International Journal of Computer Applications (0975 – 8887)
Volume 74– No.17, July 2013

Serial and Parallel Bayesian Spam Filtering using Aho-Corasick and PFAC

Saima Haseeb
TIEIT
Bhopal

Mahak Motwani
TIEIT
Bhopal

Amit Saxena
TIEIT
Bhopal

ABSTRACT

With the rapid growth of Internet, E-mail, with its convenient and efficient characteristics, has become an important means of communication in people's life. It reduces the cost of communication. It comes with Spam. Spam emails, also known as 'junk e-mails', are unsolicited one's sent in bulk with hidden or forged identity of the sender, address, and header information. It is vital to pursue more effective spam filtering approaches to maintain normal operations of e-mail systems and to protect the interests of email users. In this paper we developed a Spam filter based on Bayesian filtering method using Aho-corasick and PFAC string matching algorithm. This filter developed an improved version of spam filter based on traditional Bayesian spam filtering to improve spam filtering efficiency, and to reduce chances of misjudgement of malignant spam. For further improvement of Spam filtering process we are transform the filter in to parallel spam filter on GPGPU's by using PFAC Algorithm.

Keywords

Spam Filter, Bayesian Spam Filter, Aho-Corasick, PFAC.

1. INTRODUCTION

With the growing use of electronic mail the problem of having spam becomes the major issue in today's concern [1]. Spams are the unwanted emails which floods the internet with many copies of the same message. Sometimes spam carries malicious content that harm our system and degrades the performance [2]. It becomes a need to design a filter which is capable of handling different variety of spams and reduces false positives. Millions of email goes through the servers. This increases the demand of using email spam filter on server which is fast enough to compensate the receiving rate of spam. Further it should not miss a little bit of spam otherwise it will be costly for the receiver because there is a chance of spam being opened and activated and affects security. Therefore the filter must have high accuracy rate [3].

The proliferation of spam occupies a large number of mail server storage spaces and violates privacy of the recipients [4]. Spam not only costs recipients time to deal with, more importantly, the harm of spam's unhealthy content, including pornography and violence, is difficult to estimate and measure[5].

Various techniques have been used to design spam filters. Some of them are list based filters and content based filters [6]. List based filters check the mails on the basis of their servers. There is a predefined list of all the servers which distinguishes spammers with legitimate server. The mails from spammers are rejected otherwise accepted. Blacklist filters [7], Real time Black hole list filters and White list filters, Grey list filters comes in this category. Content based filters evaluate words or phrases found in each individual message to determine whether an email is spam or legitimate. Some content based filters are word based filters, heuristic filters and Bayesian filters.

Bayesian filters, considered the most advanced form of content-based filtering, employ the laws of mathematical probability to determine which messages are legitimate and which are spams. In order for a Bayesian filter to effectively block spam, the end user must initially "train" it by manually flagging each message as either junk or legitimate. Over time, the filter takes words and phrases found in legitimate emails and adds them to a list; it does the same with terms found in spam.

In this paper we design a Bayesian sequential and parallel spam filter based on Enron data set and its parallel version is based on PFAC algorithm.

2. RELATED WORK

Blacklist spam filters created a list of all the addresses and IP that have been previously used for sending spam. When a mail arrives it checks the mail against the list if it is from the listed sender it is rejected otherwise accepted [4].Real time black hole list filters is also based on same concept except there is a involvement of third party which creates the list of spam senders for the organization. It reduces the burden of IT staff. Third party receives the mail, checks it against the list and decides accordingly [5]. White list works exactly opposite to that of a blacklist. Instead of creating a list of spammers, a list of legitimate senders is created [4].Grey list is based on the fact that many spammers send bulk of email once. If any bulk of email reaches server it rejects it and reports error message to sender. If it is attempted to send twice it is considered as the legitimate mails and its address is added to the list of legitimate senders created by grey list filter. List based filters may misidentify legitimate senders as spammer [4, 5].

Word based filter is content based filter in which a list of spam words is created. If the receiving mail contains the blocked words it is reported as the spam but there may be a chance that spammer misspells certain words to pretend its spam as a legitimate mail. It increases the burden of updating blocked word database regularly.

