Accelerating Enhanced Boyer-Moore String Matching Algorithm on Multicore GPU for Network Security

Manjit Jaiswal
Department of Computer Science & Engineering
Guru Ghasidas Central University, Bilaspur, India, 495001

ABSTRACT

Graphics Processing Units (GPUs) were developed for graphics processing and it was not highly-parallel. But to overcome this problem developed General Purpose Computing on GPU, this is known as GPGPU.Boyer-Moore exact string matching algorithm are heavily used in the application of antivirus engines, DNA sequencing, text editors, intrusion detection etc. In this environment, the GPU was highly-parallel, multithreaded. In this Paper extend the GPU application into other area such as string matching problem. This paper shows the results on adapting the enhanced Boyer-Moore (EBM) string matching algorithm to run on GPU paradigm and comparison with serial version and multithreaded version on CPU.The experimental results demonstrate that GPU version of enhanced Boyer-Moore (EBM) string matching algorithm 10 times faster than CPU version and 9 times faster than the multithreaded version. It can be also see there that multithreaded version of EBM algorithm about 12% to 13% peak performance than serial version of EBM.Speedup of EBM algorithm is grow and 12x to 13x than serial one.

General Terms

Pattern Recognition, Security, Algorithms etc.

Keywords

EBM, GPU, Multithreaded, Parallel, BM, GPGPU, AMD, NVIDIA CUDA, OpenCL.

1. INTRODUCTION

Open Computing Language (OpenCL) is a framework for writing programs that execute across heterogeneous platforms consisting of central processing unit (CPUs), graphics processing unit (GPUs),and other processors such as DSP and Cell BE processors. OpenCL includes a language for writing kernels (functions that execute on OpenCL devices), plus application programming interfaces (APIs) that are used to define and then control the platforms. OpenCL provides parallel computing using task-based and data-based parallelism. Kernels are function within the OpenCL that are executed on the devices (GPU) of the AMD, NVIDIA and Intel. As article [3] there have string matching implementation on GPU using CUDA .Within the kernel there are thousands of threads executed in parallel. Unlike CPUs that are optimized for use on sequential code, all commodity GPUs follow a streaming, data parallel programming model resembling SPMD (single-program multiple-data).In string matching algorithm to find out the location of one or many pattern in the text string. The Boyer-Moore string searching algorithm was developed in the mid-1970s as in [17] by Bob Boyer and J. Strother Moore. It is an efficient string searching algorithm that does not need to search every character in a text string records due to the use of MovDist value preprocessing table based on the search pattern. While use of the preprocessing table the searching Boyer-Moore algorithm skip the maximum number of characters as possible in the text record. NVIDIA corporation provides the flexible programming language CUDA (the Compute Unified Device Architecture) on GPU-based application developing [22].This paper shows the parallelization of the enhanced Boyer-Moore algorithm with the OpenCL software development environment on the GPU (AMD) and compares with serial and multithreaded version on CPU.Algorithms attempt to find the location of one or several strings (search patterns) in a string or text (search record). This paper is shows that accelerating the enhanced Boyer-Moore (EBM) algorithm by using the GPUs. Rest of the paper is organized as follows. Section two briefly describes the prior work in GPU based string matching algorithm. Section three describes the implementation and performance of string matching Boyer-Moore algorithm on the GPU.The fourth section carries out the various results of BM algorithm on GPU, multithreaded and serial on CPU after then shows the enhance performance of the EBM algorithm. Eventually section five draws the conclusion and future direction of this work.

2. RELATED WORK

String matching is an important problem in text processing and is commonly used to locate the appearance of one dimensional arrays (the so-called pattern) in an array of equal or larger size (the so-called text).The string matching problem can be defined as: let Σ be an alphabet, given a text array T[n] and a pattern array P[m], report all locations [i] in T where there is an occurrence of P, i.e. [i + k] =P[k] for $m \leq n$. Jacob and Brodley were the first that tried to use the GPU as a pattern matching engine for NIDS [21].Scalpel uses the Boyer-Moore single pattern search algorithm as in [20]. Boyer-Moore [14], BMH [13], and BMHS [11] have developed pattern matching algorithms Central to these algorithms is a bad character function for *P* that specifies how many characters to shift *P* right before reexamining pairs of characters from *P* and *S* for a match. Serial version of enhanced BM algorithm shows that how to performance calculates by BM algorithm. The program’s string-matching kernel executes parallelized searching of string matching algorithms using global memory. The parallel implementations were presented of the Naive, Knuth-Morris-Pratt, Boyer-Moore-Horspool and the Quick-Search on-line exact string matching algorithms using the CUDA toolkit as in [3].

