.0	1.39
2000	28.0	43.0	1.54
3000	29.0	43.0	1.48
4000	30.0	43.0	1.43
5000	28.0	43.0	1.54
6000	25.0	43.0	1.72
7000	28.0	43.0	1.54
8000	30.0	43.0	1.43
9000	31.0	43.0	1.39
10000	32.0	43.0	1.34

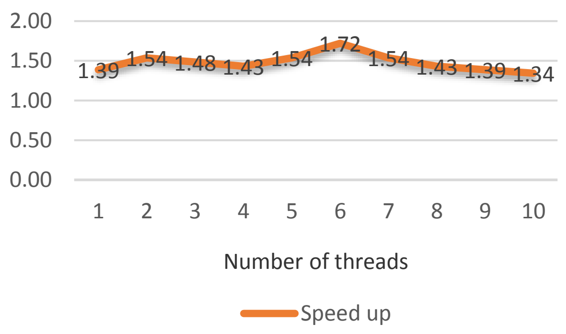

Figure 5. Speedup of the PDMA.

Besides, Figures 6, 7, and 8 represent the comparison results between the PDMA, WEMA, Boyer-Moore, and Quick Search in a parallel manner, the efficiency and overhead.

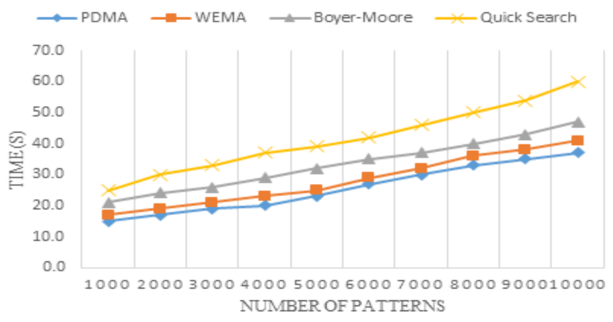

Figure 6. The comparison results between PDMA, WEMA, Boyer-Moore, and Quick Search in parallel.

As well, the efficiency (*E*) is defined as the ratio of the speedup to the number of processing elements that have been processed. The value of (*E*) is between zero and one, depending on the effectiveness of the processing elements utilization. (*E*) can be calculated by Equation (2) [10].

$$E = \frac{S}{P} \tag{2}$$

Where *E*: efficiency, *S*: speedup, and *P*: number of processors.

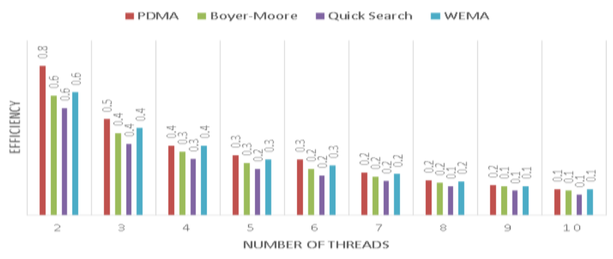

Figure 7. The efficiency results of the PDMA, WEMA, Boyer-Moore, and Quick Search.

The overhead (*To*) describe as the average time spent by all processing elements over the time taken to solve the same problem on a single work element using the fastest-known sequential algorithm. Overhead is given by Equation (3) [10].

$$To = (P * Tp) - Ts \tag{3}$$

Where *To*: Overhead, *P*: number of processors, *Tp*: parallel time, and *Ts*: sequential time.

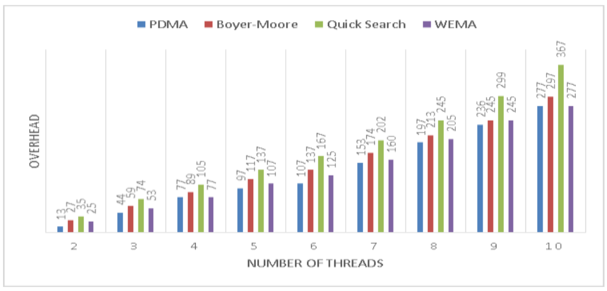

Figure 8. The overhead results of the PDMA, WEMA, Boyer-Moore, and Quick Search.

