GPU - ACCELERATED PERSONALIZATION IN WEB USAGE MINING WITH NEURAL NETWORK ALGORITHM

Dr.S.SANTHI[1], Ms.D.THAMARAISELVI[2]
[1]Associate Professor, Department of Computer Science and Engineering,
[2]Assistant Professor, Department of Computer Science and Engineering,
SCSVMV University, Enathur, Kanchipuram, Tamilnadu
Email: [1]ssanthi@kanchiuniv.ac.in

Abstract— This paper adopt the general purpose GPU parallel computing model and show how it can be leveraged to increase the accuracy and efficiency of personalized recommendation system to the web users. To illustrate the competence of the GP-GPU (General Purpose – Graphics Processing Unit) computing, Back propagations algorithm is implemented with CUDA (Compute Unified Device Architecture) for execution on a GPU device. The results indicate that the algorithm is markedly faster than the sequential algorithm and is not affected by the dimensionality of the web data being classified and well suited for high dimensional problems.

Index Terms—Back propagation algorithm, CUDA, GP-GPU, web usage mining.

I. INTRODUCTION

Web users feel comfortable if they reached the desired web page within the minimum navigation on a web site. A study of Users' recent behavior on the web will be useful to predict their desired target page. Generally Users' browsing patterns are stored in the web logs of a web server. These patterns are learned through the efficient algorithms to find the target page. Back propagation algorithm is implemented for learning the patterns. With learned knowledge various set of users' browsing patterns are tested. The results are observed and presented as an analysis on computational efforts of the algorithm. The analysis on the results proves the correctness of the algorithm. Still to improve the latency on learning, the algorithm is implemented with CUDA – MATLAB. Thus the BPA on GPU device leads to improved web usage mining than the numerous conventional methods.

A. *Literature Review*

Fran feinbube et al. [3] described the best practices of CPU– GPU algorithms. Jan Platos et al[4] described a document classification algorithm based on Particle Swarm Optimization with implementation of one and two GPUs. Mai Zheng et al. [5] proposed GMRace, a new mechanism for detecting races in GPU programs. GMRace combines static analysis with a carefully designed dynamic checker for logging and analyzing information at runtime. Pisit Makpaisit and Putchong Uthayopas [7] adopted the OpenACC standard for directive-based approach and proposed some extension to support GPU cluster programming. The extensions are constructs and clauses used to define the memory distribution and dependency of tasks on cluster nodes. The framework and technique used to implement a source-to-source compiler to support the proposed constructs and clauses. Youngsok Kim et al. [8] proposed a novel GPU architecture to enable high-performance memory-unaware GPU programming. ScaleGPU uses GPU memory as a cache of CPU memory to provide programmers a view of CPU memory-sized programming space. ScaleGPU also achieves high performance by minimizing the amount of CPU-GPU data transfers and by utilizing the GPU memory's high bandwidth. Zhiguang Xu, [9] addressed the challenges using a biologically inspired computational model that imitates the flocking behavior of social animals (e.g. birds) and implement it in the form of parallel programs on the Graphics Processing Unit (GPU) based platform of CUDA from NVIDIA™.

II. Problem Definition

The Users' browsing patterns are gathered from the web server and then extracts only the valid logs i,e., The logs that doesn't contain robots.txt, .jpg, .gif etc and unsuccessful request. These logs are codified with Meta data of the web site. Then the codified patterns are applied for preprocessing. The preprocessed data are fed to back propagation algorithm for training the usage patterns.

Machine learning theory based web usage mining assumes no statistical information about the web logs. This work falls under the category of supervised learning by employing two phase strategies such as a) Training phase b) Testing phase. In training phase, original logs are codified by simple substitution of unique page_id instead of page name for all the successful html requests and are interpolate by preprocessing into polynomial vector. The n dimensional patterns are inner- product to obtain 2 dimensional vectors which is trained by neural classifier to learn the nature of the logs. BPA takes the role of neural classifier in this work. By training the classifier for a specific users' logs a reasonably accurate suggestions can be derive. In testing phase, various users' logs are supplied to the trained classifier to decide which page-id is to be suggested. The flow charts of both phases are given in Figure1.a and Figure 1.b

