ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 5, Issue 5, May 2016

GPU Implementation of Parallel SVM as Applied to Intrusion Detection System

Sudarshan Hiray
Research Scholar,
Department of Computer Engineering,
Vishwakarma Institute of Technology,
Pune, India
sdhiray7@gmail.com

Noshir Tarapore
Assistant Professor,
Department of Computer Engineering,
Vishwakarma Institute of Technology,
Pune, India
ntarapore@yahoo.com

***Abstract*--Intrusion Detection System (IDS) is an important and necessary component in ensuring network security that involves separation of attacks and Non-attacks. Support Vector Machine (SVM) is one of the most popular tools for solving general classification problems because of its high prediction accuracy. However, the training phase of nonlinear kernel based SVM algorithm is a computationally expensive task, especially for large datasets. In this paper, we propose an algorithm to solve large classification problems based on parallel SVM. The system utilizes powerful GPU device and Multithreading to improve the speed performance of SVM training phases. Initial experiments done with KDD CUP 1999 dataset show that the training time of SVM is shortened and the detection accuracy obtained by our method is exactly same or more than as that obtained by serial SVM.**

***Keywords*--Intrusion Detection System (IDS), Support Vector Machine (SVM), Graphics Processor Unit(GPU), Parallel, Cascade, Multithreading**

I. INTRODUCTION

With the rapidly growing connectivity of the Internet, networked computer systems are increasingly playing vital roles in our modern society. While the Internet has brought great benefits to this society, it has also made critical systems vulnerable to malicious attacks. Since a preventive approach such as firewall is not sufficient to provide sufficient security for a computer system, intrusion detection systems (IDS) are introduced as a second line of defense. IDS can detect and identify intrusion behavior or intrusion attempts in a computer system by monitoring and analyzing network packet or system audit log, and then send intrusion alerts to system administrators in real time. This analysis of network packet is basically classification task to separate out attacks and non-attacks. Mostly machine learning tasks are used for this classification process.

The goal of machine learning is discovering, learning, and then adapting to the situations that may change, thus, making improvements in the machine performance. In the intrusion detection field, the input of the references is applied in the machine learning algorithms so that they can learn the attack patterns. After that, the algorithms are used in the unknown input attacks for performing the actual detection process [1]. Some of the known approaches of machine learning are Artificial Neural Network (ANN), Support Vector Machine (SVM) and Genetic Algorithm (GA) [2].

Out of many available machine learning approaches SVM is most popular because of its high prediction accuracy. SVM is based on statistical learning theory, the learning algorithm can solve the small sample learning problems. Using characteristics of SVM, we can extract features and class data, distinguish normal and abnormal data in order to take effective steps to prevent the system from further infringement.

When learning samples set are small, SVM appears to be best approach available but when datasets are larger SVM training consumes large amount of time. This limitation of SVM can affect IDS to great extent [3]. To overcome this limitation use of GPU is proposed which can process SVM training algorithm on different cores simultaneously. GPUs have become faster and more efficient than traditional CPUs in certain types of computations because of its ability to make parallel operations [4]. Also Multithreading can significantly improve training process by dividing dataset into smaller chunks and processing them individually.

In this paper we propose an algorithm which divides large datasets into smaller chunks and then processes these smaller chunks in parallel. We are trying to minimize time required for SVM training using GPU and multithreading approach.

In this paper we take KDD CUP 1999 dataset which is a large dataset containing different types of intrusion types and their features. We apply Cascade SVM approach to process it in parallel in order to get minimum training time for training SVM.

II. RELATED BACKGROUND

A. Support Vector Machine

Intrusion detection is a classification problem in essential. It aims to distinguish normal and abnormal examples with detection algorithm. Support Vector

Machine tries to find an optimization hyper-plane, which can classify the examples into two categories in the feature space [4].

Given training example:

$$(x_1, y_1),...,(x_l, y_l),\ x \in R^d,\ y \in \{+1,-1\},$$

l is the number of training example, d is the dimension of the example feature, $X= x_1, x_2,. . . ,x_d$, y is the class label. If y equal to 1, the corresponding example is normal, else if the label is -1, the corresponding example is abnormal. According to the SVM algorithm, for linear separable problem, there is an optimal separating hyper-plane (decision boundary) to classify two class examples. The optimal separating hyper-plane can be described as:

$$\omega.x + b = 0$$

Here ω is weight and the normal direction of the hyper-plane, b is a bias. The process of solving this hyper-plane, can be considered as, for a given training dataset, seeking weight ω and bias b under some constrains condition to get a minimal cost function:

$$min\ \frac{1}{2}\|\omega\|^2 + C\sum_{i=1}^{n}\xi_i \qquad (1)$$

$$s.t.\quad y_i(\omega.x + b) - 1 >= 0$$

Where, C is a penalty coefficient, ξ is a relax variable, $\xi_i > 0$ [5].

