ISSN: 2586-7652 | 2635-7607 Vol. 02, No. 01, March 2019

International Journal of Advanced Engineering

Source: http://ictaes.org

Manuscript received : January 14, 2019 ; Revised : February 25, 2019 ; Accepted : March 10, 2019

Performant Anomaly Based Network Intrusion Detection System Using GPU

Bibek Shrestha, Rupak Karna, Puja Shrestha, Bhuwan Bhatt, Anil Verma

Department of Electronics and Computer Engineering, Pulchowk Campus, Institute Of Engineering, Tribhuvan University, Pulchowk, Lalitpur, Nepal

sbibek@hotmail.com, karan_rupak@hotmail.com, puja5shrestha@gmail.com, b_bhuwan@hotmail.com, anil@ioe.edu.np

Abstract

Networks have been exposed to attack classes such as probe, denial of service, root to local and user to root with increasing connectivity and exposure to the internet. Network Intrusion Detection Systems (NIDSs) alongside firewall is being widely used to identify and minimize attacks. NIDSs can be classified as signature-based or anomaly based depending on the detection method. Anomaly-based NIDSs have the advantage of detecting novel threats over signature-based NIDSs. However, such solutions demand high computing resources when used in high network traffic setups as the number of packets and sessions become high. Graphics Processing Units (GPUs) consist of a large number of logical cores and are optimized for floating point operations. This research aims to analyze the performance gain that can be achieved using GPUs as compared to pure Central Processing Units (CPUs) implementation. The analysis has been performed on subtasks namely session resolution, attributes calculation, normalization, Principal Component Analysis and distance vector calculation with maximum performance gain factor of 13, 185, 7, 1.5 and 10 respectively. The experimental results suggest that GPUs can be used in NIDSs to maximize the performance when used in large scale networks.

Keywords: *GPU Based NIDS, Anomaly Based NIDS, TCP Header Analysis.*

1. INTRODUCTION

Network Intrusion Detection Systems has been a major area of interest in the cybersecurity field owing to the rising threats to normal users, network infrastructures and critical information systems. NIDSs complement other security solutions such as firewalls and SIEM in early detection and prevention from such threats. In general, NIDSs sniff packet from the network infrastructure and perform analysis on each packet or sessions to identify the threats. NIDSs can be broadly classified as signature-based or anomaly based depending on the method used for detection. Signature-based NIDSs such as Snort compares the packets to saved patterns to detect threats and hence are short on detecting novel threats. Anomaly-based NIDSs compares the packets against an established baseline, deviation from which is taken as a threat. Data-mining and machine learning techniques are used to create classifier to classify any network behaviors as an attack or not. Several techniques such as Decision Tree, Support Vector Machine, Artificial Neural Network, Kernel-based Online Anomaly Detection algorithm, etc have been used to model anomaly-based NIDSs as discussed upon by Jyothsna et al.[1].

NIDSs needs to perform various mathematical operations in order to detect threats. With increasing

Corresponding author : Bibek Shrestha
Author's affiliation : Department of Electronics and Computer Engineering, IoE, TU, Nepal.
Email: sbibek@hotmail.com

Copyright © ICT-AES

network traffic, amount of packets or sessions to be analyzed increase as well, thus requiring higher computational power. Performance of modern CPU still seems to be hit when large-scale data is to be processed. Different techniques such as multi-node architecture, FPGA and GPU have been used to overcome such processing limitations. Multi-node architectures depend on multiple deployments of core processing systems with load balancing algorithms. It splits data into independent chunks and process them across multiple subsystem deployments. Field Programmable Gate Arrays (FPGAs) are prefabricated silicon chips that can be programmed electrically to implement custom system designs. FPGAs based NIDSs take advantage of hardware level acceleration as FPGAs are designed to perform specific functions only. GPUs have many smaller-sized logical cores laid down in an architecture capable of processing a set of large amount of identical computations in parallel.

P. Manandhar et al.[2] have proposed a noble technique for anomaly detection using packet header analysis. The method is based on a two-step approach of k-means clustering and One Class SVM algorithms. This approach has yielded promising results in terms of true positive rate, recall, and overall accuracy. This paper reviews their work from a performance perspective by comparing CPU and GPU implementations of sub-tasks.

2. METHODOLOGY

The MIT DARPA '99 dataset was used for training the model, validation, testing, and evaluation. The dataset was created by MIT Lincoln Laboratory, USA by simulating network traffic for about five weeks [3]. The dataset is categorized into weeks. The first three weeks of dataset consists of normal data while fourth and fifth week's dataset includes both normal and attack data. The dataset consists of TCP packets (with protocols such as HTTP, SSH, FTP, Telnet, POP, IRC, XMPP, etc) and UDP packets (with protocols such as DNS, CUPS, WHO, RTP, NBNS, DHCP, NTP, etc). Attacks present in the dataset can be categorized into one of Remote to Local, Probe, User to Remote and Denial of Service [4].