3. IMPLEMENTATION AND PERFORMANCE

3.1 Parallel Enhanced Boyer-Moore algorithm (EBM)

EBM algorithm based on single-string matching BM algorithm as in [5]-[12]. The pseudo code of parallel Enhanced BM algorithm is given below in Fig 1.

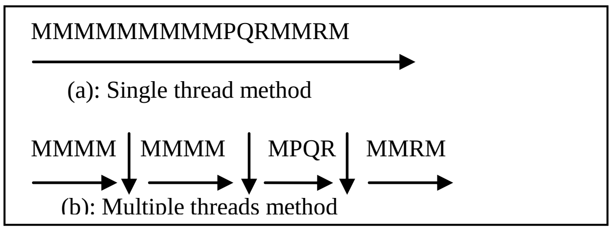
```
        EBM (P, T)

1.   Preprocessing phase //execution on CPU
2.   Index[c]←——— refers to 2.
3.   End of preprocessing phase
4.    Kernel function EBM ( T , P, Index[c] )
     //execution on GPU
5.   While(i<n){
6.   if(Index[T[i]]]==-1)
7.   i++}
8.   else
9.   break
10.   i=i+m-1
11.  j=j-1
12.  do{
13.  while(T[i]==P[j]){
14.   i=i-1
15.   j=j-1}
16.  While(i<n-1){
17.   If(Index[T[i+1]==-1]
18.  i++}
19.  i=i+m-Min (j, 1+Index [T[i]])
20.  j=m-1;
21.  }while(i<n)
22.  End of do-while loop
23.  End of the kernel function.
```

Fig 1: GPU based EBM algorithm

Before pattern matching; there is a preprocessing phase which is calculated only once time for further use by kernel source and it creates an index table with entry according to refer to 2.Preprocessing phase have time complexity O (m+∑).

3.2 Problem of Direct implementation of Boyer-Moore algorithm on GPU

In this paper, we study the use of parallel computation on GPUs for accelerating string matching. A direct implementation of Boyer-Moore 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 .But in this method it we cannot detect the boundary condition i.e. we cannot find out occurring of the pattern on the boundary of the adjacent segments. For example in the Fig. 2 assume that using a single thread to find out pattern PQR in the text takes 16 cycles as [1]. If we divide the single thread into four threads and allocate each segment a thread to find out the same pattern simultaneously then four threads takes only four cycle to detect same pattern.



However in this method there it cannot detect the occurring in the boundary of adjacent segment. For example in Fig. 3 shows such type of problem. The pattern PQR appeared in the segment 3 and 4.In this condition thread 2 and 3 cannot find out pattern. To overcome these problems as in fig.4 with any despite the boundary detection problem can be resolve by using refer to (1).We divide the text record into threads such as that number of character in the each threads stimulated by refer to (1).

$$NCT=n_t/t_n + (m_p\text{-}1)$$

Where

NCT=total number of character in the each threads

$n_{t=}$text length

m_p= pattern length

$t_{n\,=}$ number of threads

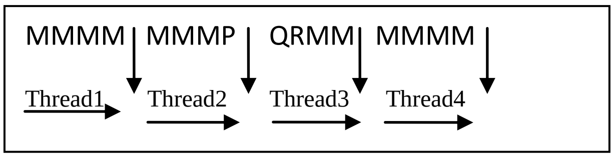


Fig: 3 Boundary detection problems

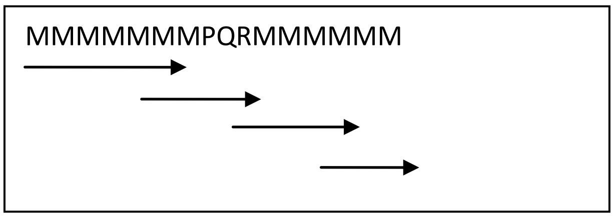


Fig:4 Solution of Boundary detection problem

In this paper it proposed a parallel enhanced Boyer-Moore algorithm (EBM) which is accelerated the string matching problem.OpenCl SDK uses to port the enhanced Boyer-Moore string searching algorithm to execute in parallel on the GPU.

