DOI: http://dx.doi.org/10.26483/ijarcs.v8i8.4711

ISSN No. 0976-5697

Volume 8, No. 8, September-October 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

AN EFFECTIVE APPROACH TOWARDS PARALLELIZATION OF NETWORK TRAFFIC ANOMALY DETECTION SYSTEM

D. Ashok Kumar
Associate Professor, Department of Computer Science
Government Arts College, Thuvakudimalai,
Tiruchirappalli, India

S. R. Venugopalan
Scientist, Information and Computing technologies
Aeronautical Development Agency (Ministry of Defence)
Bangalore, India

Abstract: Network traffic data is huge in volume and needs to be processed in real time to detect Intrusions. By utilizing the power of latest Hardware with multi-core processors and GPGPU computing, there is a scope for processing the huge volume of network traffic data in near real-time. This study is intended for examining the potential of Network Anomaly Detection Algorithm (NADA) presented by the authors [9] for parallelization. NADA was parallelized using parallel toolbox functions in Matlab. Other classification algorithms such as Naive Bayes, SVM and Decision trees were also implemented using the pre-defined functions in Matlab and the time taken for execution of these algorithms were compared with NADA for various sizes of data. This study uses the new version of Kyoto University's Intrusion Detection/Evaluation benchmark dataset for experimentation. The parallel performance measures such as time taken, speedup and efficiency are encouraging.

Keywords: Network traffic, Network anomaly detection, speedup, efficiency, multi-core processor, parallelization,

1. INTRODUCTION

Intrusion Detection is the process of analyzing the stream of network traffic for possible intrusions. IDS can be classified into two type based on the detection techniques namely Misuse or signature based and Anomaly based. Signature based systems compare the network packets with the existing malicious signature for any possible intrusions/attacks. Anomaly detection algorithms attempt to profile normal behaviour of the network [6]. In the present digital age and with the huge volume of data floating around, the information security has become utmost importance. The data growth rate and the higher bandwidth & network speed makes it very difficult to process the data in real-time. With the advent of latest processor architectures (with many cores) and GPGPU computing, efficient Intrusion Detection has become a reality if the data is processed in parallel. This study does not attempt to compare the performance metrics such as Detection Rate, Accuracy, False Alarm Rate and F-Score etc. of Intrusion detection classifier. This study intents to compare the execution time for various classifiers and the parallel performance of NADA since NADA outperforms all the other classifiers in terms of serial execution time.

In this paper, the following contributions are made:

- The execution time taken for some of the classifications algorithms were compared with different data sizes.
- The NADA algorithm proposed by Ashok Kumar et al is parallelized.
- The performance metrics such as time taken, speedup and efficiency are presented.

The rest of the paper is structured as follows. In chapter 2, the literature on various classification algorithms for intrusion detection are presented and parallel program performance metrics are discussed. In chapter 3, Experimental setup for this study and the dataset used, data set generation for this study are discussed. Chapter 4 discusses the experimental results. Conclusions of the study are given in Chapter 6.

2. BACKGROUND

The speeds of networks have increased than the speed of processors and the centralized IDS have not scaled [5]. Anomaly detection is one of the important area of research in information security and different algorithms were proposed by various researchers for Anomaly Detection. Naïve Bayes was proposed by Panda, M. et al p for Network Intrusion Detection and the same was compared with back propagation neural network algorithm. It was found that the performance of Naïve Bayes is better in term of False Positive rate [10]. SM Hussein et al compared the performance of Naïve Bayes-net and J48 and recorded that Naïve Bayes performs better in terms of detection rate and time to build model [11]. Amor et al. have compared Naïve Bayes and Decision trees and found that Naïve Bayes performs better for KDD Cup99 dataset [12]. Support Vector Machines is used by many researchers for classification problems. In general SVM deals with two-class problems [13]. Sandhya P., et al have compared the performance of Decision trees with SVM for Intrusion Detection and found that decision trees is better than SVM for Probe, U2R and R2L classes of attacks in terms of accuracy. For DOS attacks SVM is better than Decision trees in terms of accuracy. Decision Tree is capable of handling multi-class problems [3]

Various researchers have used different machine learning techniques for Intrusion Detection. The study of the existing literature reveals that very little had been done towards parallelization of the classification algorithms for intrusion detection. By parallelizing the machine learning & soft computing algorithms and using them for intrusion detection,

the required efficiency requirements can be met. . The main objective of this study is to compare the sequential execution time of various classification algorithms and compare it with NADA and examine the potential for parallelization of NADA algorithm.