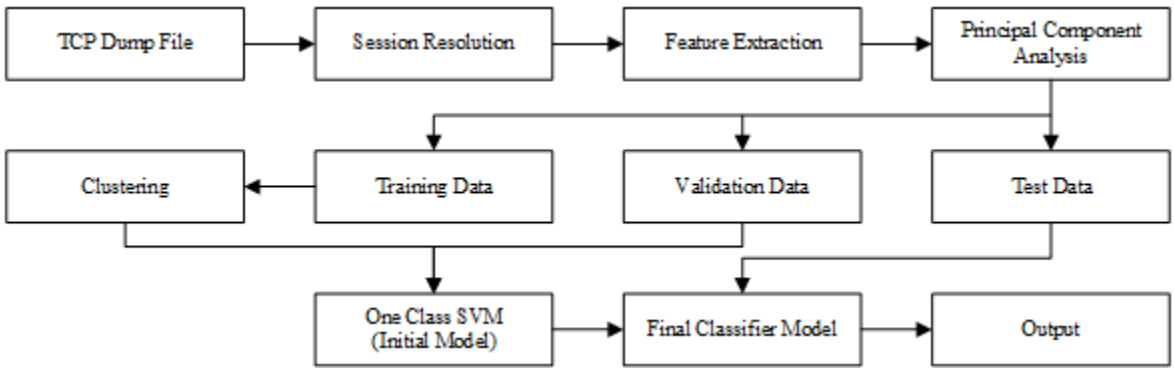


Fig 1. Intrusion Detection System Pipeline

In the first phase, unique sessions were resolved from the dataset. Source address, source port, destination address, destination port were used as parameters to identify unique sessions among the packets. Resolving packets into session using iterative methods is an expensive process. Such method of generating session would be infeasible for high-speed networks slowing down the overall detection rate. To speed up the process, a GPU based algorithm which efficiently helps resolve the huge dataset of network packets into the unique set of sessions has been implemented.

2.1. Resolution of the session from packet in GPU

This algorithm intends to resolve all unique sessions from huge set of packet leveraging parallel capability of GPU. Stable sort implementation in Thrust library was used for sorting the packets in GPU [5].

Precondition: p, q, r, s and t are source IP, source port, destination IP, destination port and array of original indices of data respectively. SortWithinPacket sorts p,q,r,s in a specific sequence for all packets so that every session will have unique key composed of (p,q,r,s). GpuStableSort sorts all packets on basis of sorted key. Final step groups each key and finds the original indices falling in each key representing the session.

```
function RESOLVESESSION(p, q, r, s, t)
        x <= zip(p, q, r, s)
        x <= gpuTransform(x, sortWithinPacket)
        x <= gpuStableSort(x)
        y <= resolveOriginalIndicesForSession(x)
        return y
end function
```

Each of the sessions generated can have a huge amount of TCP packets. Based on the classical information gain theory[6], attributes from each of the packets were selected that appropriately represented the session. Average of selected attributes were then used to model the session. A total of 18 such average attributes were calculated for each session resolved from the dataset and then normalized using logarithmic normalization approach. In case of high-speed networks, the number of concurrent sessions that are running can be extremely high. Hence, attribute calculation and normalization are offshored to GPU for performance gains over CPU implementation.

2.2. Attribute calculation in GPU

This algorithm intends to parallelize the attribute computation using GPU. Attribute calculations include average of numerical features of each the session resolved in the above step.

Precondition: N threads in GPU will execute attribute calculation for each of N sessions resolved. X denotes each of such session.

```
function CALCULATEATTRIBUTE(x)
        y <= calculateAttribute(x)
        return y
end function
```

2.3. Attribute normalization in GPU

This algorithm intends to parallelize the logarithmic normalization by running the routine for normalization in GPU for each of sessions resolved in the step above.

Precondition: N threads in GPU will execute normalization for each of N sessions resolved.

```
function NORMALIZE(x)
        y <= Log(x+1)
        return y
```

end function

Data modeled by a total of 18 dimensions can be a major performance overhead further down the processing pipeline. Hence, Principal Component Analysis (PCA) [7] was used to decrease the dimension of the data along with an analytical method to find out features that explains the maximum variance in data. PCA was performed on all of training, validation and test data. As a result, we were able to reduce the dimensionality of data to 8 with 99.6 percent variance explained. GPU_GSPCA library was used to enhance the performance of this process [8].