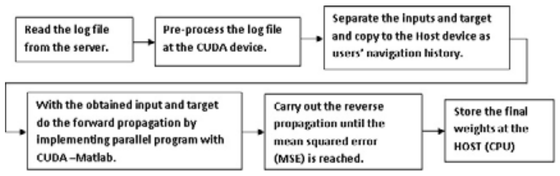
Read the log file from the server.
Pre-process the log file at the CUDA device.
Separate the inputs and target and copy to the Host device as users' navigation history.
With the obtained input and target do the forward propagation by implementing parallel program with CUDA –Matlab.
Carry out the reverse propagation until the mean squared error (MSE) is reached.
Store the final weights at the HOST (CPU)

Figure1.a Training Phase

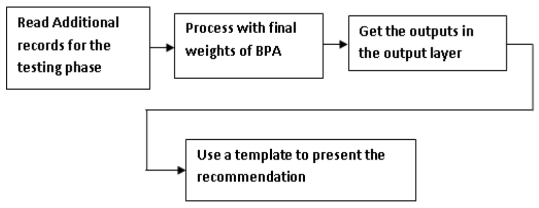
Read Additional records for the testing phase
Process with final weights of BPA
Get the outputs in the output layer
Use a template to present the recommendation

Figure1.b Testing Phase

III. IMPLEMENTATION

The simulation of personalization through web usage mining has been implemented using MATLAB 2013. Sample sets of logs are taken from ProtechSC's web server. These logs are filtered and codified. Users' 50 days patterns have been collected. 25 patterns have been used for training and the remaining patterns used for testing.

A. Filter the Log File

the web logs are collected from the web server of www.protechsc.net .

Sample web log file of this site is given in Fig.2

```
88.164.4.53 -- [25/Jul/2014:11:28:28 +0530] "GET /images/banner_webhosting.jpg HTTP/1.1" 200 35367 http://my-ptr.com/pages/ptcontest.php?blur=1&startpos=0 "Mozilla/5.0 (X11; U; Linux i686; fr; rv:1.9.2.6) Gecko/20100628 Ubuntu/10.04 (lucid) Firefox/3.6.6"
66.249.71.171 - - [25/Jul/2014:18:50:11 +530] "GET/robots.txtHTTP/1.1" 404- "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://google.com/bot.html)"
72.20.109.34 - - [25/Jul/2014:21:43:07 +0530] "GET/index.html HTTP/1.1" 200 23061 "-" "Mozilla/5.0 (compatible; GurujiBot/1.0; +http://www.guruji.com/en/WebmasterFAQ.html)"
72.30.79.32 - - [25/Jul/2014: 22:11:42 +0530] "GET/What_we_do.html HTTP/1.0" 200 20954 "-" "Mozilla/5.0 (Compatible; Yahoo! Slurp/3.0; http://help.yahoo.com/help/us/ysearch/slurp)"
72.20.109.34 - - [25/Jul/2014:23:01:41 +530] "GET/robots.txt HTTP/1.1" 404 "-" "Mozilla/5.0 (compatible; Googlebot/2.1; +http://google.com/bot.html)"
```

The Filtering Process as follows:

Step 1:Select the logs which don't contain Robots.txt and request of image files.

Step2: Group by IP address of the logs

Step3: Codify the requested page with following information

Step4: Store only IP address, visited page-id into database and make use of it for the polynomial preprocessing.

B.Back Propagation Algorithm

A neural network is constructed by highly interconnected processing units (nodes or neurons) which perform simple mathematical operations. Neural networks are characterized by their topologies, weight vectors and activation function which are used in the hidden layers and output layer. The topology refers to the number of hidden layers and connection between nodes in the hidden layers. The activation functions that can be used are sigmoid, hyperbolic tangent and sine. The network models can be static or dynamic. Static networks include single layer perceptrons and multilayer perceptrons. A perceptron or adaptive linear element (ADALINE) refers to a computing unit. This forms the basic building block for neural networks. The input to a perceptron is the summation of input pattern vectors by weight vectors. In most of the applications one hidden layer is sufficient. The activation function which is used to train the Artificial Neural Network is the sigmoid function.

