Available online at www.ijpe-online.com

International Journal of Performability Engineering

vol. 14, no. 5, May 2018, pp. 1088-1098
DOI: 10.23940/ijpe.18.05.p28.10881098

GPU-Accelerated Support Vector Machines for Traffic Classification

Guanglu Sun[a,b], Xuhang Li[a], Xiangyu Hou[a], and Fei Lang[b,c,*]

[a]*School of Computer Science and Technology, Harbin University of Science and Technology, Harbin, 150080, China*
[b]*Research Center of Information Security & Intelligent Technology, Harbin University of Science and Technology, Harbin, 150080, China*
[c]*School of Foreign Languages, Harbin University of Science and Technology, Harbin, 150080, China*

Abstract

Machine learning model tackles traffic classification effectively. But, it consumes considerable computing resources and computing time, resulting in the difficulty to accommodate large-scale network. In the presented study, GPU-accelerated Support Vector Machines (SVM) is proposed for traffic classification. GPU is used to parallelly calculate the kernel matrix and process the grid traversal of iterative-tuning scheme, in order to accelerate the training and parameters optimization procedure of SVM. Parallel traffic classification is applied to accelerate the classification procedures through the single instruction multiple data paradigm, multithreading and the shared memory of the threads. The experimental results show that the presented method achieves the similar accuracy comparing to the existing CPU-based LibSVM. Furthermore, it ramps up the training speed to 1.53 times and the classification speed to 24 times, which is suitable for the real time classification of high speed backbone networks.

Keywords: traffic classification; graphic processing unit; parallel machine learning; support vector machines; parameters optimization

(Submitted on January 26, 2018; Revised on March 2, 2018; Accepted on April 26, 2018)

1. Introduction

Network traffic classification plays a critical role for network administrators to carry out intrusion detection, network monitoring and management. Increasing network applications and protocols use dynamic ports, hidden ports, and encrypted payloads, which invalidates traditional methods based on port and deep packet inspection [7]. The method based on traffic characteristics and machine learning (ML) has attracted more attention [16]. ML-based method does not need to match port number or string in payload. Este et al. proposed a network flow classification method based on support vector machines (SVM) in 2009, which achieved good classification results [8]. Although SVM model gets outstanding results in many classification problems, it consumes a lot of CPU resources and time in training and classification [1]. With the growth of Internet traffic from TB level to ZB level, traffic classification needs to speed up to 10000 per second [18]. Due to SVM, neural networks, decision tree and other machine learning models consume large CPU resources, it is difficult to solve a large-scale and real time network traffic classification problem on traditional commercial machine platform [13].

The graphics processing unit (GPU) is a parallel computing resource, which has advantages of high floating-point computing power, high bandwidth, low energy consumption and high price-quality ratio [2]. With the rapid increasing of the processing ability and the advent of CUDA architecture, GPU was chosen as a general-purpose processing platform, or collaborative computation with CPU [9,20]. Using the CUDA architecture, the parallel processing method based on GPU speeds up the machine learning model to a great extent [12].

In this paper, a traffic classification method is proposed based on GPU-Accelerated SVM. GPU is utilized to parallelly compute the kernel function in model training. Parallel grid traversal is proposed with parallel cross validation. Two-dimensional hash table is applied to select parameters in order to speed up the training and parameters optimization. Single instruction multiple data, multithreading architecture and GPU shared memory are also used to speed up the classification. Experiments are carried out on the open network data set developed by Nprobe project at University of Cambridge. 1.53

* Corresponding author.
E-mail address: langfeihust@163.com

times speed-up can be achieved in model training, and 24 times speed-up can be achieved in the classification.

2. Related Work

2.1. Traffic classification method based on machine learning model

Since 2004, a traffic classification method based on statistical features and machine learning models has been proposed. Moore et al. extracted 248 kinds of traffic statistical characteristics, and used the Bayesian classification model for traffic classification [15]. Williams compared the performance of traffic classification methods based on 5 machine learning models such as Bayesian Network, C4.5 decision tree, Naive Bayes (NBD, NBK), Bayesian tree and so on [14]. Nguyen, et al. summarized traffic classification methods based on machine learning model in the aspects of models and features [16].

The SVM model is applied to traffic classification, and the method based on SVM has a very high accuracy in traffic classification [14,17]. But, only a small number of data can be used to carry out the experiment because of the long training and classification time of the SVM model. In order to enable the machine learning model to carry out real time traffic classification on high speed Internet, it is necessary to reduce the training and classification time of the model.