Let $a = a_1, a_2, ..., a_l$, be the nonnegative Lagrange multipliers, one for each inequality constraints in Eq.(1) The solution to Eq.(1) is equal to the solution to the constrained quadratic optimization problem using the Wolfe dual theory as:

$$\max W(\alpha) = \sum_{i=1}^{l}\alpha_i - \frac{1}{2} * \sum_{i=1}^{l}\sum_{j=1}^{l}\alpha_i\alpha_j y_i y_j x_i.x_j$$

$$\text{s.t.}\ \sum_{i=1}^{l} y_l\alpha_i = 0 \quad 0 \le \alpha_i \le C \quad i = 1,2,..l$$

Where α_i are the Lagrange multipliers. If $\alpha_i > 0$, the corresponding example is called support vector (SV). The decision function is:

$$f(x) = sgn\{\textstyle\sum_{i=1}^{l}\alpha^*\ y_i(x.x_i + b^*)$$

sgn() is a sign function, if $f(x) > 0$, the example belongs to normal class, if $f(x) < 0$, the example belongs to abnormal class and if $f(x) = 0$, user can classify it freely. For linear non-separable problem, the training examples in input space X can be mapped into a high dimension space H by kernel function:

$$K(x_i , x_j) = \phi(x).\ \phi(y)$$

then linear classifier in this high dimension space H can be constructed [5].

A number of CPU-based implementations of the SVM methodology are publicly available today. LIBSVM, SVM light, mySVM and TinySVM are only some examples. LIBSVM is one of the most complete implementations, providing many options and state-of-the-art performance [6]. Therefore, it has become the SVM implementation of choice for a large part of the research community. Thus, it was chosen as the basis of our work on GPU implementation of parallel SVMs.

B. Graphics Processor Unit

The Graphics Processing Unit (GPU) is a massively parallel device. It is proved to have excellent performance on floating point calculations, which is even more efficient than multi-core CPUs. The GPU is capable of processing the large amounts of data required to perform the classification and detection tasks very quickly. Many scholars have used GPU to speed up applications. NVIDIA has introduced a programming framework for their GPUs, which is Compute Unified Device Architecture (CUDA) [7]. We can use it to take the advantage of the NVIDIA GPUs' architecture.

There are some SVM implementations offering many functions including different kernels, multiclass classification and cross-validation. LIBSVM is one of the famous SVM implementations which provides different SVM types of classification and regression. It also contains the most popular kernel function such as the linear function, the polynomial function and the radial basis function. Many researches are conducted using LIBSVM. It also provides GPU-accelerated LIBSVM which is a modification of the original LIBSVM that exploits the CUDA framework to significantly reduce processing time while producing identical results [8].

III. RELATED WORK

In this section we provide currently available methods to process parallel SVM, use of GPU for training SVM to reduce time required. We also focus on approaches that can be used to modify currently available methods to get better results.

Xueqin Zhang, Yifeng Zhang, Chunhua Gu have worked on GPU implementation of parallel SVM [9]. They have proposed algorithm GSVM that makes use of GPU to reduce training time. Sequential Minimal Optimization (SMO) only allows a subset containing two samples to be optimized at each step hence training time increases. They have come up with parallel SMO algorithm to improve training time. They have used Scatter-Parallel-Reduce-Gather (SPRG) parallel reduction method to get results from different memory blocks of GPU into one final block. They have achieved great results in terms of speed up and accuracy. Hans Peter Graf, Eric Cosatto, Leon Bottou, Igor Durdanovic, Vladimir Vapnik have proposed an algorithm where Instead of analyzing the whole training set in one optimization step, the data are split into subsets and optimized separately with multiple SVMs [10]. The partial results are combined and filtered again in a next set of SVMs, until best results are achieved. Qi Li, Raied Salman, Vojislav Kecman they propose an intelligent system to solve large classification problems based on parallel SVM [3]. The system utilizes the latest powerful GPU device to improve the speed performance of SVM training and predicting phases. All these techniques have great algorithms to parallelize SVM and achieve speed up using GPU. Drawback in these methods is either using only GPU or only parallel algorithm but not combining these two approaches where combination of these two approaches can lead to even higher speedup.