Heuristic filtering is again a concept based on content based filtering. It created a list of suspected words with its heuristic count. Whenever any new message arrives it scans the content of message for the list and calculates the total heuristic count of all the keywords if it is greater than the current count it is considered as a spam otherwise ham [6].

Bayesian filter is probability based filtering technique. It learns from spam as well as good mails. At the initial stage filter is trained by calculating spam probability of known spam and ham keywords. Later this list is used to calculate total spamicity of testing mail. If the spamicity is found greater than or equal to threshold value it is rejected as spam [6, 7]. Content based filters are considered as the most efficient filters as it checks the content of message and can easily identify spams sending through legitimate users also[8].

Content based filters use Aho-Corasick algorithm to calculate count of each pattern in test mail. Aho-Corasick is the multipatterns string matching algorithm which locates all the occurrence of set of keywords in a text of string. It first creates deterministic finite automata for all the predefined keywords and then by using automaton, it processes a text in a single pass. Aho-Corasick works in two phases: preprocessing phase and searching phase as shown in figure 1 and 2.

Preprocessing phase constructs finite state automata for the set of predefined keywords (or keyword tree) which are supposed to be found in the text string. After constructing automata, failure function of each node is calculated. Failure function of a node is defined as the longest suffix of the string that is also the prefix of some node. Output function for final states has to be calculated. Searching phase proceeds with scanning the testing mail using automata build in previous phase and reports the count of each keywords [9,10,11].

Aho-Corasick is previously applied in many areas of networks and computer security, and bioinformatics. These networks and bioinformatics applications are computationally demanding and require high speed parallel processing. To speed up the performance of Aho-Corasick algorithm, a parallel version of Aho-Corasick PFAC (parallel failure less Aho-Corasick) is developed [12,13,14].

PFAC uses the concept of GPGPU to fix the occurrence of keywords in a string. In preprocessing phase PFAC build DFA with no back track links. No failure function is calculated for the DFA. Suppose we have 3 patterns [HER, IRIS, IS]. The PFAC DFA for the patterns without back track lines is built as:

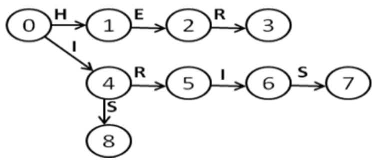
$$\begin{tikzcd}
0 \arrow[r, "H"] \arrow[dr, "I"'] & 1 \arrow[r, "E"] & 2 \arrow[r, "R"] & 3 & \\
 & 4 \arrow[r, "R"] \arrow[d, "S"] & 5 \arrow[r, "I"] & 6 \arrow[r, "S"] & 7 \\
 & 8 & & &
\end{tikzcd}$$

Fig 1: pre processing phase

DFA is placed in global memory from where it is going to accessed by each thread to take the copy of DFA. The concept of allocating DFA to each thread increases the efficiency of Aho-Corasick algorithm. In searching phase, each alphabet of text is assigned to each thread and total no. of thread is equal to text length. Supposed the text to be scanned is IRISTHER.

Each thread accesses the copy of automata from global memory and processes its alphabet if a valid transition found, it proceeds otherwise terminates itself.

Thread 0 searches the automata for alphabet I it gets valid transition. After taking the input "IRIS", Thread 0 reaches state 7, which indicates pattern "IRIS", is matched. Thread 1 starts with scanning alphabet R, no transition is found for R so it terminates at state 0. Thread 2 gets transition for I, after taking input "IS", Thread 2 reaches state 8, which indicates pattern "IS" is matched, no transition is found for T, terminates in state 8. Thread 3 and 4 found no transition for S and T, terminate early at state 0. Thread 5 found transition 1 for H, after taking input "HER" reaches state 3, pattern "HER" is matched, terminates in state 3. Thread 6 and 7 found no transition for E and R, terminates at state 0.