2.2. Parallel traffic classification method

Since GPU shows good performance in image processing and scientific computing, GPU is applied to speed up machine learning models [10]. The parallelization of machine learning models can be carried out from two perspectives: model parallelism and data parallelism [13].

Many researchers have studied parallelization from different hardware perspectives to improve the speed of ML-based traffic classification. Santiago et al. used commercial parallel hardware resources to classify multiple traffic simultaneously [19]. Groleat et al. designed a set of parallel classification architecture based on high performance FPGA and used SVM model to carry out high-speed classification [9]. Jiang et al. used location sensitive hash algorithm to improve K nearest neighbor algorithm on FPGA [11]. Barrionuevo et al. reviewed the problem of using GPU to solve large-scale data. He proposed that GPU was suitable for network traffic analysis and classification [3]. Zhou et al. stored the tree models through the shared memory of GPU and put forward C4.5 decision tree method based on GPU [5]. Lopes et al. suggested a GPU-Oriented Stream Decision Tree (GSDT) and used SIMD architecture to parallel classify network traffic [13].

Due to the excellent performance of GPU in parallel processing large-scale data and the good performance of SVM, GPU-Accelerated SVM model is applied to traffic classification, and GPU-Accelerated method is proposed for parameter selection in model optimization.

3. Traffic classification method based on SVM

3.1. SVM model

SVM is a supervised classification model [6]. The linear and non-separable samples in low dimensional space can be transformed into high dimensional space based on relaxation variable and kernel function. SVM can also solve the multi-classification problem by combining multiple binary classifiers [21]. As shown in Figure 1, the points distributed on both sides of the hyperplane are two kinds of sample points. The SVM model constructs a classification hyperplane H (H is expressed in $wx+b=0$), and it separates two kinds of sample points. In Figure 1, A and B are support vectors for determining the boundary H_1; C and D are support vectors for determining the boundary H_2; and the interval is expressed in $2/\|w\|$. The essence of SVM is to ensure that the interval is maximized, which is the optimization problem of convex quadratic programming. In Equation (1), x is represented as the eigenvector of the sample, and y_i is represented the category of the sample. By computing the optimal value of w^* and b^*, the classified hyperplane can be obtained.

$$\begin{aligned} &\min_{w,b}(\|w\|^2/2) \\ &s.t: y_i(w\cdot x+b)\geq 1, i=1,2,...,N \end{aligned} \tag{1}$$

Therefore, the classification decision function can be expressed as Equation (2):

$$f(x) = sign(w^* \cdot x + b^*) \tag{2}$$

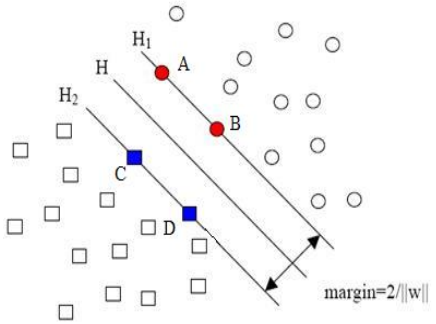

Figure 1. The graphic representation of a linear SVM

However, it is difficult to use strict hyperplane classification to classify because of the unavoidable data noise. Therefore, a slack variable ξ is added to SVM to control the tolerance to the wrong samples. Therefore, the SVM model minimizes the error samples at the maximum interval, and the optimization problem for its convex quadratic programming is expressed as Equation (3).

$$\begin{aligned} &\min_{w,b}(\|w\|^2 / 2 + C\sum_{i=1}^{n}\xi_i) \\ &s.t: y_i(w \cdot x + b) \geq 1 - \xi_i, i = 1, 2, ..., N \end{aligned} \tag{3}$$

C is a constant, which represents a penalty factor. The greater the C value, the more penalty for the error classification. In SVM model optimization, we need to select an appropriate value of *C*, which makes the error samples relatively small, and the interval is maximization. The trained model is the best classification model. By introducing the Lagrange multiplier and KKT conditions, the optimization problem has a dual form, as shown in Equation (4):

$$\begin{aligned} &\min\left(\frac{1}{2}\sum_{i,j=1}^{n} y_i y_j \left(x_i^T x_j\right)\alpha_i\alpha_j - \sum_{i=1}^{n}\alpha_i\right) \\ &s.t.\ \sum_{i=1}^{n} y_i\alpha_i = 0, 0 \leq \alpha_i \leq C \end{aligned} \tag{4}$$

Linear SVM is effective when the feature space of the input data set is large and the sample is sparse distribution. But when the sample feature space is small, the classification effect of linear SVM will be relatively poor. In order to solve this problem, SVM uses a kernel function to map the samples from low dimensional space to high dimensional space, so that the sample points can be divided. As a result, the problem of the optimization of the dual form can be transformed into a Equation (5).