1) Training

1. Read log files and filter it
2. Separate the data into inputs and target
3. Preprocess the data to any NL
4. Calculate Principal Component Vector by

$Z=Z*Z^T$

(1)

Where, Z denotes the cleaned logs

5. Train the BPA.

5.a Forward Propagation

(i) The weights of the network are initialized.

(ii) The inputs and outputs of a pattern are presented to the network

(iii) The output of each node in the successive layers is calculated.

$$O \text{ (output of a node)} = 1/(1+\exp(-W_{ij}X_i)) \quad (2)$$

(iv) The error of a pattern is calculated

$$E(p) = (1/2)\,(d(p)-o(p))^2 \quad (3)$$

5.b Backward Propagation

(i) The error for the nodes in the output layer is calculated

$$\delta(\text{output layer}) = o(1-o)(d-o) \quad (4)$$

(ii) Weights between output layer and hidden layer are updated.

$$W(n+1) = W(n) + \eta\delta(\text{output layer})\, o(\text{hidden layer}) \quad (5)$$

(iii) The error for the nodes in the hidden layer is calculated.

$$\delta(\text{hidden layer}) = o(1-o)\delta(\text{output layer})\ W \text{ (updated weights between hidden and output layer)} \quad (6)$$

(iv) The weights between hidden and input layer are updated

$$W(n+1) = W(n) + \eta\delta(\text{hidden layer})o(\text{input layer}) \quad (7)$$

The above steps complete one weight updating. Second pattern is presented and the above steps are followed for the second weight updating. When all the training patterns are presented, a cycle of iteration or epoch is completed. The errors of all the training patterns are calculated and displayed on the monitor as the mean squared error (MSE).

2) Testing

1. Read filtered logs and separate into inputs and target
2. Preprocess the data with a polynomial function
3. Process with final weights of BPA
4. Generate the suggestions from the output layer
5. Present the suggestions through templates