As depicted in Figure 5, the optimal speedup result in our experiment was obtained using 6 threads. The overhead was acceptable in most results except when using 10 threads. Besides, there are no communication issues in the implementation environment, such as network problems or any message-passing techniques. Therefore, we can relate the amplified overhead in the last result to the following factors: First, The number of threads is more than the number of cores. Second, the time complexity of Threads' creation, sequential distribution of tasks over the threads, distribute the threads into the available cores, and finally, gathering the results from threads.

The proposed framework of the PDMA has an improvement of 19.7% compared to the DMA; also, the PDMA has an advantage over WEMA, Boyer-Moore, and Quick search algorithms in terms of the time needed to complete the searching process.

7.2. Evaluation Matching Process of the PDMA in NIDS

As shown in the previous subsection, the speed performance of the PDMA has improved for more than 19.7%. Thus, the PDMA applied in NIDS using multi-processors with multi-cores architecture. The PDMA is testes with data sets. The results are obtained according to the comparisons between the implementations of the PDMA, Boyer-Moore, Quick Search, and WEMA in the NIDS detection engine. A range of various file sizes (15 KB, 20 KB, 25 KB, and 30 KB) reads to search for in the data sets. The comparison result between PDMA Boyer-Moore, Quick Search, and WEMA is shown in Figures 9.

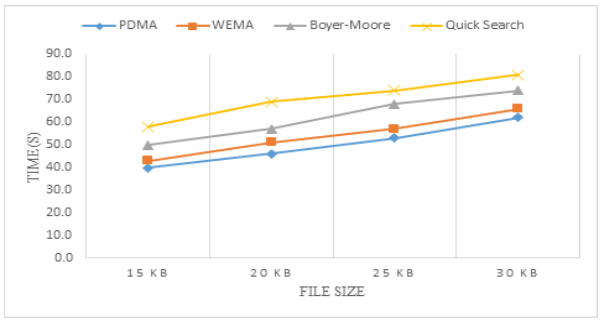

Figure 9. The comparison result between PDMA Boyer-Moore, Quick Search, and WEMA.

7.3. Evaluation Performance

Our experiments have also described the performance of the PDMA over boyer-moore, quick Search, WEMA, Levenshtein Distance, and DMA. For instance, if the incoming packet payload is equal to "gcatcgcag," and one of the SNORT-ruleset is equal to "gaatcggag," then Boyer-Moore, quick Search and WEMA will not be able to detect the similarity between the packet payload and SNORT-ruleset. However, PDMA, Levenshtein Distance and DMA can find the similarity percentage and accordingly apply an appropriate action.

As we have mentioned in section 2.2, DMA has an advantage over Levenshtein Distance. In section 7.1, we proved that PDMA has an advantage over DMA, which leads us to conclude that PDMA has got the best performance (based on time and functionality) to find the exact matching and the similarity between the Text "T" and the Pattern "P." Table 6 summarizes the performance functionality of these algorithms.

Table 6. Summary of performance functionality, where E: Exact matching and S: similarity matching, L.D: Levenshtein Distance.

PDMA		**Boyer-Moore**		**Quick Search**		**WEMA**		**L.D**		**DMA**	
E	S	E	S	E	S	E	S	E	S	E	S
√	√	√	×	√	×	√	×	√	√	√	√

In practice, the evaluated results demonstrated the performance potential of the PDMA, which reached 19.7% in comparison with DMA, and 8% improvement over the rate of NIDS detection engine. It was compared with the current NIDS by using different file sizes, and different packet payload lengths.