IV. METHODOLOGY

Proposed methodology uses parallel SVM algorithm which divides large dataset into smaller chunks to be processed in parallel. It uses layered approach towards processing such large datasets where only support vectors from previous layer are passed onto next layer thus discarding unnecessary data points to get accurate results while building SVM model required for prediction. It uses multithreading to process data faster on serial machine so even if we don't use GPU it gives us significant speedup. Though we have combined this multithreading approach with GPU implementation to benefit from both approaches and achieve even higher speedup.

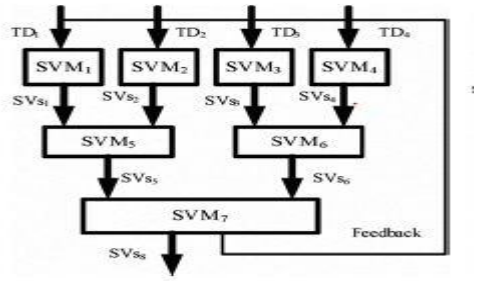

Figure 1: Schematics of Parallel SVM Architecture

If we see the Figure 1 there are different levels in SVM training involved where two SVMs from upper level pass on their Support Vectors to the next level and so on up to last level where only support vectors are present and to improve accuracy of SVM we again send these support vectors to first level and repeat the process until desired accuracy is obtained.

A. Proposed Algorithm of Parallel SVM

Here, Training Data D_T, Support Vector Machine SVM $_i$, Support Vectors Set in current iteration SVs, Support Vector Set in previous iteration SVs ' and Threshold ε denotes minimum change in no. of Support Vectors

1. The original complete training set D_T is divided into 2^n subsets where n = 1, 2,.., n
2. Original SVM training problem is decomposed into 2^n sub-problems. 2^n sub-problems are resolved with parallel SVM. 2^n sub-classifiers and 2^n subsets of support vectors are generated.
3. 2^n support vector sets are combined in groups of two for next step
4. New classifiers are obtained through SVM algorithm with 2^{n-1} new training sets. 2^{n-1} new results and support vector sets are generated.
5. Repeat step 2 until only remaining one support vector set SVs.
6. If, |SVs ' – SVs| <= ε
   Terminate
   Else
   Return to step 2

Example depicted in Figure 1

1. n = 2 therefore D_T is divided into 4 subsets namely D_{T1}, D_{T2}, D_{T3}, D_{T4} and apply it to SVM_1, SVM_2, SVM_3, SVM_4
2. 2^n subsets of support vectors SVs_1, SVs_2, SVs_3, SVs_4 are generated
3. SVM_1 and SVM_2, SVM_3 and SVM_4 are combined.
4. 2^{n-1} new results and support vector sets are generated SVM_5 and SVM_6
5. Only one SVs SVs_7 is remaining
6. Continue iterations until desired results are achieved i.e. if ε = 50 then |SVs ' – SVs| should be greater than 50 to go for another iteration.

B. Dataset

To train and test the parallel SVM model generated by the proposed algorithm, a standard data set is required. We have used KDD CUP '99 and NSL-KDD (improved version of KDD CUP '99) dataset [12]. Both these datasets are standardized and freely available on the internet. This dataset contains information about 22 types of network attacks. Each record in this dataset has 41 features and a class label depicting type of attack. We have used this dataset since it has a large number of records, nearly half a million records, and so we can verify if the proposed algorithm can create an efficient model and is fast enough for a large training dataset also it can fully utilize potential of GPU if the dataset is large enough.

V. EXPERIMENT AND RESULTS

A. Experiment Environment

For Linear SVM we have used LIBSVM version 3.21 which happens to be most widely used SVM library. For making use of GPU to train SVM we have used GPU-accelerated LIBSVM which has LIBSVM accelerated with GPU using the CUDA Framework. For parallel implementation of Parallel algorithm we have taken multithreading approach where multiple threads are assigned to different SVMs for training them simultaneously. In our experiment, the CPU is Intel i3-core E5500 with a frequency of 3.30GHz. The GPU is NVIDIA GeForce GTX 750 ti. The memory capacity is 2 GB, the number of multiprocessors is 16, the number of cores (stream processors) is 640, multiprocessors shared memory size is 16KB, and memory bandwidth is 86.4 GB/S. The integrated development environment is Microsoft Visual Studio 2010 with CUDA SDK 6.5.