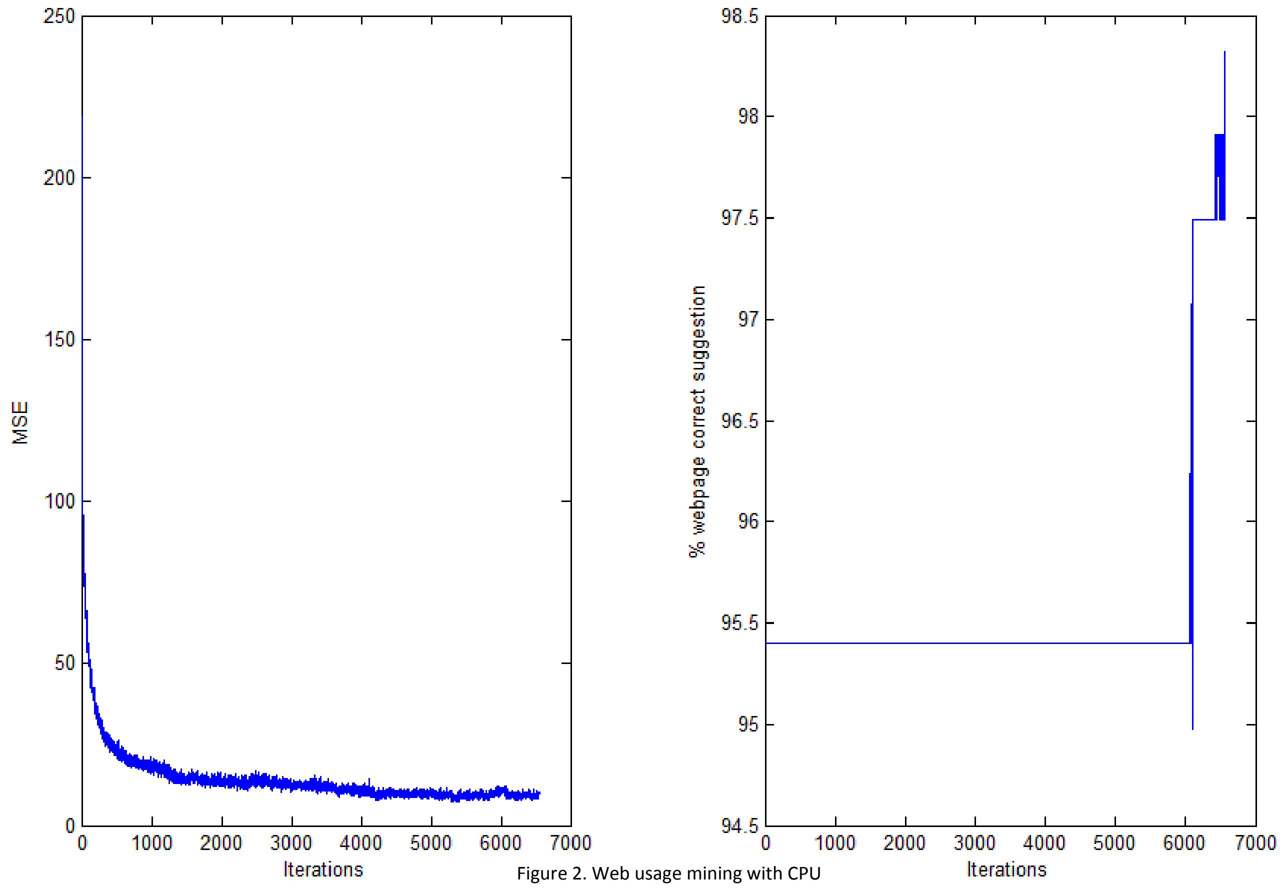

Figure 2. Web usage mining with CPU

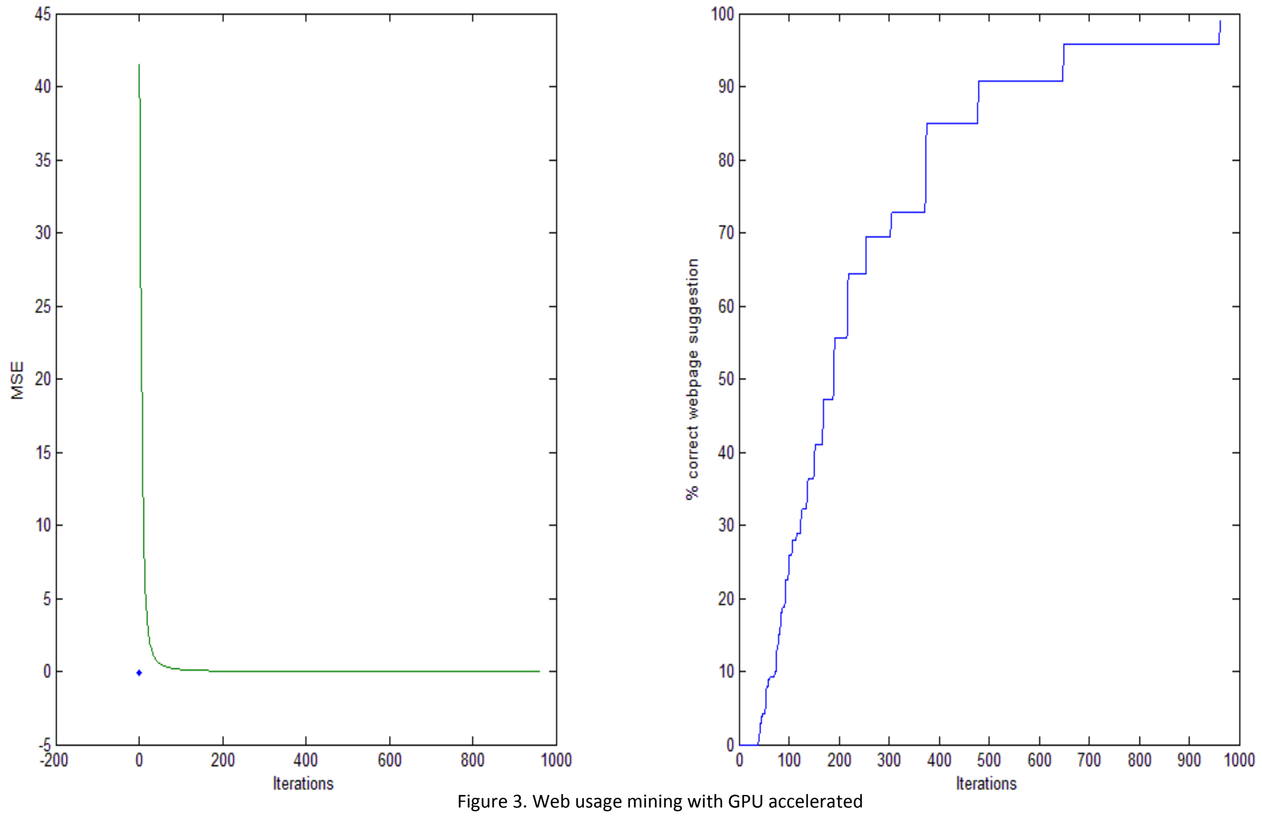

Figure 3. Web usage mining with GPU accelerated

To accelerate the above said process, these steps are executed between transferring data from host to device and GPU computation which can improve resource utilization rate through hiding overhead of loading data . So the whole procedures are shown as follow:

(1) Load log file data to CPU memory.

(2) Initialize GPU device.

(3) Create streams and initialize them

(4) Create events to monitor devices progress and record the exact execution time.

(5) Load n kernels, each kernel deals with specific data of itself (n slices data were processed by a kernel each time. A slice data serves as a cross session of the web server data cube).

(6) Load n times 'memcopy' asynchronously (n equals the number of streams).

IV. Experimental Results

The experimental results have shown that comparing to the CPU implementation GPU accelerated scheme improves the performance by about 70%. Figure 2 and Figure 3 are the result graph obtained by applying same logs into GPU accelerated and CPU implementation. The GPU accelerated scheme not only reduces the runtime overhead of web logs and also reduces the space overhead.

V. Conclusion

In this paper, GPU computing and the CUDA programming environment is implemented for an GPU-enabled web usage mining. This demonstrates the huge potential of GPU processing for web usage mining. If the source data is huge amount, GPU computing is the first candidate method; if the operation amount increase for the algorithm structure, accelerating by GPU achieves better effect. In future research, in order to improve the precision of our method, the reuse between thread blocks will be taken into account.