$$\begin{aligned} &\min\left(\frac{1}{2}\sum_{i,j=1}^{n} y_i y_j K\left(x_i, x_j\right)\alpha_i\alpha_j - \sum_{i=1}^{n}\alpha_i\right) \\ &s.t.\ \sum_{i=1}^{n} y_i\alpha_i = 0, 0 \leq \alpha_i \leq C \end{aligned} \tag{5}$$

$K\left(x_i, x_j\right)$ is represented as the kernel function between two samples. The common kernel functions in SVM include linear kernel function, polynomial kernel function, RBF kernel function, Sigmoid kernel function and so on. The kernel function used in this paper is RBF kernel function that is radial basis function, which is usually a monotonic function between the distances from any point in space to a central point, as shown in Equation (6):

$$k(x, y) = \exp(-\gamma \| x - y \|^2) \tag{6}$$

3.2. Framework of traffic classification based on SVM

The proposed method can be divided into three procedures: \ data preprocessing procedure, training procedure and classification procedure, as shown in Figure 2:

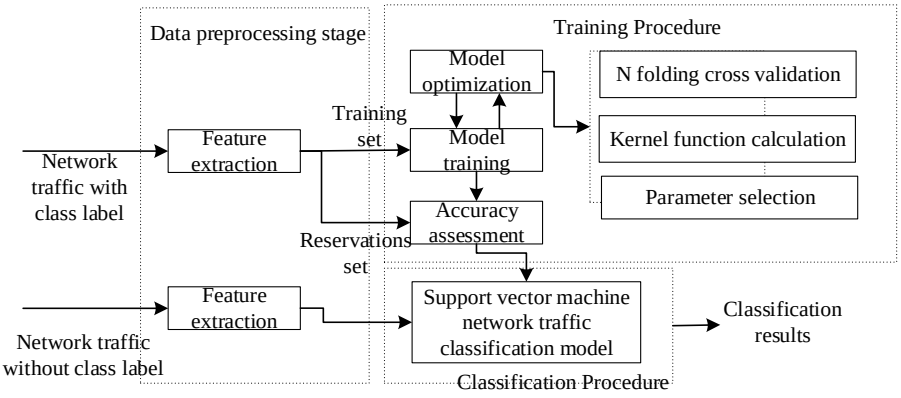

Figure 2. Traffic classification flowchart based on SVM

In the data preprocessing procedure, the statistical characteristics of traffic data with class labels and without class labels are extracted respectively, and the network traffic is expressed as a feature vector form [8,15]. In the traffic classification, based on supervised learning, a large number of data with accurate category tagging are needed to train and optimize the classifier. Among them, the data set used for training is called the training set, and the data set for accuracy evaluation becomes a reservation set.

The training procedure involves the kernel function calculation, the penalty factor C and the kernel parameter γ (RBF kernel function parameter). After the basic model and parameters are obtained, cross validation and parameter selection are used to optimize the C and γ, which can improve the accuracy and robustness of the model. The training of SVM model needs to calculate a large number of kernel functions, mainly including algebraic operation and matrix calculation. The calculation consumes considerable computing resources and computing time with many features. Therefore, the use of parallel computing technology will improve the training speed of the model. In addition, the kernel function used in this study is RBF. In model optimization, cross validation and parameter selection are generally used to get the optimal parameter. Either cross validation or parameter selection requires a lot of computation time. Therefore, the parallelization of these two methods effectively improve the efficiency of model optimization.

In the classification procedure, the unknown traffic is classified by the training SVM traffic classification model, and the classification results are obtained. Because of the independence of different flows, parallel classification of traffic by data parallelization is a feasible way to improve the speed of the classification.

4. Traffic classification method based on GPU accelerating SVM

From the aspect of model parallelism, GPU is used to accelerate parallel computation of kernel function in training procedure, and a method of parallel parameter selection method based on two-dimensional hash data structure and GPU two-dimensional thread block mapping is proposed. From the aspect of data parallelism, in the classification procedure, the parallel classification of network traffic is implemented by SIMD and multithreading architecture, and GPU shared memory.

4.1. Parallel computing method based on GPU in training procedure

The network traffic classification method based on SVM has higher classification accuracy; this method of training model consumes considerable CPU time. The most three time-consuming parts are the kernel function calculation, cross validation and parameter selection under large sample conditions [10].