Table 1: Training Time (in sec) for different datasets for serial execution

Dataset	CPU Training Time (in sec)
KDD CUP '99	9890
KDD CUP '99 10%	858
NSL-KDD	323
NSL-KDD 20%	78

Table 2: Training Time (in sec) for different datasets for varying partitions using parallel SVM approach

Dataset	Parallel SVM (Multithreading) (time in sec)						
	Number of Partitions						
	4	8	16	32	64	128	256
KDD	3130	2740	2660	2430	2280	2310	2300
KDD 10%	76	48	36	21	15	14	13
NSL-KDD	28	17	15	14	11	10	11
NSL-KDD 20%	18	16	14	12	9	9	9

Table 3: Training Time (in sec) for different datasets for varying partitions using parallel SVM and GPU approach

Dataset	Parallel SVM (Multithreading) + GPU (time in sec)						
	Number of Partitions						
	4	8	16	32	64	128	256
KDD	2890	2620	2460	2380	2340	2270	2290
KDD 10%	67	39	28	20	14	11	11
NSL-KDD	23	15	13	11	10	10	10
NSL-KDD 20%	15	13	12	11	11	10	12

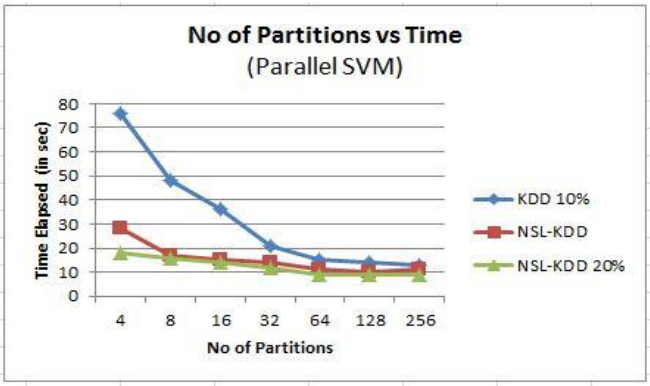

Figure 2: Training time (in sec) for different datasets for varying size of partitions (Parallel SVM)

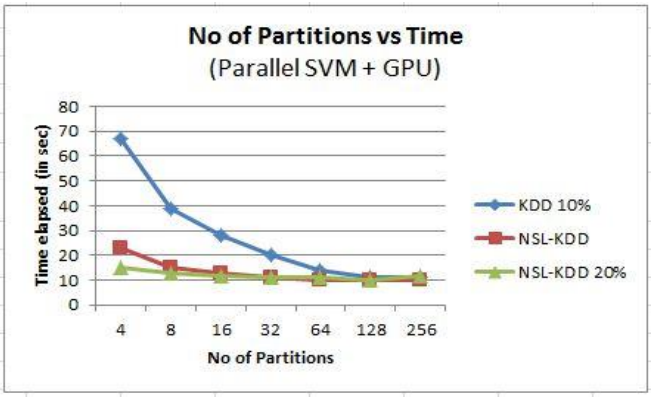

Figure 3: Training time (in sec) for different datasets for varying size of partitions (Parallel SVM + GPU)

Table 4: Speed Up for different datasets for varying partitions using parallel SVM approach

Dataset	Parallel SVM (Multithreading) (Speed Up)						
	Number of Partitions						
	4	8	16	32	64	128	256
KDD	3.1	3.6	3.7	4.0	4.2	4.3	4.3
KDD 10%	11.2	17.8	23.8	40.8	57.2	61.2	66
NSL-KDD	11.5	19.0	21.5	23.0	29.3	32.3	29.3
NSL-KDD 20%	4.3	4.8	5.5	6.5	8.6	8.6	8.6

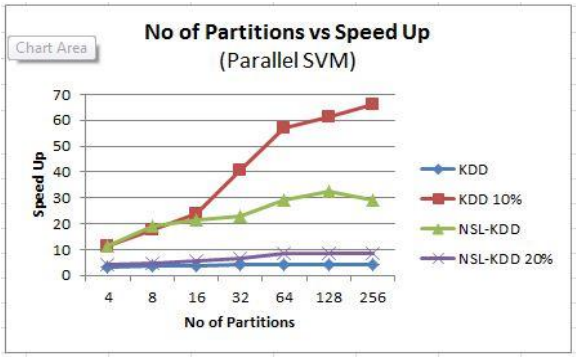

Figure 4: Speed Up for different datasets for varying size of partitions (Parallel SVM + GPU)

Table 5: Speed Up for different datasets for varying partitions using parallel SVM and GPU approach

Dataset	Parallel SVM (Multithreading) (Speed Up)						
	Number of Partitions						
	4	8	16	32	64	128	256
KDD	3.4	3.7	4.0	4.1	4.2	4.3	4.3
KDD 10%	12.8	22	30.6	42.9	61.2	78	78
NSL-KDD	14.0	24.8	26.9	29.3	29.3	32.3	26.9
NSL-KDD 20%	5.2	6.0	6.5	7.0	7.0	7.8	8.6