A. Performance Metrics

There are several performance metrics associated with parallel programs. These metrics are used to determine best parallel algorithm, Evaluate Hardware platforms and examine the benefits of Parallelism. To evaluate a parallel program, the basic measure required is serial execution time. i.e. time taken by a program if it is executed serially on one processor. Like serial computing in parallel computing also the time and memory are important performance measures. The important goals of parallel programs are Performance and Scalability and the main factors limiting the performance are architectural limitation and algorithmic limitations [4]. Performance is the measure of the capacity to reduce the time to solve the problem when the computing resources are increased. In parallel computing Amdhal's Law is used to predict the theoretical maximum speedup for a program using multiple processors. Speedup and Efficiency are important measures for any parallel programs.

Speedup: It is the ratio between time taken by a serial execution and the parallel execution and is given by the below formula.

$$S(p) = T(1)/T(p) \text{ -->3} \qquad [4]$$

Where T(1) is the execution time with one processor
T(p) is the execution time with p processors

Efficiency: It is the measure of usage of computational resources. It is the ratio performance and resources used to achieve performance and are given below.

Where Efficiency= E(p) = S(p)/p --> 4 [4]
S(p) is the speedup for p processor

A. NADA

Ashok Kumar et al [9] proposed Network Anomaly Detection Algorithm and claims that their algorithm out performs the popular classification algorithms such as Support Vector Machines, Naïve Bayes, ONE-R and Logistic Regression in terms of Detection Rate, False Alarm Rate and F-Score. But the authors have not measured execution time and compared it with other schemes. The numbers of test records were small in number which make it difficult to measure the execution time. The NADA algorithm is given the following Fig. 1.

Input: Training Dataset & Testing Dataset - attack training dataset (a), normal training dataset (n) and testing dataset (t).
Output: Anomaly Detection Performance metrics such as Detection rate, FAR, F-Score etc.

BEGIN // Start of Algorithm

1. *Generate Initial Population/training Dataset that has equal number) of attacks and normal traffic features.*
2. *Read the attack, normal and test traffic data.\Initialize the necessary variables and Read the attack normal traffic data.*
3. *Compute the Centroids. Find the centroid of the attack class and normal class. For numerical attributes the mean (or) average is calculated and for the categorical attributes median is calculated. The centroids will be a set of values (mean and median).*
4. *Compute the distance between the test data and the centroid of the attack/normal dataset using 2.0 norm as given in equation 1.*

$$|X| = \sqrt[2]{\sum_{k=1}^{n} |ai - ti|^2} \qquad \rightarrow 1$$

5. *If the test data is closer to normal centroid and the distance between test data and normal centroid is less than 1.5 times of the distance between the normal and attack centroid then it is labelled as normal else an attack.*
6. *Repeat the above steps (5 and 6) for all the test data.*
7. *Calculate the TP, TN, FP, FN, sensitivity, specificity, FAR, Accuracy, detection rate, F-Score etc.*

END //end of algorithm.

Fig. 1: NADA Algorithm

In this study the execution times is computed for various classification schemes and are compared with NADA.