The limitation of PDMA that appeared in the experimental work was packet loss due to the slower speed of the scan port of the switch. This leads to reducing the accuracy of the NIDS detection engine. Also, the signature-based NIDS detection engine operates effectively as long as there is no update on the ruleset.

References

[1] Abu-Alhaj M., Abu-Hashem M., Hnaif A, Abouabdalla1 O., Halaiyqah M., and Manasrah A., "An Innovative Platform to Improve the Performance of Exact String-Matching Algorithms," *International Journal of Computer Science and Information Security*, vol. 7, no. 1, pp. 225-227, 2010.

[2] Ashraf S., Aslam Z., Yahya A., and Tahir A., "Underwater Routing Protocols: Analysis of Intrepid Link Selection Mechanism, Challenges and Strategies," *International Journal of Scientific Research in Computer Science and Engineering*, vol. 8, no. 2, pp. 1-9, 2020.

[3] Berman K. and Paul J., *Algorithms: Sequential, Parallel, and Distributed*, Thomson/Course Technology, 2005.

[4] Brakensiek J. and Rubinstein A., "Constant Factor Approximation of Near-Linear Edit Distance in Near-Linear Time," *arXiv:1904.05390v2*, pp. 1-40, 2019.

[5] Charras C. and Lecroq T., "http://www-igm.univ-mlv.fr/~lecroq/string/," Last Visited, 2020.

[6] Goldenberg E., Krauthgamer R., and Saha B., "Sublinear Algorithms for Gap Edit Distance," *in Proceedings of IEEE 60th Annual Symposium on Foundations of Computer Science*, Baltimore, pp. 1101-1120, 2019.

[7] Hlayel A. and Hnaif A., "A New Exact Pattern Matching Algorithm (WEMA)," *Journal of Applied Science*, vol. 14, no. 2, pp. 193-196, 2014.

[8] Hlayel A. and Hnaif A., "An Algorithm to Improve the Performance of String Matching," *Journal of Information Science*, vol. 40, no. 3, pp. 357-362, 2014.

[9] Hnaif A., "A New Platform NIDS Based on WEMA," *International Journal of Information Technology and Computer Science*, vol. 7, no. 6, pp. 52-58, 2015.

[10] Hnaif A., Aldahoud A., Alia A., Al'otoum I., and Nazzal D., "Multiprocessing Scalable String Matching Algorithm for Network Intrusion Detection System," *International Journal of High Performance Systems Architecture*, vol. 8, no. 3, pp. 159-168, 2019.

[11] Hnaif A., Mohammad A., Abouabdalla O., Ramadass S., and Kadhum M., "Parallel Quick Search Algorithm to Speed Packet Payload Filtering in NIDS," *Journal of Engineering Science and Technology*, vol. 4, no. 2, pp. 220-230, 2009.

[12] Jaber K., Alia O., and Shuaib., "M P-HS-SFM: A Parallel Harmony Search Algorithm for the Reproduction of Experimental Data in the Continuous Microscopic Crowd Dynamic Models," *Journal of Experimental and Theoretical Artificial Intelligence*, vol. 30, no. 2, pp. 235-255, 2018.

[13] Jaber K., Dyala R., Al-Sanhani A., and Hamad N., "A Framework for Parallel Boyer-Moore-Quick Search Algorithm (P-BM-QS)," *in Proceedings of 30th IBIMA Conference*, Madrid, pp. 1623-1628, 2017.

[14] Jyothsna V., Prasad V., and Prasad K., "A Review of Anomaly-Based Intrusion Detection Systems," *International Journal of Computer Applications*, vol. 28, no. 7, pp. 26-35, 2011.

[15] Magán-Carrión R., Urda D., Díaz-Cano I., and Dorronsoro B., "Towards a Reliable Comparison and Evaluation of Network Intrusion Detection Systems Based on Machine Learning Approaches," *Applied Sciences*, vol. 10, no. 5, 2020.