Firstly, the kernel function calculation is parallelized based on GPU. The partial expansion of the RBF kernel function

is as shown in Equation (7):

$$k'(x,y) = -\gamma \| x - y \|^2 = -\gamma (x \cdot x + y \cdot y) + 2\gamma (x \cdot y) \tag{7}$$

GPU cannot directly access data from main memory; therefore, cudaMalloc function provided by CUDA allocates video memory and copies the data from main memory to video memory. It is necessary to consider the storage method of matrix when the matrix is copied, because a large number of vector calculations are needed in kernel function calculation. CPU usually adopts the way of row storage, while GPU uses the method of column storage to ensure the accuracy of the calculation of the kernel function through the transformation of the matrix. In addition, the cublas linear algebra acceleration library is applied to simplify the calculation operation of the matrix to optimize the computing process of the RBF kernel function [4].

Secondly, cross validation and parameter selection are parallelized based on GPU, which improves the speed of model parameter optimization. Each GPU grid is made up of multiple blocks, each of which includes a number of threads as a computing unit. The interblock thread is independent and different computing tasks can be performed at the same time; it can support the cross validation of parallelization. For example, five sets of cross validation experiments are needed, and a set of calculations on five blocks of GPU can be carried out respectively, which greatly reduces the time of cross validation.

On the basis of the parallel N-fold cross-validation method, this paper proposes a method for selecting parallel parameters based on grid traversal. Grid traversal method is adopted to select parameters based on CPU [21]. By setting the range of C and γ, step size is calculated. It traverses and selects the optimal parameters from multiple parameters. A parallel optimization method based on GPU is also proposed. First, a two-dimensional hash data structure is designed to store the set of parameters (as shown in Figure 3). Each group of parameters corresponds to a two-dimensional cable number. Since the thread index based on GPU started from 1, the first set of parameters is indexed with (1, 1). Then, parameter matrices are set. The two-dimensional thread index supported by GPU is used to get the parameter value stored in the two-dimensional hash data structure, so that a large number of threads of GPU can perform different parameter optimizations at the same time. For example, 5 sets of parameters and 50-fold cross-validation are used to do model optimization. Serial computation for 25 times is needed by using CPU serial computing, and GPU parallel computation can complete 25 times serial computation process at once.

C_1, γ_1	C_1, γ_2	...	C_1, γ_n
C_2, γ_1	C_2, γ_2	...	C_2, γ_n
...	...	...	...
C_n, γ_1	C_n, γ_2	...	C_n, γ_n

Figure 3. Parameter selection matrix

Calculation of the kernel function is also needed in the model optimization procedure. The existing kernel function computing method based on GPU can compute parallel vectors in GPU by calling CPU. It needs to perform the complete computation process of Equation (7). The problem is that partial computation process cannot be parallelized and vector is repeated computation. Therefore, this paper improves the Equation (7) to the Equation (8):

$$k_i'(x,y) = -\gamma_i \| x - y \|^2 = -\gamma_i (x \cdot x + y \cdot y) + 2\gamma_i (x \cdot y) = \gamma_i \left[2(x \cdot y) - (x \cdot x + y \cdot y) \right] \tag{8}$$

In this case, $\Delta = 2(x \cdot y) - (x \cdot x + y \cdot y)$. For each vector pair (x, y) in the same set of cross-validations, Δ is a fixed value.

In CUDA, two keywords are provided for "__device__" and "__global__", and two different called functions are defined respectively. "__device__" defines the device function, which is called and calculated based on GPU. The kernel function calculation process is revised based on the Equation (8). The whole computation process is parallelized in order to avoid repeated computation of vectors. "__global__" defines the global function, which is called based on CPU and calculated based on GPU. In the presented calculation process, the calculation of Δ vector is first parallelized. Then, the

results are stored in the video memory. The calculation results of kernel function are shared by different parameter optimization procedure in the same group of cross validation. Finally, parallel calculation and model optimization of parameters are implemented in the condition that Δ is a constant, C and γ are the variable.