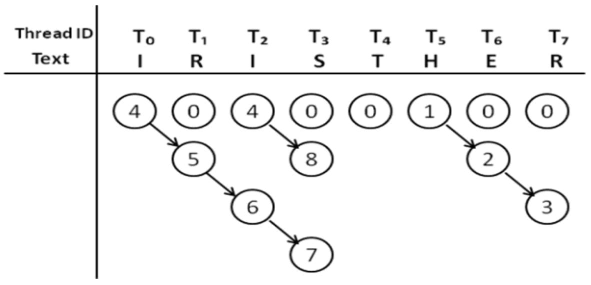

Fig. 2: Searching Phase

GPGPU is the concept used to boost many applications in real world [15,16].GPGPU is the use of GPU for general purpose computation[17,18]. To use GPU for general computation it must be programmed by using parallel programming language like Cuda and OpenCL [19, 20].GPU typically handles computation for computer graphics.

GPU was originated in the late 1990s as the coprocessor for accelerating the simulation and visualization of 3D images. From 2006 GPU have developed to be more flexible and even considered for GPGPU. In today scenario the high performance of applications is mandatory; in order to meet that requirement GPU comprises parallization of many applications to hike their performance[21,22,23].

A direct implementation of parallel computation on GPUs is to divide an input stream into multiple segments, each of which is processed by a parallel thread for string matching [24,25].

3. PROPOSED ALGORITHM

We design a parallel spam filter using GPGPU (general purpose computation on GPU).For this purpose we design serial spam filter and parallelize this approach to make it parallel spam filter. To design a filter we use the Bayesian approach. It works in two phases: training phase and filtering phase. In training phase, it creates 3 databases, Database for keywords taken from ham and spam mails, Database for ham mails, Database for spam mails. After creating database, it calculates spam probability of every keyword by using Bayesian statistics.

Bayesian statistics tell us that if a word "connect" appears in 35 of 1000 ham mails and in 750 of 1000 spam mails. Then the presence of word "connect" means that the given message has 95.54%chance of being spam. Spam probability of "content"=750(750 + 35) =95.54 %. This phase creates a file containing list of keywords with their corresponding probability which is later used in filtering phase. Filtering

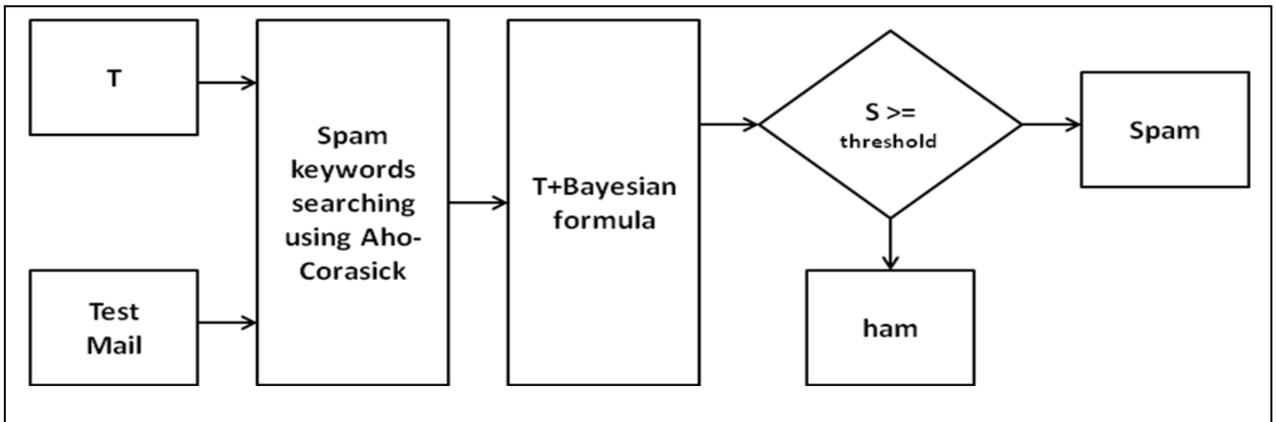

Fig. 3: Serial Spam Filter

phase takes the file created in training phase and testing mail as input and check whether the mail is spam or ham mail. Spamicity of mail is calculated by using the formula Spamicity = p1*p2*p3……Pm / [(p1*p2*…pm) + ((1-p1) *(1-p2)*… (1-pm))].