[16] Mighan S. and Kahani M., "A Novel Scalable Intrusion Detection System Based on Deep Learning," *International Journal of Information Security*, pp. 1-17, 2020.

[17] Navarro G. and Fredriksson K., "Average Complexity of Exact and Approximate Multiple String Matching," *Theoretical Computer Science 321*, vol. 321, no. 2-3, pp. 283-290, 2004.

[18] Raju S. and Vinayababu A., "Optimal Parallel Algorithm for String Matching on Mesh Network Structure," *International Journal of Applied Mathematical Sciences*, vol. 3, no. 2, pp. 167-175, 2006.

[19] Raju S. and Vinaya A., “Parallel Algorithms for String Matching Problem on Single and Two Dimensional Reconfigurable Pipelined Bus Systems,” *Journal of Computer Science*, vol. 3, no. 9, pp. 754-759, 2007.

[20] Sundararajan R. and Arumugam U., “FBMT: Fuzzy Based Merkle Technique for Detecting and Mitigating Malicious Nodes in Sensor Networks,” *The International Arab Journal of Information Technology*, vol. 16, no. 6, pp. 1106-1113, 2019.

[21] Tabash M., Abd Allah M., and Tawfik B., “Intrusion Detection Model Using Naive Bayes and Deep Learning Technique,” *The International Arab Journal of Information Technology*, vol. 17, no. 2, pp. 215-224, 2020.

[22] You J., Park S., and Kim I., “An Efficient Frequent Melody Indexing Method to Improve the Performance of Query-By-Humming Systems,” *Journal of Information Science*, vol. 34, no. 6, pp. 777-798, 2008.

Adnan Hnaif is an associate professor at the computer science department, Faculty of Science and information technology, Al Zaytoonah University of Jordan. Dr. Hnaif received his Ph.D. degree in Computer Science from University Sains Malaysia-National Advanced IPv6 Centre and Excellence (NAV6) in 2010. He received his MSc degree in Computer Science from the Department of Computer Science in 2003, and obtained his Bachelor's degree in Computer Science from the Department of Computer Science, in 1999/2000. His researches focus on computer networks and communications, wireless sensor networks, network security, parallel processing, and algorithms.

Khalid Jaber is an Associate Professor of Computer Science at the Faculty of Science and Information Technology at the Al-Zaytoonah University of Jordan, director of the E-learning and Open Educational Resource Center, and IEEE Jordan section treasurer since September 2015. He received his B.Sc. degree in Computer Science from Al-Isra University, Amman, Jordan in, 2005. Furthermore, he obtained his M.Sc. and Ph.D. degrees in Computer Science from the Universiti Sains Malaysia, Penang, Malaysia, in 2007 and 2011, respectively. Dr. Jaber's research interest focuses on data representation and the associated algorithms and parallel programming.

Mohammad Alia is the dean of Scientific Research at Al Zaytoonah University of Jordan (ZUJ). He is a professor at the computer information systems department, Faculty of Science Computer and information technology ZUJ. He received the B.Sc. degree in Science from the Al Zaytoonah University, Jordan, in 2000. He obtained his Ph.D. degree in Computer Science from the University Science of Malaysia, in 2008. During 2000 until 2004, he worked at Al-Zaytoonah University of Jordan as an instructor of Computer Sciences and Information Technology. Then, he worked as a lecturer at Al-Quds University in Saudi Arabia from 2004 - 2005. His research interests are in the field of Cryptography and Network security.

Mohammed Daghbosheh is an Assistant Professor of Comp uter Information System at the Faculty of Science and Information Technology -Irbid National University of Jordan. He received his B.Sc. degree in Computer Science from Al-Zaytonneh University, Amman, Jordan in, 2000. Furthermore, he obtained his M.Sc. degrees in Information Technology in 20003, and Ph.D. degrees in Computer Information System from the University of Arab Academy for Banking and Financial Sciences 2012. Dr. Daghbosheh research interest focuses on data security and artificial intelligence