Algorithm 1. Parallel optimization algorithm for model optimization

Input M //Network traffic training set

P //Training parameter set

Output *Param* //Optimal parameters obtained by model optimization

1. Load training set and parameter set： M and P
2. Store M and P in the GPU space through the memory copy : *cuM* and *cuP* / Model Optimization: cross validation, kernel function calculation, parameter selection
3. Let i and j, $i \in [1,\ldots,5]$, $j \in [1,\ldots,cuM_{row} - 1]$// cuM_{row} is expressed as the line of *cuM*
4. cuM_train_i is a training set, and cuM_test_i is a test set
5. for $block_i \in$ Blocks
6. for $thread_i \in$ Threads
7. Get a row vector v_i of the cuM_train_i corresponding to the $thread_i$
8. for $v_{i+1} \in cuM\_train_i$
9. Compute $\Delta_i(v_i, v_{i+1})$of v_i and v_{i+1} //according to formula 8
10. Store $\Delta_i(v_i, v_{i+1})$ to Results
11. end for
12. end for
13. Let $s \in [1,\ldots,cuP_{len}]$,$n \in [1,\ldots,Results_{len}]$
    //cuP_{len} is the length of *cuP*, $Results_{len}$ is the length of *Results*
14. for $param_s \in cuP$
15. for $\Delta_n \in Results$
16. Compute all k_n according to formula 8 , $param_s$ and Δ_n
17. end for
18. end for
19. Training model *models* according to different parameters and the value of a kernel function
20. Compute accurary set $Accs_i$ of models on cuM_test_i
21. end for
22. Calculate the average accuracy *Acc* of the different parameter groups on all blocks
23. Get the maximum *Acc* corresponding to parameter group *Param*

In algorithm 1, *M* is a network traffic training data set, and *P* is a set of pre-set training parameters, which includes *C*, γ, and other parameters of the SVM training model. Param is the optimal parameter obtained by the model optimization. Training data set and a set of parameters are loaded by step 1 and 2. *M* and *P* are stored to the GPU's memory through the way of memory copy. Step 3 to 23 is a model optimization algorithm for SVM, which includes cross validation, kernel function calculation, parameter selection, and so on. Among them, step 3 defines intermediate variables; step 4 uses cross validation method to divide the training set and tests different block set; step 6 to 12 calculate the Δ in the Equation (8) in the kernel function and its calculation results are stored; step 13 to 18 carry out the corresponding calculation process of kernel function under different parameter groups; step 19 to 20 trains the model according to different parameter values and core values, and the corresponding accuracy rate set on the cross validation test set is calculated. Step 22 combines the cross

validation results on different blocks and the average accuracy of the corresponding parameters under different parameters is obtained. In step 23, the parameter group corresponding to the maximum average accuracy is obtained, and the group of parameters are the optimal parameters.

4.2. Parallel computing method based on GPU in classification procedure

Due to GPU memory architecture, thread structure and SIMD principles can ensure a large number of threads in block calculate operation data at the same time. At traffic classification procedure, parallel computing method can also enhance the performance of the model. The classification procedure of SVM network traffic based on GPU acceleration mainly consists of two parts: data copy and parallel classification.

(1) Data copy

First of all, the classifier model is stored in main memory and traffic data, which is classified, is copied to the GPU memory through the cudaMemcpy memory copy method. The classifier model is stored by GPU's shared memory, which ensures that a large number of threads can read the corresponding model data of each thread at the same time.

(2) Parallel classification

Multithreading architecture is used to classify the flows. Firstly, the number of GPU threads used in the classification is set. The feature vector of each flow is copied to a register of thread. Then, the flows are parallel classified by each thread. Finally, the classification results are returned to the CPU storage. Because the parameter information of the model is given, the computation of kernel function in the classification procedure does not involve the problem of multi-parameter computation. The kernel function can be computed parallelly by the original method of Equation (7).

5. Experiments

5.1. Experimental environment and data set

The experimental environment used in this paper is as follows：Intel(R) Core(TM)2 Quad CPU Q9500 @ 2.83GHz；6G DDR3 RAM；Windows 8.1 professional edition；NVIDIA GeForce GTX 750Ti(Graphic Processing Unit Card). NVIDIA GPU has 2GB GDDR5 memory, 640 core stream processing, and computing capacity of 5. The frequency of each core is 1.19GHz. The bandwidth of GPU test is as follows: 4.937GB/s from the host to the device, 4.613GB/s from the device to the host is, and that of the video memory is 86.4GB/s. The program compilation environments used in this article are Microsoft Visual Studio 2013 and CUDA Toolkit 7.

In this paper, SVM method is applied, which is accelerated by GPU, to solve the problem of network traffic classification. The open network data set developed by Nprobe project at University of Cambridge is used to ensure the fairness of the experiment. The data set consists of 10 subsets of data (as shown in Table 1). Each traffic information is composed of 248-dimensional traffic characteristics and one application category information. There are 12 application categories: ATTACK, DATABASE, FTP-CONTROL, FTP-DATA, FTP-PASV, GAMES, INTERACTI, MAIL, WWW, MULTIMEDIA, P2P, SERVICES.