If the spamicity comes out to be greater than or equal to threshold the mail is reported as spam. Filtering process uses Aho-corasick, a multipattern string matching algorithm, to calculate count of each keyword in testing mail. This count is used to calculate spamicity of mail. Overall diagram for serial spam filter is represented in figure 3 and 4. Here T is training Bayesian probability data of keywords.

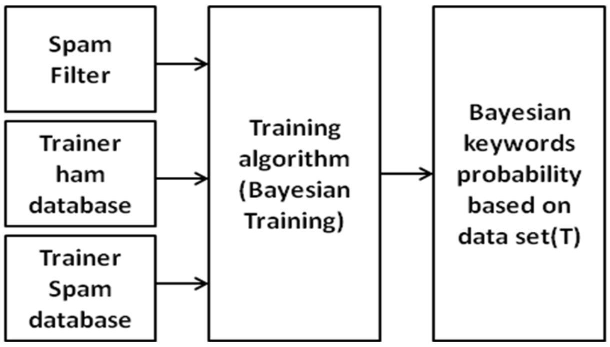
Spam Filter
Trainer ham database
Trainer Spam database
Training algorithm (Bayesian Training)
Bayesian keywords probability based on data set(T)

Fig. 3: Serial Spam Filter

The project is divided into two modules: Filter trainer and Filter. In filter trainer module, three databases are created: spam keywords, trainer ham database, and trainer spam database. These databases are passed to training algorithm i.e. Bayesian training. Training algorithm calculates spam probability of all keywords based on training data set and named it T.

In second module, Aho-Corasick algorithm is used which takes spam keywords and test mail as inputs and provides count of all spam keywords in test mail.

3.1 Training Algorithm for serial spam filter

1. First we created a list of Spam Keywords and search them in ham and spam database.

2. Spam probability of all keywords is calculated by using Bayesian statistics

3. We create a list having name of keywords and their corresponding probability and save this file being used in second module.

3.2 Filtering Algorithm for serial spam filter

1. First we fetch the file produced in first module having name of keywords and their corresponding probability.

2. Fetch mail for which we want to know that is spam or ham

3. Scan the mail by using Aho-Corasick algorithm and calculate frequency count of Spam Keywords in mail.

4. Calculate Spamicity of the mail by using Bayes theorem:
Spamicity = (p1*p2*p3……Pm) / [(p1*p2*…pm) + ((1-p1) *(1-p2)*… (1-pm))].

5. Compare Spamicity of mail with Threshold value (which is set by reverse Engineering).

6. If Spamicity is greater than Threshold then mail is Spam otherwise It is Ham.

Parallel spam filter uses PFAC approach. The filter trainer module of parallel spam filter is same as in serial spam filter.

In second module, PFAC algorithm is used which takes spam keywords and test mail as inputs. Each thread is assigned to each alphabet of test mail and report count of keywords. These counts and Bayesian formula are used to calculate spamicity of mail. If it comes out to be greater than or equal to threshold value it is reported as spam otherwise ham. The overall diagram is shown in Figure 5 and 6.

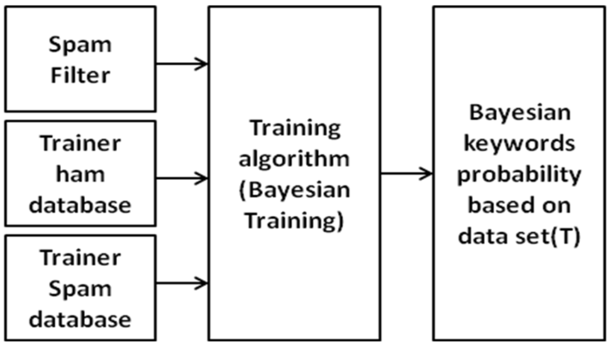
Spam Filter
Trainer ham database
Trainer Spam database
Training algorithm (Bayesian Training)
Bayesian keywords probability based on data set(T)