3. Dataset and Experimental Setup

Network Intrusion Evaluation/Detection dataset from Kyoto University popularly known as Kyoto 2006+ dataset which is used in this study. The new version of Kyoto University Benchmark dataset [1] consists of 10 years of data i.e. from Nov.01, 2006 – Dec. 31, 2015. Earlier version of dataset which has traffic traces from Nov.01, 2006 – Aug. 31, 2009. The dataset has different sets of data with IP addresses and without IP addresses. This dataset consists of 14 conventional factors based on KDD Cup 99 dataset [2], and additional 10 features for effective investigation. The version of dataset has one more new feature which is 'protocol type'. Out of 15 features 3 features are categorical (flag, service & protocol type) and rest 12 features are numerical in nature. The added feature 'protocol type' in the new version is not used and only 14 conventional features are used here. According the authors of [7 and 8], probability function for categorical data and mean-range normalization for numerical data yields better results in terms of detection rate and time to build model for intrusion detection classifiers. In this study, categorical data was normalized using the following probability function (Equation 1) and the numerical data was normalised using the mean range normalization technique (Equation 2).

$$fx\ (x) = Pr\ (X = x) = Pr(\{\, s \in S : X(s) = x\}) \text{ --> 1}$$

$$xi = \frac{vi - min(vi)}{max(vi) - min(vi)} \qquad \text{-->2}$$

The list of features which is used in this study is given below

- ✓ *duration*: length (number of seconds) of the connection
- ✓ *service*: network service on the destination, e.g., http, telnet, etc.
- ✓ *src_bytes*: number of data bytes from source to destination
- ✓ *dst_bytes*: number of data bytes from destination to source

- ✓ *count*: number of connections to the same host as the current connection in the past two seconds
- ✓ *same_srv_rate*: % of connections in the count feature to the same service
- ✓ *serror_rate*: % of connections in the count feature that have ``SYN" errors
- ✓ *srv_serror_rate*: % of connections whose service type is the same to that of the current connection in the past two seconds that have "SYN" errors
- ✓ *dst_host_count*: among the past 100 connections whose destination IP address is the same to that of the current connection, the number of connections whose source IP address is also the same to that of the current connection
- ✓ *dst_host_srv_count*: the number of connections in the dst_host_count feature whose service type is also the same to that of the current connection
- ✓ *dst_host_same_src_port_rate*: % of connections in the dst_host_count feature whose source port is the same to that of the current connection
- ✓ *dst_host_serror_rate*: % of connections in the dst_host_count feature that have "SYN"
- ✓ *dst_host_srv_serror_rate*: % of connections in the dst_host_srv_count feature that "SYN" errors
- ✓ *flag*: normal or error status of the connection
- ✓ *Protocol type*: indicates the type of packets such as TCP, UDP and ICMP.
- ✓ label: indicates whether the session is an attack or not

The last four days of data (new version) i.e. data between 28th Dec 2015 to 31st Dec 2015 is used for experimentation.

Out of 1188869 records, 49.1% (583809) of the records are duplicates and were removed and in the remaining 607060 records 92.9% of the records are attack. Only 43148 records are normal records. This data was split into two sets, 70% for training and 30% for testing. Out of 70% records 20000 records were selected for training by random sampling using IBM SPSS Statistics V20. For data processing Microsoft Office Profession 2010 (Excel) was used. There are 10000 records each of attack and normal class in the training dataset. Similarly 50000 records were selected in random for testing.

The experiments were carried out on a system with Intel Xeon E5 2650 2 Ghz processor with two processors of each having 8 cores of PEs and 32GB memory running Window 7 Professional 64-bit Operating System. The test dataset was processed in parallel with 2, 4, 8 and 16 cores respectively. Test dataset has only 50000 records and is too small for measuring the parallel performance. The dataset was replicated using 'repmat' function in Matlab and five test cases were generated with 1 Million, 2 Million, 5 Million, 10 Million and 20 Million records.

4. EXPERIMENT AND RESULTS

The NADA algorithm was implemented in Matlab V R2015a. Similarly other classification algorithms such as Naïve Bayes, Support Vector Machine (SVM) and Decision Tree were implemented in Matlab using the built-in functions. Experiments were carried out on each test dataset for the above mentioned classification algorithms and the results are given in Table 1.