Table 1. Cambridge data set

data set	duration(s)	the number of flows
Entry 01	1821.8	24863
Entry 02	1696.7	23801
Entry 03	1724.1	22932
Entry 04	1784.1	22285
Entry 05	1794.9	21648
Entry 06	1658.5	19384
Entry 07	1739.2	55835
Entry 08	1665.9	55494
Entry 09	1664.5	66248
Entry 10	1613.4	65036

5.2. Evaluation index

(1) Accuracy

The accuracy is used to evaluate the classification performance of the classifier, as shown in Equation 9.

$$accuracy(\%) = \frac{Correct\ number\ of\ classified\ samples}{Total\ sample\ number} * 100\% \quad (9)$$

(2) Speedup ratio

Speedup ratio is the ratio of consuming time by the same task in single processor system and parallel processor system. It is used to measure the performance and effectiveness of parallel system or program parallelization, as shown in Equation 10.

$$S_p = T_s / T_p \quad (10)$$

S_p is represented as the program speedup ratio; T_s is represented as the running time of the program in the serial case; T_p represents the running time of the program in parallel. Assuming that *p* threads execute a parallel program at the same time, the speedup ratio in the ideal situation is $S_p = p$.

(3) Throughput

The throughput of this article is defined as the number of streams that can be classified within the unit time of the classification model, as shown in Equation 11.

$$throughput(stream/s) = \frac{Total\ traffic\ number}{Classification\ time} \quad (11)$$

5.3. Experimental results

(1) The accuracy and the speedup ratio of training based on CPU and GPU

The training data set uses the data sets except of entry 07; the test data set uses entry 07, which contains 55835 samples. In the experiment, the two methods use the same RBF kernel function; the parameter C is 1000, and the γ is 0.1. The results of the experiment are shown in Table 2.

Table 2. Experimental results of two methods on different scales of training data sets

the number of training samples	accuracy (%)	training time(s)		speedup ratio
		LIBSVM	GPU-SVM	
10000	94.4676	3.844	3.188	1.206
20000	96.5392	40.937	30.371	1.348
30000	97.3451	69.484	51.388	1.352
40000	97.5902	97.360	72.750	1.338
50000	98.1693	163.531	120.635	1.356
60000	98.3015	221.891	161.422	1.375
70000	98.4962	342.021	226.651	1.509
80000	98.6550	481.265	326.683	1.504
90000	98.6953	715.187	505.138	1.416
100000	98.7079	935.971	611.673	1.530

Results show that the model solution processes of the two platform methods are the same, and the implementation methods of parallel and non-parallel are different. Therefore, the two methods can get the same accuracy. Experiments on

CPU and GPU are carried out respectively to obtain an average speedup ratio of 1.39 times, and the highest speedup ratio in the experiment is 1.53 times. When the accuracy of model increases, the number of training samples increases. But, it also leads to increasing time of training. The more training samples there are, the more training time can be reduced; then, the more the SVM model training can be sped up with the GPU method. In addition, due to the large data characteristics of network traffic nowadays, new data needs continuously updating by using models. Using GPU can quickly train models to adapt various network environments.

(2) The speedup ratio of classification based on CPU and GPU

The training data consists of 100000 samples, and the test data set gradually increases from 10000 to 100000; parameter setting is the same as (1). The experimental results are shown in Figure 4.

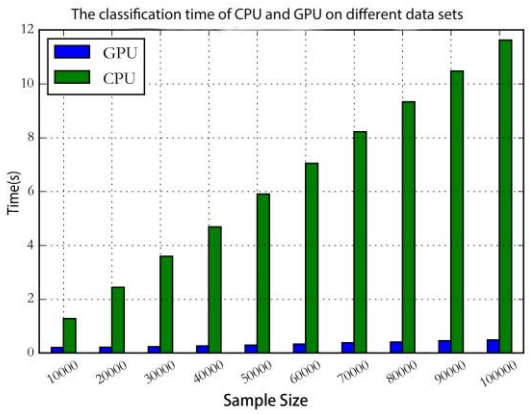

Figure 4. Classification time based on CPU and GPU platforms

From Figure 4, the classification time of the GPU method is much shorter than that of the CPU method. The highest classification speed of 24 times can be reached. With the increase of test sample sets, the time spent on GPU classification is relatively less, which greatly improves the classification speed. Because there are a large number of parallel threads on the GPU, it is impossible to classify multiple streams in parallel at the same time. However, a large number of continuous flows increase the time consuming of data exchange between main memory and video memory, which results in GPU classification system being difficult to achieve the theoretical speedup ratio.