REFERENCES

[1] Brian Dushaw, Matlab and CUDA AT apl.washington.edu February 12, 2010

[2] D. B. Kirk and W. mei W. Hwu. Programming Massively Parallel Processors: A Handson Approach. Morgan Kaufmann Publishers, 2010.

[3] Frank Feinbube, Peter Tröger, and Andreas Polze, "Joint Forces: From Multithreaded Programming to GPU Computing", JANUARY/FEBRUARY 2011, IEEE SOFTWARE 51-57

[4] Jan Platos, Vaclav Snasel, Tomas Jezowicz, Pavel Kromer, Ajith Abraham, "A PSO-Based Document Classification Algorithm accelerated by the CUDA Platform" , 2012 IEEE International Conference on Systems, Man, and Cybernetics, October 14-17, 2012, COEX, Seoul, Korea.

[5] Mai Zheng, Vignesh T. Ravi, Feng Qin, "GMRace: Detecting Data Races in GPU Programs via a Low-Overhead Scheme" , IEEE , TRANSAONS ON PARALLEL AND DISTRIBUTED SYSTEMS, VOL. 25, NO. 1, JANUARY 2014, pp. 104 -115

[6] NVIDIA. NVIDIA CUDA Programming Guide, Second edition, August 2009.

[7] Pisit Makpaisit and Putchong Uthayopas, "The Framework and Compilation Techniques for Directive-based GPU Cluster Programming" 2014 11th international joint conference on computer science and software engineering, pp.229 – 235.

[8] Youngsok Kim, Jaewon Lee, Donggyu Kim, and Jangwoo Kim, "ScaleGPU: GPU Architecture for Memory-Unaware GPU Programming", Published by the IEEE Computer Society,2013, pp.1556- DOI 10.1109/L-CA.2013.19 1556-6056/13.

[9] Zhiguang Xu, "The Flocking Based and GPU Accelerated Internet Traffic Classification",Proceedings of the 2014 International Conference on Mathematical Methods, Mathematical Models and Simulation in Science and Engineering, pp.88-93, ISBN: 978-1-61804-219-4 .