The GPU is used to implement the Boyer-Moore in the following manner:

1. Pre-compute Boyer-Moore index value table on the CPU for the search pattern. All the character is initialized with -1 and that type of character which is appeared in the pattern then initialized with maximum index value of the pattern by refer to 2.

$$\text{Index}[c] = \begin{cases} \text{J, if j of c in the pattern P is the last occurred position value where } 0\leq j\leq m\text{-}1 \\ \text{- 1, otherwise} \end{cases} \qquad (2)$$

2. Transfer the index value table from CPU to GPU.
3. Transfer the pattern string from CPU to GPU global memory.
4. Transfer the text record from the CPU to GPU global memory.

5. Run the enhanced Boyer-Moore algorithm for search the pattern string on the OpenCL kernel GPU global memory for all the text record.
6. Transfer the result of the OpenCL kernel from GPU to CPU.The results includes the total number of occurrences of the pattern in the text otherwise pattern did not find.
7. The moving distances (MovDist) value find out by refer to 3.

   MovDist= {i+m-MIN (j, 1+Index [T[i]])} (3)
   Where m=pattern length
   j=m-1
   i=n-1
   T[i] = text record

It is calculated by multiple threads simultaneously. The above steps number 1, 2, 3, 4 are initialized steps and execute once for particular pattern. Step 5 is executed in SIMD manner depending upon text data. Step 6 is transfer the results from GPU to CPU.Maximum number of thread in each block that will be execute simultaneously by the GPU depending upon requirement for different data file.

4. EXPERIMENT RESULTS AND COMPARISONS

4.1 Environment of Hardware

Serial, multithreaded and parallel platform are show below in the table 1.The hardware specification of the serial and multithreaded show in the table 1.

Table 1. Hardware specification of CPU

CPU version	Intel core™ i3
Frequency of CPU core	3 GHz
Max power consumption	73 Watt
Total RAM	4GB
OS	32-bit Window 7
Number of core in Intel i3	2 core and 4 threads

The hardware specification of the parallel platform show in the table 2.

Table 2. Hardware specification of parallel platform

GPU version	AMD 6850
RAM of CPU	4GB DDR3
CPU version	Intel core™ i3
Engine frequency	775MHz
Frequency of CPU core	3 GHz
Max power consumption	500 watt
Memory of GPU	1 GB 256 bit GDDR5
Core frequency	1000 MHz
Number of cores in GPU	192
OS	32-bit window 7

4.2 Software

The following software installed on the system device. The OpenCL SDK version 4.1 installed. The OpenCL toolkit includes sample application that demonstrates various features of the OpenCL programming environment.

4.3 GPU vs. CPU vs. Multithreaded on CPU

The experiments contain three parts, serial string-matching, multithreaded string matching and parallel string-matching. We take 50 MB, 100 MB, 150 MB, 200 MB, and 250 MB text string and 4 byte and 8 byte of pattern string. We Compare the GPU version, serial version and multithreaded version on CPU of enhanced BM algorithm show in the table 3 and 4. We can see there that GPU based enhanced BM (EBM) algorithm is approximately 10 times peak than CPU version and 9 times peak performance than multithreaded CPU version. But multithreaded version of CPU takes less times and about peak performance than serial version. Results are show below in the table 3.We can also compares speedup and see there that GPU based enhanced BM algorithm almost 12x for 4 byte pattern and almost 14x for 8 byte search pattern. Although different type of memory has different accessing speed and size. We put the text and pattern into the global memory and find out the pattern into text. The speedup results are also show in the table 3 and 4 for different pattern size and text records. Speedup is defined by the following formula refer to 4:

$$S_{P} == \frac{T_1}{T_2} \qquad (4)$$

Where,

- *p* is the number of processors.
- $_{T1}$ is the execution time of the sequential algorithm.
- $_{Tp}$ is the execution time of the parallel algorithm with p processors.

Table 3 and Fig. 4 shows consequently the result of the data record with 24% of the text records containing matches for the search patter and their resulting graph. Similarly table 4 shows the result of the data record with 35% of the text records containing matches for the search pattern and Fig 5 shows their graph of execution time. The results shown in Tables 3 and 4 only include the times for executing the searches on both the GPU and CPU. These numbers do not include any time required for preprocessing Index table calculations and data transfers to and from the GPU.As a result of Speedup when run the EBM algorithm successfully on the GPU of AMD 6850 and calculate the speedup by formula refer to 4 then finally it gets the almost 13x almost speedup on the GPU of AMD 6850 and this will be change go high when use many more core type of the GPU.