(3) The speedup of the model optimization

The training data consists of 1000 samples. Firstly, a set of parameter values is used to train and verify the performance of the GPU-accelerated method in the calculation part of the kernel function. 100 to 800 GPU threads are selected to calculate the kernel function. As the number of threads increase, the processing time changes, as shown in Figure 5. The abscissa axis represents the number of GPU threads; the ordinate axis represents the processing time. With the increase of GPU number of threads, the computation time of the GPU method will be substantially reduced. Finally, it tends to be stable. The average speedup ratio can reach 2.36 times, as shown in Figure 5.

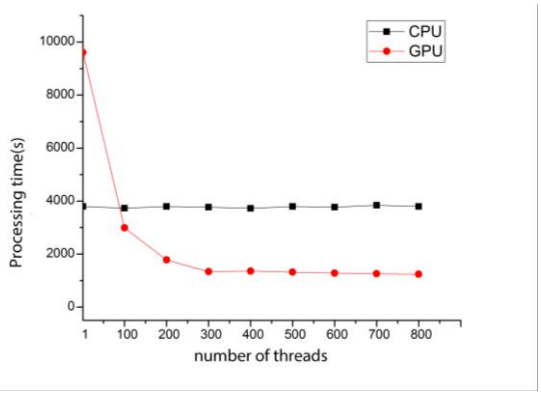

Figure 5. The comparison of processing time between GPU based method and CPU based method without parameter selection

The performance of the parallelized parameter selection method is verified in different parameter scales. The experimental results are shown in Figure 6. The abscissa axis represents the size of the parameters. In the experiment, the n

GPU threads corresponding to the size of the parameters used to select the parameters. Figure 6 shows that the minimum unit of CPU method is the processing time of a single set of parameters. Although the original GPU method carried out parallel kernel function calculation, the method of serial calculation was still used in the parameter selection. The processing time and parameters scale also satisfied monotone linear increasing. The presented method parallelizes the parameter selection in order to calculate multiple parameters simultaneously, which greatly reduces the processing time of parameter selection process. It obtains 98% classification accuracy. The training speedup ratio of 1.53 times and the classification speedup ratio of 24 times are obtained. The theoretical speedup is not achieved in the experiment. One reason is that the bandwidth of the host and equipment lengthen the time of data exchange between CPU and GPU. The other reason is that the whole system, modules and some functions are not suitable for parallelization.

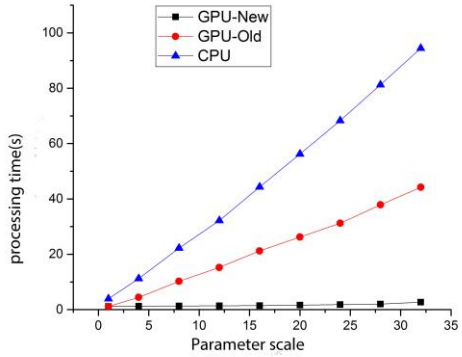

Figure 6. The comparison of processing time among various methods

6. Conclusions

Network traffic classification is one of the key problems in computer network control and management. The machine learning methods effectively solve the identification of encrypted traffic while its computing complexity is high. The learning models are difficult to adapt a large-scale network traffic. In this paper, a SVM method based on GPU is applied to network traffic classification. Parameter optimization based on parallel grid traversal is also proposed to further improve the performance of model optimization. Experiments on Cambridge open data set show that the proposed method is much faster than the CPU based SVM method at the same accuracy rate. The training speedup increases 1.53 times, and the classification speedup increases 24 times.

Acknowledgements

This work was partly financially supported through grants from the National Natural Science Foundation of China (No. 60903083 and 61502123), Scientific planning issues of education in Heilongjiang Province (No. GBC1211062), and the China Postdoctoral Science Foundation (No. 2015M571429). The authors thank the 3 anonymous reviewers for their helpful suggestions.

References

1. A. Athanasopoulos , A. Dimou , V. Mezaris , et al. “GPU Acceleration for Support Vector Machines[C].” WIAMIS 2011: 12th International Workshop on Image Analysis for Multimedia Interactive Services, Delft, The Netherlands, April 13-15, 2011. TU Delft; EWI; MM; PRB, 2011.
2. D. Blythe. “Rise of the Graphics Processor[J].” Proceedings of the IEEE, 2008, 96(5): 761-778.
3. M. Barrionuevo, M. Lopresti, N. Miranda, et al. “Solving a Big-data Problem with GPU: the Network Traffic Analysis[J].” Journal of Computer Science & Technology, 2015, 15.
4. B. Catanzaro, N. Sundaram, K. Keutzer. “Fast Support Vector Machine Training and Classification on Graphics Processors[C].” Proceedings of the 25th international conference on Machine learning. ACM, 2008: 104-111.
5. C. C. Chang, C. J. Lin. “LIBSVM: a Library for Support Vector Machines[J].” ACM Transactions on Intelligent Systems and Technology (TIST), 2011, 2(3): 27.
6. C. Cortes, V. Vapnik. “Support-vector Networks[J].” Machine learning, 1995, 20(3): 273-297.
7. A. Dainotti, A. Pescape, K. C. Claffy. “Issues and Future Directions in Traffic Classification[J].” Network, IEEE, 2012, 26(1): 35-40.
8. A. Este, F. Gringoli, L. Salgarelli. “Support Vector Machines for TCP Traffic Classification[J].” Computer Networks, 2009,