In the next phase, one-time process of building threat detection classifier model was carried out. K-means clustering [9] was used to group the normal sessions based on similarity which was trained with normal data. Silhouette value analysis was performed to select the optimal value of K [10] which was found to be three. Resulting centroids of the cluster model was used to model attacks based on proximity. In order to further improve the results, proximity results were in turn used in unison with Support Vector Machines (SVM) to build much better detection model[11][12]. Distance vector (proximity) was calculated for each of the sessions with respect to the centroids which was then used for the construction of one class SVM model in two phases. First, normal connections data were used to create a model which was then used to classify validation data consisting only of normal sessions. Finally, sessions classified as normal by the classifier were used to build the final model to be used as the final classifier for the system.

The final classifier model was then used in the NIDS pipeline where data was fed for detection in batches. For testing purposes, test data was used to generate performance metrics and performance gains at different stages of pipeline. The model takes in distance vector as input. Hence this step is also important to determine the overall throughput of the detection process. In order to speed it up, calculation of distance vector for sessions was offshored to GPU.

2.4. Distance vector calculation in GPU

This algorithm intends to parallelize the calculation of distance vector of each and every session resolved in step above by using GPU. Each session features needs to be compared with three centroids to calculate the distance.

Precondition: x is an array of centroids whose size is n. d is an array of data whose distance is to be computed from centroids. dr, dc are number of rows and columns represented in d. GPU runs the calculation in parallel with minimum of dr*dc threads. Hence, tid is id of thread which are smaller than dr*dc.

```
function CALCULATEDISTANCEVECTOR(x, n, d, dr, dc, tid)
    for i <= 0 to n do
        temp[i][tid] <= pow(d[tid]-x[i][tid%dr], 2)
    end for
    __syncThreads()
    if tid%dr = 0 then
        for j <= 0 to dc do
            for I <= 0 to n do
                temp[i][tid]<= temp[i][tid]+temp[i][j*dr+tid]
            end for
        end for
        __syncThreads()
        for i <= 0 to n do
            res[i] <= temp[i][tid]
        end for
```

```
        end if
        return res
end function
```

3. RESULT

The GPU implementation was tested on NVIDIA GeForce GT 520M while plain CPU implementation was tested on Intel Core i3-3220 processor. The experiment was performed on a very basic setup to identify the cost effectiveness of research. The number of training, validation and test sessions were 571847, 612599 and 288070 respectively. The accuracy of classifier model has been presented in Table 1.

Table 1. Performance metrics of NIDS

Performance Metric(s)	Percent
True Positive Rate	89.31
False Positive Rate	13.61
Recall	89.31
Accuracy	86.67
Precision	41.85
F-Measure	57

The TCP sessions consist of almost same type of packets, hence leading to extensive similarity between normal and attack sessions. Also, the test data consisted of relatively lesser number of attack sessions in comparison to normal sessions. As a result, number of False Positives seemed to be high in comparison to the number of True Positives thus leading to low Precision rate. The F-measure was found to be considerably satisfactory.

The results of performance analysis were recorded for a varying number of sessions from 10000 to 50000. The comparative result of CPU and GPU implementation has been presented in following graphs.