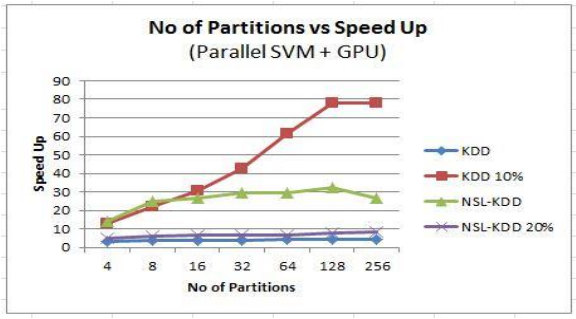

Figure 5: Speed Up for different datasets for varying size of partitions (Parallel SVM + GPU)

B. Analysis of Results

Figure 2 indicates relationship between No. of Partitions and Time elapsed when we use only Parallel SVM algorithm(Multithreading), we can observe that time required for larger dataset is significantly greater than that of smaller dataset but as the no. of partitions grow time required for training SVM has no significant difference regardless of the size of dataset. Figure 3 is similar to that of Figure 2 only little less time is required for training because GPU plays important role in making training faster.

Figure 4 and Figure 5 indicate relationship between No. of Partitions and Speed Up, we can easily see that speed up for two datasets which contains large amount of records is better as No. of partitions grow but for smallest and largest dataset speedup is lesser. Accuracy remains almost same for all of the datasets and also for all number of partitions.

VI. CONCLUSION

In this proposed approach we have applied Parallel SVM algorithm along with GPU acceleration for training SVM for Intrusion Detection. Both these approaches provide great results. Parallel SVM algorithm uses Multithreading approach which trains different SVMs simultaneously. While training Parallel SVM and its GPU implementation we found that for moderately larger datasets time required for training decreases as No. of partitions are increased. Speed Up achieved in comparison with sequential SVM can reach to higher levels. In future this technique can be used for various applications where classification is major task.

REFERENCES

[1] V. T. Goh, J. Zimmermann, and M. Looi, "Towards intrusion detection for encrypted networks," in Availability, Reliability and Security, 2009. ARES'09. International Conference on, 2009, pp. 540-545.

[2] S. Mukkamala, G. Janoski and A.Sung, "Intrusion detection: support vector machines and neural networks", In proceedings of the IEEE International Joint Conference on Neural Networks (ANNIE), St. Louis, MO, pp. 1702-1707, 2002.

[3] T. Joachims, "Making large-scale SVM learning practical", In B. Scholkopf, C. J. C. B-wges, and A. J. Smola eds. Advances in Kemel Methods – Support Vector Leaming**,** Cambridge, MA: MIT press, 1998.

[4] S.Theodoridis and K.Koutroumbas, "Pattern Recognition (Second Edition)", Academic Press, USA, 2003.

[5] Edgar E. Osuna, Robert Freund and Federico Girosi, "Support Vector Machines: Training and Applications", Massachusetts Institute of Technology, Artificial Intelligence Laboratory, A.I. Memo No. 1602 March, C.B.C.L Paper No. 144, 1997.

[6] C. C. Chang and C. J. Lin, "LIBSVM: a library for support vector machines," ACM Transactions on Intelligent Systems and Technology (TIST), vol. 2, no. 3, 2011, pp. 27.

[7] NVIDIA, CUDA Compute Unified Device Architecture Programming Guide, June 2007.

[8] Athanasopoulos, A. Dimou, V. Mezaris, I. Kompatsiaris, "GPU Acceleration for Support Vector Machines", Proc. 12th International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS 2011), Delft, The Netherlands, April 2011.

[9] X. Zhang, Y. Zhang, C. Gu.,"GPU Implementation of Parallel Support Vector Machine Algorithm with Applications to Intruder Detection", JOURNAL OF COMPUTERS, VOL. 9, NO. 5, MAY 2014

[10] Graf H. P., Cosatto E., Bottou L., Durdanovic I., Vapnik V.; "Parallel Support Vector Machines: The Cascade SVM" in Advances in Neural Information Processing Systems no. 17, 521-528, 2004.

[11] Q. Li, R. Salman, V. Kecman, "An Intelligent System for Accelerating Parallel SVM Classification Problems on Large Datasets Using GPU".

[12] KDD Dataset http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html

Authors' Profile:
Sudarshan Hiray is student pursuing Master of Technology, in Computer Science & Engineering from Department of Computer Engineering at Vishwakarma Institute of Technology, Pune.

Noshir Tarapore is Assistant Professor at Department of Computer Engineering in Vishwakarma Institute of Technology, Pune.