Fig. 5: Overall diagram for parallel spam filter

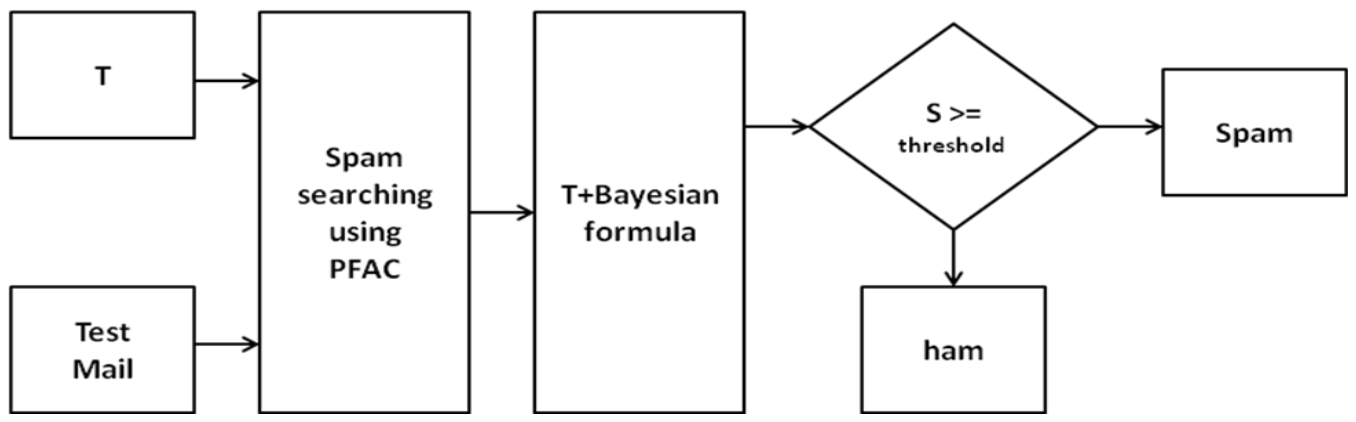

Fig. 6: Parallel Spam Filter

3.3 Training Algorithm for parallel spam filter

1. First we created a list of Spam Keywords and search them in ham and spam database.

2. Spam probability of all keywords is calculated by using Bayesian statistics.

3. Then we create a list having name of keywords and their corresponding probability and save this file being used in second module.

3.4 Filtering Algorithm for parallel spam filter

1. First we fetch the file produced in first module having name of keywords and their corresponding probability

2. Then we fetch mail for which we wish to know that it is spam or ham.

3. Scan the mail by using PFAC algorithm and calculate frequency count of Spam Keywords in mail.

4. Calculate Spamicity of the mail by using Bayes theorem:
Spamicity = p1*p2*p3……Pm / [(p1*p2*…pm) ((1-p1) *(1-p2)*… (1-pm))].

5. Compare Spamicity of mail with Threshold value (which is set by reverse Engineering).

6. If spamicity is greater than Threshold then mail is Spam otherwise It is Ham.

3.5 Calculation of Threshold by Reverse Engineering

First we scan all spam mails of Enron Data Set and calculate spamicity of each mail with help of second module. Set minimum value of Spamicity of these mails as Threshold.

4. COMPARITIVE ANALYSIS

Comparative analysis between Serial Spam Filter and Parallel Spam filter is shown in table 1 and figure 7.

4.1 Experimental Environment

Processor: Core i3
RAM: 4 GB
OS: Windows 7
Language: Visual C++ runs on Visual Studios 2008
GPGPU: AMD Radeon HD 6800 series
Language (parallel implementation): OpenCL

4.2 Experimental Data

No. of test mails size is 1000, 2000, 5000 and 10000.

4.3 Experimental Results

Table 1: Execution time for serial and parallel spam filter

S.No.	No. of Test Mail	Parallel Filter Speed	Serial Filter Speed
1.	1000	0.244 sec.	129 sec.
2.	2000	0.38 sec.	198 sec.
3.	5000	0.64 sec.	302 sec.
4.	10000	1.31 sec.	601 sec.