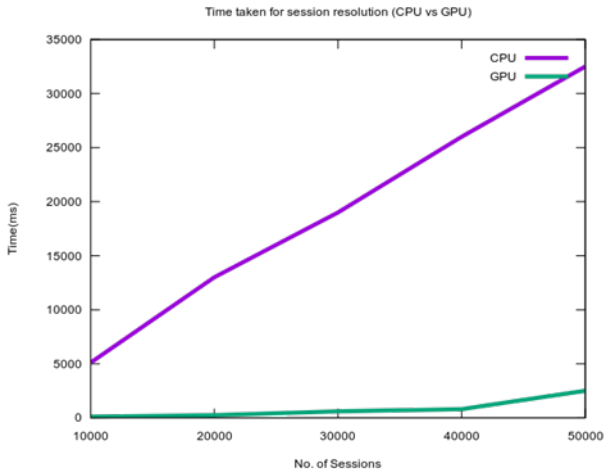


Figure 1. Session Resolution CPU vs GPU

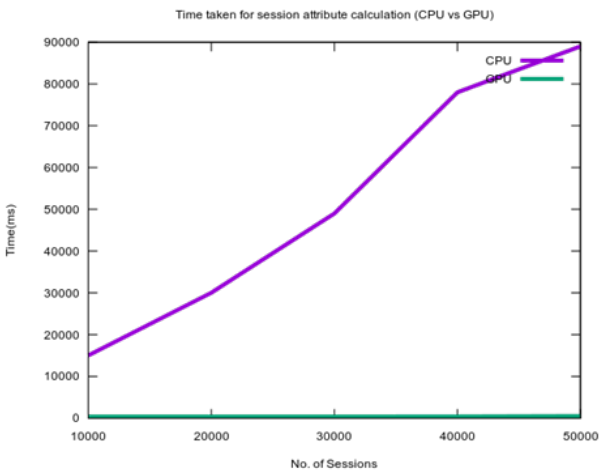
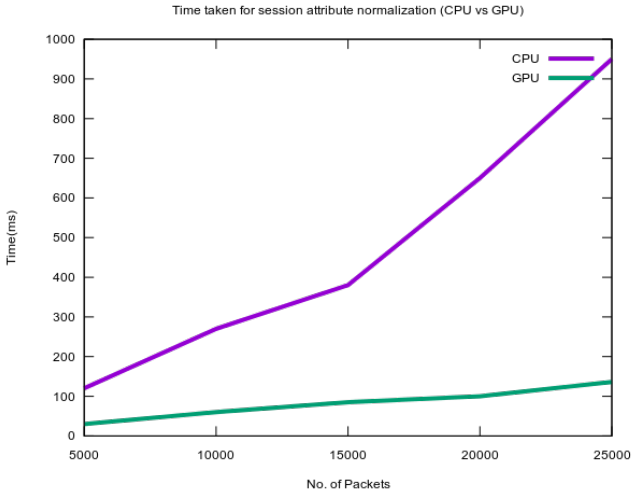


Figure 2. Session Attribute Calculation CPU vs GPU



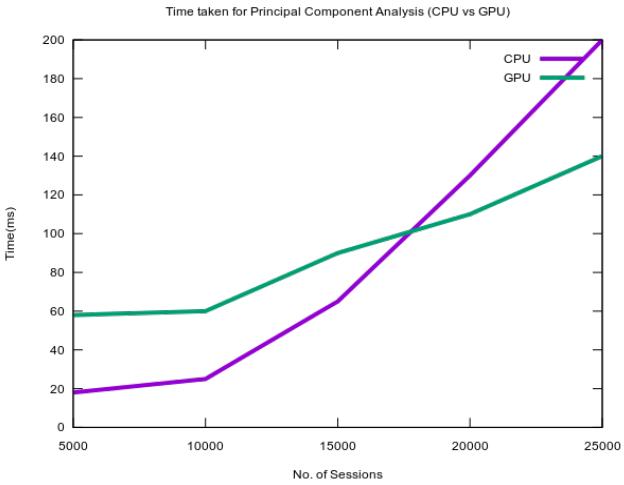


Figure 3. Session attribute normalization CPU vs GPU **Figure 4. PCA CPU vs GPU**

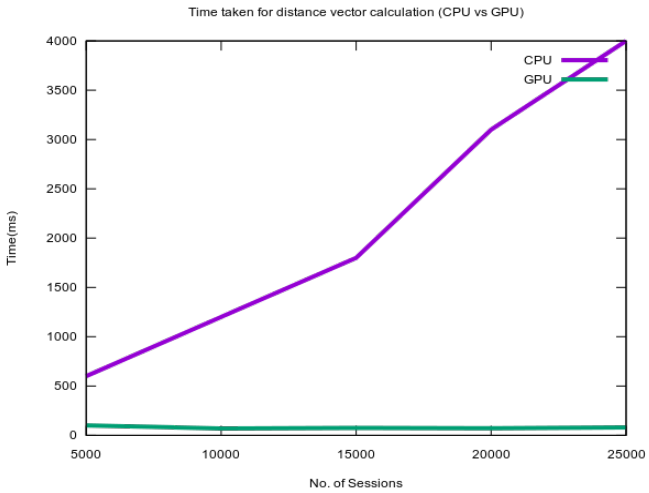


Figure 4. Distance Vector Calculation CPU vs GPU

The results show a general trend of increase in the performance of GPU implementation with an increase in the number of sessions. The maximum performance gain factors obtained during session generation, attributes calculation, normalization, Principal Component Analysis and distance vector calculation were 13, 185, 7, 1.5 and 10 respectively. The performance gain during PCA doesn't seem to be significant, however, its expected to rise with an increase in sessions.