Table 3. Comparisons of performance with 24% of Records Matching

Text Records	Pattern size	GPU(s)	Serial on CPU(s)	Multithreaded on CPU(s)	No of occurrences of pattern
50MB	4 bytes	0.007	0.070	0.062	2355994
100MB	4 bytes	0.012	0.141	0.128	4711989
150MB	4 bytes	0.020	0.231	0.174	7067983
200MB	4 bytes	0.027	0.310	0.227	9423976
250MB	4 bytes	0.031	0.402	0.261	11779969

Table 4. Comparisons of performance with 35% of Records Matching

Text Records	Pattern size	GPU(s)	Serial on CPU(s)	Multithreaded on CPU(s)	No of occurrences of pattern
50MB	8 byte	0.00621	0.066	0.052	2355994
100MB	8 byte	0.0095	0.123	0.104	4711989
150MB	8 byte	0.0144	0.181	0.141	7067983
200MB	8 byte	0.019	0.252	0.20	9423976
250MB	8 byte	0.025	0.303	0.244	11779969

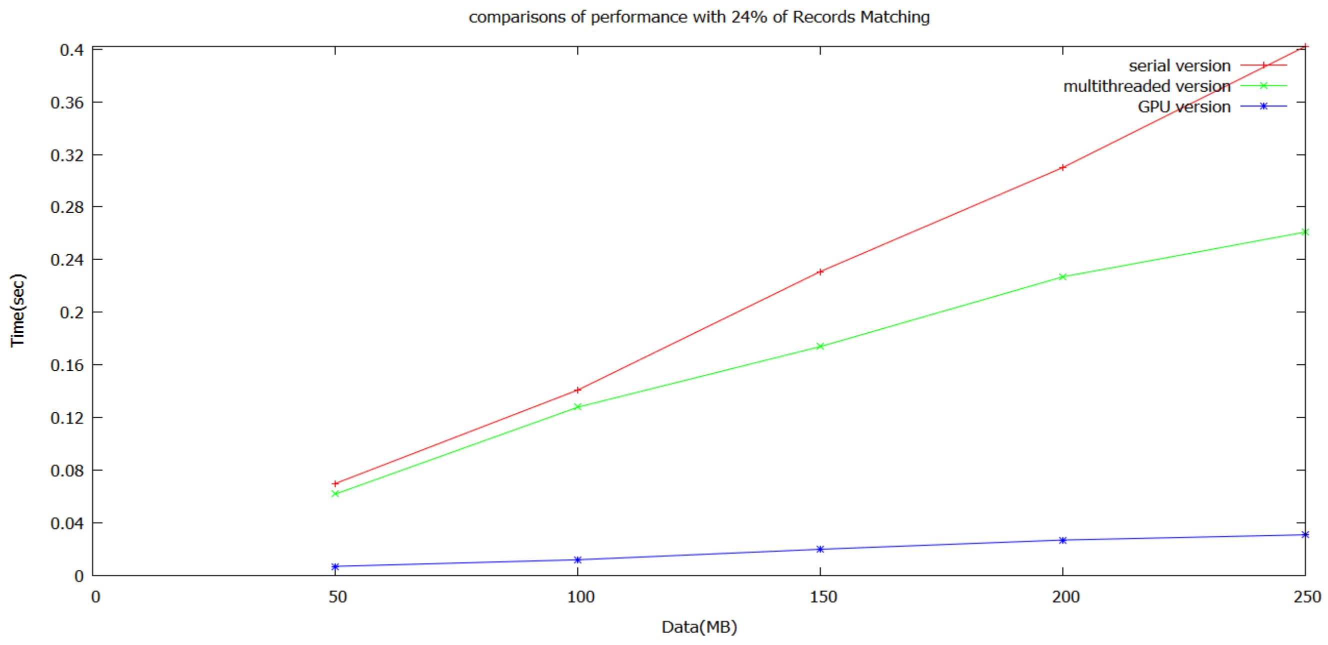


Fig 5: Execution time performance of all three EBM versions for 4 byte pattern size