Table 1: Time taken various classification algorithms

No. of Test Cases	*NADA*	*Naive Bayes*	*Decision Tree*	*SVM*
1 Million	1.79	3.15	2.98	31.77
2 Million	3.22	5.57	5.26	56.03
5 Million	7.29	12.52	12.13	126.07
10 Million	13.79	26.20	23.84	244.55
20 Million	26.71	50.43	47.05	492.41

The time taken by SVM is much higher when compared with other algorithms and is almost 17 times higher than NADA. Naïve Bayes takes almost double the time than NADA algorithm in all the cases. Whereas the time taken by Naïve Bayes and Decision tree are almost similar with marginal differences. The time taken by NADA is the lowest among all the algorithms compared in all test cases.

The above results make NADA a clear candidate for parallelization. NADA algorithm was parallelized using 'parfor' function in parallel toolbox of Matlab. The test dataset was processed in parallel with 2, 4, 8 and 16 cores respectively. The time taken to execute the program is given in Table 2 and Fig. 2.

Table 2: Time taken by Parallel NADA

No. of Test Cases	*No. of Cores/Workers*				
	1	*2*	*4*	*8*	*16*
1 Million	1.79	2.90	1.99	1.45	1.25
2 Million	3.22	5.02	3.01	2.10	1.65
5 Million	7.29	11.07	6.60	4.41	3.20
10 Million	13.79	21.64	12.05	7.35	5.77
20 Million	26.71	41.64	24.02	14.29	9.39

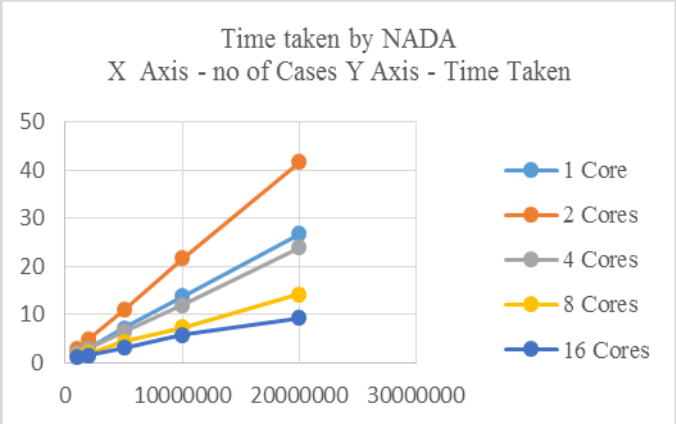

Fig. 2: Time taken by NADA

From the above Table and Figure it is very clear that the time taken by 2 cores/workers is almost 1.5 times higher than the time taken by a single core/worker. Similarly the time taken by 1 core and 4 cores are almost equal and there is slight improvement in 4 core configuration. Clear performance improvements are seen from 8 core onwards. The delay in 2 cores and 4 cores can be attributed to the inter processor/process communication time.

Similarly the other performance measures such as Speedup and Efficiency are calculated from the time taken as given in Table 2. Table 3 lists the Speed up of the NADA Algorithm.

Table 3: Speedup of NADA Algorithm

No. of Test Cases	*2 Workers*	*4 Workers*	*8 Workers*	*16 Workers*
1 Million	0.6172	0.8995	1.2345	1.4320
2 Million	0.6414	1.0698	1.5333	1.9515
5 Million	0.6585	1.1045	1.6531	2.2781
10 Million	0.6372	1.1444	1.8762	2.3899
20 Million	0.6415	1.1120	1.8691	2.8445

From the above table, it can be observed that the speedup increases with the large dataset and number of cores. The efficiency of NADA is given in Table 4.

Table 4: Efficiency of NADA Algorithm

No. of Test Cases	*2 Workers*	*4 Workers*	*8 Workers*	*16 Workers*
1 Million	0.3086	0.2249	0.1543	0.0895
2 Million	0.3207	0.2674	0.2011	0.1220
5 Million	0.3293	0.2761	0.2066	0.1424
10 Million	0.3186	0.2861	0.2345	0.1494
20 Million	0.3207	0.2780	0.2336	0.1778

The scalability of the algorithm needs to be checked for more cores and processors. The parallel measures of Parallel NADA are encouraging.