53(14): 2476-2490.
9. T. Groleat, M. Arzel, S. Vaton. “Hardware Acceleration of SVM-based Traffic Classification on FPGA[C].” Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International. IEEE, 2012: 443-449.
10. Y. Hong, C. Huang, B. Nandy, et al. “Iterative-tuning Support Vector Machine for Network Traffic Classification[C].” Integrated Network Management (IM), 2015 IFIP/IEEE International Symposium on. IEEE, 2015: 458-466.
11. W. Jiang, M. Gokhale. “Real-time Classification of Multimedia Traffic Using Fpga[C].” Field Programmable Logic and Applications (FPL), 2010 International Conference on. IEEE, 2010: 56-63.
12. N. Lopes, B. Ribeiro. “GPUMLib: An Efficient Open-source GPU Machine Learning Library[J].” International Journal of Computer Information Systems and Industrial Management Applications, 2011, 3: 355-362.
13. P. Lopes, S. Fernandes , W. Melo, et al. “GPU-oriented Stream Data Mining Traffic Classification[C].” Computers and Communication (ISCC), 2014 IEEE Symposium on. IEEE, 2014: 1-7.
14. Z. Li, R. Yuan, Guan X. “Accurate Classification of the Internet Traffic based on the Svm Method[C].” Communications, 2007. ICC'07. IEEE International Conference on. IEEE, 2007: 1373-1378.
15. A. W. Moore, D. Zuev. “Internet Traffic Classification Using Bayesian Analysis Techniques[C].” ACM SIGMETRICS Performance Evaluation Review. ACM, 2005, 33(1): 50-60.
16. T. Nguyen, G. A. Armitage. “Survey of Techniques for Internet Traffic Classification Using Machine Learning[J].” Communications Surveys & Tutorials, IEEE, 2008, 10(4): 56-76.
17. J. D. Owens, M. Houston, D. Luebke, et al. “GPU Computing[J].” Proceedings of the IEEE, 2008, 96(5): 879-899.
18. P. M. Santiagodel Rio , D. Rossi, F. Gringoli, et al. “Wire-speed Statistical Classification of Network Traffic on Commodity Hardware[C].” Proceedings of the 2012 ACM conference on Internet measurement conference. ACM, 2012: 65-72.
19. A. Vishwanath, V. Sivaraman, M. Thottan. “Perspectives on Router Buffer Sizing: Recent Results and Open Problems [J].” ACM SIGCOMM Computer Communication Review, 2009, 39(2): 34-39.
20. E. H. Wu. “State of the Art and Future Challenge on General Purpose Computation by Graphics Processing Unit[J].” Journal of Software, 2004, 15(10): 1493-1504.
21. S. Zhou, P. R. Nittoor, V. K. Prasanna. “High-Performance Traffic Classification on GPU[C].” Computer Architecture and High Performance Computing (SBAC-PAD), 2014 IEEE 26th International Symposium on. IEEE, 2014: 97-104.

Guanglu Sun is currently a professor in the School of Computer Science and Technology, and a director in the Center of Information Security and Intelligent Technology at Harbin University of Science and Technology, China. He received his B.S., M.S. and Ph.D. from Harbin Institute of Technology. His research interests include computer networks and security, machine learning, and intelligent information processing.

Xuhang Li is a master candidate in the School of Computer Science and Technology at Harbin University of Science and Technology. His research interests lie in computer networks and security, machine learning.

Xiangyu Hou is a master candidate in the School of Computer Science and Technology at Harbin University of Science and Technology. His research interests lie in computer networks and security, machine learning.

Fei Lang is an associate professor in the School of Foreign Languages, and a researcher in the Center of Information Security and Intelligent Technology at Harbin University of Science and Technology. She received her M.S. from Nottingham University and her Ph.D. from Shanghai International Studies University. Her research interests include corpus linguistics, cognitive psychology and machine-aided translation.