Graphical representation of this experimental results is shown in figure 7

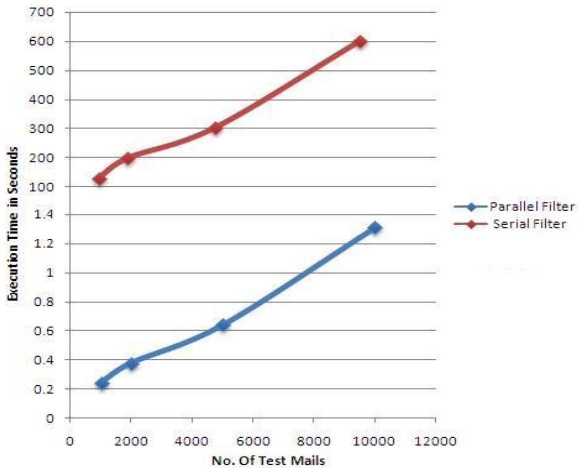

Fig. 7 Experimental Results

Figure 7 analyze the empirical comparison between serial execution and parallel execution. Parallel execution takes less than 2 seconds to execute larger data sets where as for same data set serial execution takes more than 10 minutes. Figure 7 explain that parallel execution is much efficient than serial execution. Two different lines represented the execution time in above given graph. One represents parallel execution and another one represents serial execution. Looking into graph we will find that parallel execution is much efficient than serial execution.

Table 2 represents accuracy for serial spam filter and parallel spam filter. Accuracy for both versions will be same. As the no. of mails increase accuracy will decrease and after a point it will be constant.

Table 2 Accuracy calculation

S.No.	No. of Test Mails	Accuracy
1.	1000	76 %
2.	2000	72 %
3.	3000	72. 5%
4.	4000	71.4 %
5.	5000	70.9 %
6.	6000	70.2 %
7.	7000	70.1 %
8.	8000	70.2%
9.	9000	70.1%
10.	10000	70.1%

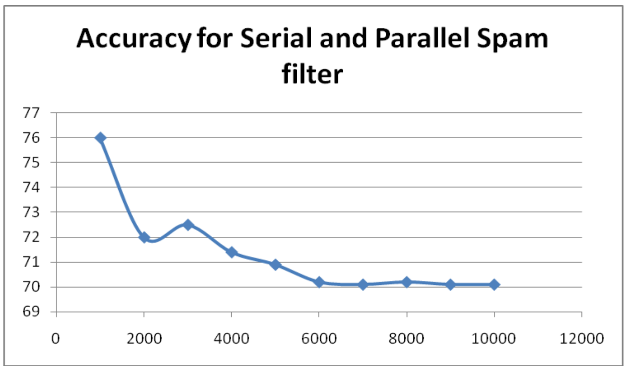

Fig. 8: accuracy calculation

Spam filter’s accuracy rate is approximately 70%. For training we have taken Enron data sets. Keywords are limited. If we will increase training data sets and keywords than results will be more accurate and efficient.

5. CONCLUSION

In order to filter the emails Bayesian spam filter is an adequate spam filter. It is more advanced form of content based filtering. To ameliorate efficiency of Bayesian spam filter we have implemented it parallel on GPGPU with Parallel Failure-less Aho-Corasick technique. Parallel spam filter is efficient on larger data sets and processing time is much better than serial spam filter. Accuracy of our spam filter is approximately 70%.

6. REFERENCES

[1] Wu, Y. L., “Using Visual Features For Anti-Spam Filtering, ”2005 IEEE International Conference on Image Processing (ICIP2005), pp. 509–512, 2005. Postini : Email Monitoring + Email Filtering Blog.http://www.dicontas.co.uk/blog/quick-facts/emailspam-trafficrockets/65/.

[2] Toshihiro Tabata, “SPAM mail filtering : commentary of Bayesian filter, ” The journal of Information Science and Technology Association, Vol.56, No.10, pp.464-468, 2006.

[3] http://www.cs.nmt.edu/~janbob/SPAM, Spam corpus, SMS corpus,

[4] http://www.comp.nus.edu.sg/~rpnlpir/downloads/corpora/smsCorpus/

[5] Amayri O, Bouguil N (2009). Online Spam Filtering Using Support Vector Machines.IEEE., pp. 337- 340.