4. DISCUSSION

GPUs are being extensively used in sectors such as gaming, virtual reality, genome sequence analysis, crypto-mining, complex encryption/decryption, financial data analytics and machine learning. The growing use of GPUs is due to the cost effective performance gain, ease of use and maintenance, optimized driver support from vendors and operating systems and their availability in consumer systems. GPUs have previously found their way into network security researches as well. Vasiliadis et al.[13] proposed GPU accelerated signature based NIDS on top of Snort. The research was able to enhance the performance of existing Snort system but didn't

include feasibility analysis of GPU in anomaly based NIDS. Van et al.[14] in their work were able to decrease the training time of neural networks using GPU. However, the research only focuses on detection phase enhancements and not on collection, normalization and dimension reduction phases. Besides GPU, other techniques like multi node deployments and FPGA based implementations are gaining attention as well. Shiri et al.[15] demonstrated a parallel technique to improve performance by deploying multiple nodes to process data. However, this approach is costly and needs additional layers for handling fault tolerance and load balancing. Das et al.[16] used Field Programmable Gate Arrays (FPGAs) based architecture to design high performant NIDS. FPGAs are promising in regards to performance but they require hardware level knowledge for usability and incur some additional design cost. GPUs are cost-efficient and flexible in terms of modification of hardware implementation of the synthesized code as compared to FPGAs, thereby making them a good alternative for practical NIDSs. The performance gain achieved in our research strengthens case for GPUs to be used in anomaly based NIDS.

5. CONCLUSION

This research complements the design of practical anomaly based NIDSs from a performance perspective. The experimental results suggest that GPUs can be used to boost the overall performance of NIDSs and suffice them to be applicable in large-scale networks. The cost-effective performance gain, scalability, and maintainability make them suitable for enterprise-level NIDS systems. The proven efficiency further imply that GPUs can enhance the performance of systems whose subtasks are based on mathematical processing of large-scale data that can be parallelized.

REFERENCES

[1] Jyothsna, V., & Rama Prasad, V. V. (2011, September). A Review of Anomaly based Intrusion Detection Systems. International Journal of Computer Applications, 28(7).

[2] Manandhar, P., & Aung, Z. (2014). Towards practical anomaly-based intrusion detection by outlier mining on tcp packets. International Conference on Database and Expert Systems Applications(DEXA) Part II, 164–173.

[3] Lincoln Laboratory. https://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset.

[4] Thomas, C., Sharma, V., & Balakrishnan, N. (2008). Usefulness of DARPA dataset for intrusion detection system evaluation. 10.1117/12.777341.

[5] CUDA Toolkit Documentation. https://docs.nvidia.com/cuda/thrust/index.html#sorting

[6] Azhagusundari, B., & Thanamani, A. S. (2013, January). Feature Selection based on Information Gain. International Journal of Innovative Technology and Exploring Engineering (IJITEE).

[7] Minka, T.P. (2000). Automatic Choice of Dimensionality for PCA. NIPS. 13. 598-604.

[8] GPU PCA. https://github.com/nmerrill67/GPU_GSPCA.

[9] Münz, G., Li, S., & Carle, G. (2007). Traffic Anomaly Detection Using K-Means Clustering.

[10] Silhouette (clustering). https://en.wikipedia.org/wiki/Silhouette_(clustering).

[11] Laskov, P., Gehl, C., Krueger, S., & Müller, K. (2006). Incremental Support Vector Learning: Analysis, Implementation and Applications. Journal of Machine Learning Research (JMLR). 7.

[12] Tian, S., Mu, S., & Yin, C. (2007). Sequence-similarity kernels for SVMs to detect anomalies in system calls. Neurocomputing, 70, 859-866.

[13] Vasiliadis, G., Antonatos, S., Polychronakis, M., Markatos, E. P., & Ioannidis, S. (2008). Gnort: High Performance Network Intrusion Detection Using Graphics Processors. In Proceedings of the 11th International Symposium on Recent Advances in Intrusion Detection (RAID'08).

[14] Van, N. T. T., & Thinh, T. N. (2015). Accelerating Anomaly-Based IDS Using Neural Network on GPU. 67-74. 10.1109/ACOMP.2015.30.

[15] Shiri, F. I., Shanmugam, B., & Idris, N. B. (2011). A parallel technique for improving the performance of signature-based network intrusion detection system. 2011 IEEE 3rd International Conference on Communication Software and Networks(pp. 692-696). 10.1109/ICCSN.2011.6014986.

[16] Das, A., Nguyen, D., Zambreno, J., Memik, G. & Choudhary, A. (2008). An FPGA-Based Network Intrusion Detection Architecture. Information Forensics and Security. IEEE Transactions on. 3. (118 - 132). 10.1109/TIFS.2007.916288.