5. CONCLUSIONS AND FUTUE WORK

The Network Anomaly Detection Algorithm was implemented and parallelized using Matlab parallel toolbox functions. The popular classification algorithms such as Naïve Bayes, SVM and Decision Trees were also implemented in Matlab and the time taken by these methods were compared with NADA. NADA outperforms all the above algorithms with regard to time taken for execution. The parallel performance measures are calculated and discussed in the earlier section. NADA algorithm is a potential candidate for GPGPU parallelization.

6. REFERENCES

1) Song, Jungsuk, Hiroki Takakura, and Yasuo Okabe. “Kyoto University Benchmark Data dataset”, November 2011. URL http://www.takakura.com/kyoto_data/.

2) The third international knowledge discovery and data mining tools competition dataset KDD99-Cup http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999.

3) Peddabachigari, Sandhya, Ajith Abraham, and Johnson Thomas. "Intrusion detection systems using decision trees and support vector Machines." International Journal of Applied Science and Computations, USA 11.3 (2004): 118-134.

4) Fernando Silva & Ricardo Rocha.Parallel and Distributed Programming URL: ttp://www.dcc.fc.up.pt/~fds/aulas/PPD/1112/metrics_en.pdf. Accessed on 2 February 2016.

5) Foschini, Luca, et al. "A parallel architecture for stateful, high-speed intrusion Detection.”International Conference on Information Systems Security. Springer, Berlin, Heidelberg, 2008.

6) Shanbhag, Shashank, and Tilman Wolf. "Accurate anomaly detection through parallelism."IEEE network 23.1 (2009): 22-28.

7) Ihsan, Zohair, Mohd Yazid Idris, and Abdul Hanan Abdullah. "Attribute normalization techniques and performance of intrusion classifiers: A comparative analysis."Life Science Journal. 10.4 (2013).

8) D. Ashok Kumar, and S. R. Venugopalan. "The Effect of Normalization on Intrusion Detection Classifiers (Naïve Bayes and J48)”. International Journal on Future Revolution in Computer Science & Communication engineering, 3.7 (2017): 60-64.

9) D. Ashok Kumar, and S. R. Venugopalan. "A DISTANCE BASED ALGORITHM FOR NETWORK ANOMALY DETECTION USING INITIAL CLASSIFICATION OF'PROTOCOL TYPE'ATTRIBUTE." International Journal of Advanced Research in Computer Science 8.7 (2017).

10) Panda, Mrutyunjaya, and Manas Ranjan Patra. "Network intrusion detection using naive bayes."International journal of computer science and network security, 7.12 (2007): 258-263.

11) Hussein, Safwan Mawlood, Fakariah Hani Mohd Ali, and Zolidah Kasiran. "Evaluation effectiveness of hybrid IDs using snort with naive Bayes to detect attacks." Digital Information and Communication Technology and it's Applications (DICTAP), 2012 Second International Conference on. IEEE, 2012.

12) Amor, Nahla Ben, Salem Benferhat, and Zied Elouedi. "Naive bayes vs decision trees in intrusion detection systems."Proceedings of the 2004 ACM symposium on Applied computing. ACM, 2004.

13) Inadyuti Dutt, Samarjeet Borah. “Some Studies in Intrusion Detection using Data Mining Techniques.” International Journal of Innovative Research in Science, Engineering and Technology, 4.7 (2015):.5500-5511

7. BIOGRAPHIES

Dr. D. Ashok Kumar is an Associate Professor in the Department of Computer Science, Government Arts College, Tiruchirappalli. His current research interests include Data Mining Algorithms, Pattern Matching and Information Security and Systems. His research works have appeared in a variety of international journals and international conference proceedings. He has guided several M. Phil. and Ph. D. Scholars.

S. R. Venugopalan holds M. Sc., M. Phil in Computer Science. He obtained his M.S (by research) in Management from IIT Madras and he is a Scientist in Information & Computing Technologies Directorate of Aeronautical Development Agency, Bangalore. His current research interests are in Information Technology and Information Security, Project Management, Product Lifecycle Management and Enterprise Information Systems and their implementation. His research works have appeared in of international journals and international conference proceedings.