[6] C. Pu, S. Webb, O. Kolesnikov, W. Lee, and R. Lipton. Towards the Integration of Diverse Spam Filtering Techniques. In Proc. of IEEE International Conference on Granular Computing, pages 7 – 10, 2006.

[7] I. Androutsopoulos and et., “An Evaluation of Naïve Bayesian Anti-Spam Filtering”, 11th EurpoeanConference on Machine Learning, pp 9-17, Barcelona, Spain, June 2000

[8] Paul Graham, “Better Bayesian Filter” ,http://www.paulgraham.com/better.htm

[9] A.V. Aho and M. J. Corasick, “Efficient String Matching: An aid Bibliographic search”. In Communication of the ACM Vol. 18, issues 6, pp.-333-340, 1975.

[10] Cheng-Hung Lin and Shih-Chieh-Chang," Efficient pattern matching algorithm for memory architecture", Vol. 19, issue 1, pp. 33-41, January 2011.

[11] Chengguo Chang and Hui Wang," Comparison of Two-Dimensional String Matching Algorithms"In the proc. International Conference on Computer Science and Electronics Engineering (ICCSEE), Vol. 3, pp. 608-611,march 2012.

[12] Raphael Clifford, Markus Jalsenius, Ely Porat and Benjamin Sach,"Pattern matching in multiple stream", in the proc. 23rd Annual conference on Combinatorial Pattern Matching, pp. 97-109,2012.

[13] R. Takahashi, U. Inoue, "Parallel Text Matching Using GPGPU", in the proc. 13th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel & Distributed Computing (SNPD), pp. 242-246, Aug. 2012.

[14] C. Lin, et al., "Accelerating String Matching Using Multi-Threaded Algorithm on GPU," Proc. IEEE Global Telecommunications Conf., pp. 1-5, 2010.

[15] J. D. Owens, et al., "A Survey of General-Purpose Computation on Graphics Hardware," Computer Graphics forum, Vol. 26, No. 1, pp.80-113, 2007.

[16] C. Lin, C. Liu, L. Chien, and S. Chang," Accelerating Pattern Matching Using a Novel Parallel Algorithm on GPUs", IEEE Transactions on computers, vol. pp, issue 1.

[17] ZhaXinyan and S. Sahni," Multipattern string matching on a GPU", In the proc. IEEE conference on Computers and Communications (ISCC), pp. 277-282, July 2011.

[18] Tran Nhat-Phuong, Lee Myungho, Hong Sugwon and Minho Shin," Memory Efficient Parallelization for Aho-Corasick Algorithm on a GPU", IEEE 14th International Conference on High Performance Computing and Communication, pp. 432-438, June 2012.

[19] Jungwon Kim, Honggyu Kim, Joo Hwan Lee and Jaejin Lee," Achieving a single compute device image in OpenCL for multiple GPUs", Proceedings of the 16th ACM symposium on Principles and practice of parallel programming, pp. 277-288,2011.

[20] NVIDIA, "CUDA Best Practices Guide: NVIDIA CUDA C Programming Best Practices Guide – CUDA Toolkit 4.0", May, 2011

[21] Xinyan Zha and Sartaj Sahni," GPU-to-GPU and Host-to-Host Multipattern String Matching on a GPU", IEEE Transactions on Computers, Volume 62, Issue 6, pp. 1156-1169,2013

[22] J.E. Stone, D.Gohara, and G.Shi, "OpenCl: A parallel programming standard for heterogeneous computing systems, "Computing in Science Engineering,vol. 12,no.3,pp.66-73,2010.

[23] HyeranJeon, Xia Yinglong and V.K. Prasanna," Parallel Exact Inference on a CPU-GPGPU Heterogeneous System", In the proc. 39th International Conference on parallel Processing (ICPP), pp. 61-70,Sept. 2010.

[24] Liang Hu, CheXilong and XieZhenzhen,"GPGPU cloud: A paradigm for general purpose computing", Tsinghua Science and Technology, Vol. 18, issue 1, pp. 22-23, Feb. 2013.

[25] M. C. Schatz and C. Trapnell, "Fast Exact String Matching on the GPU," Technical report