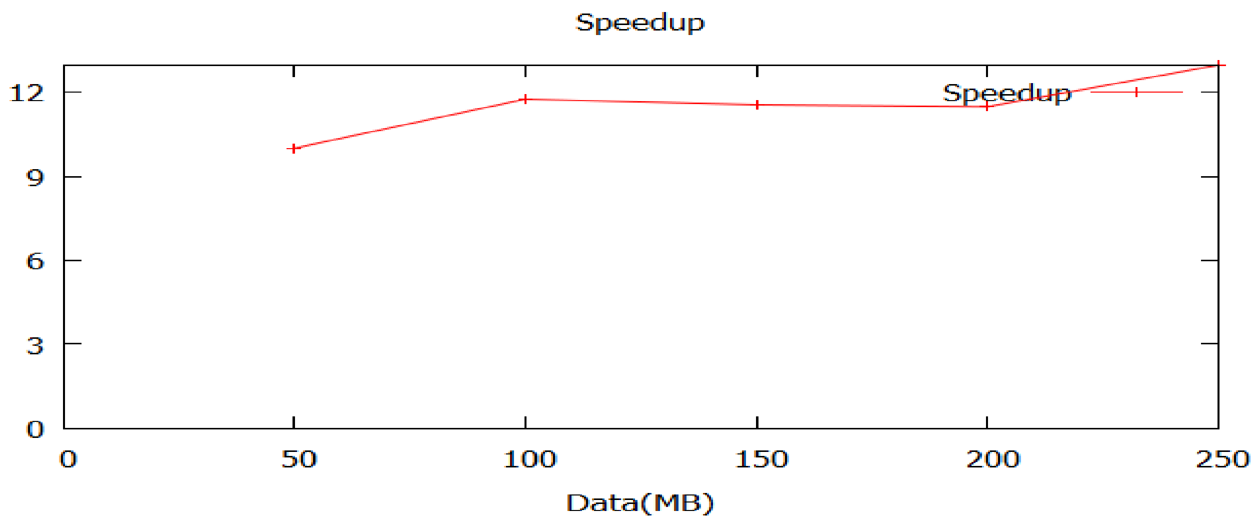


Fig 6: Speedup performance for 4 byte pattern

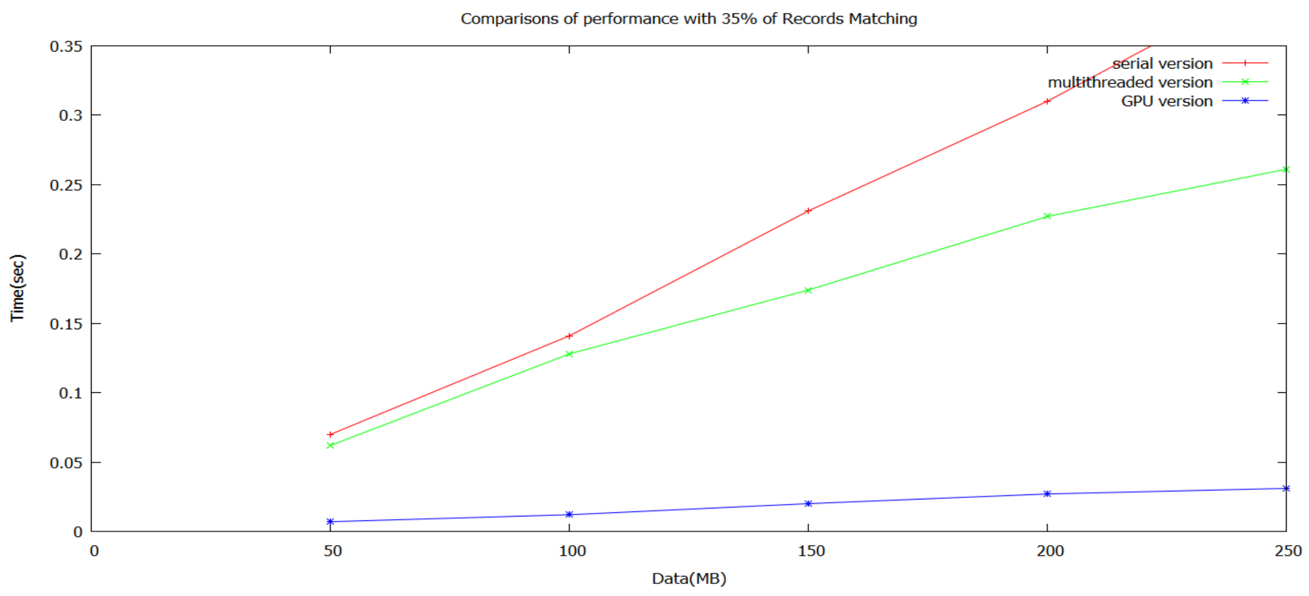


Fig 7: Execution time performance of all three EBM versions for 8 byte pattern size

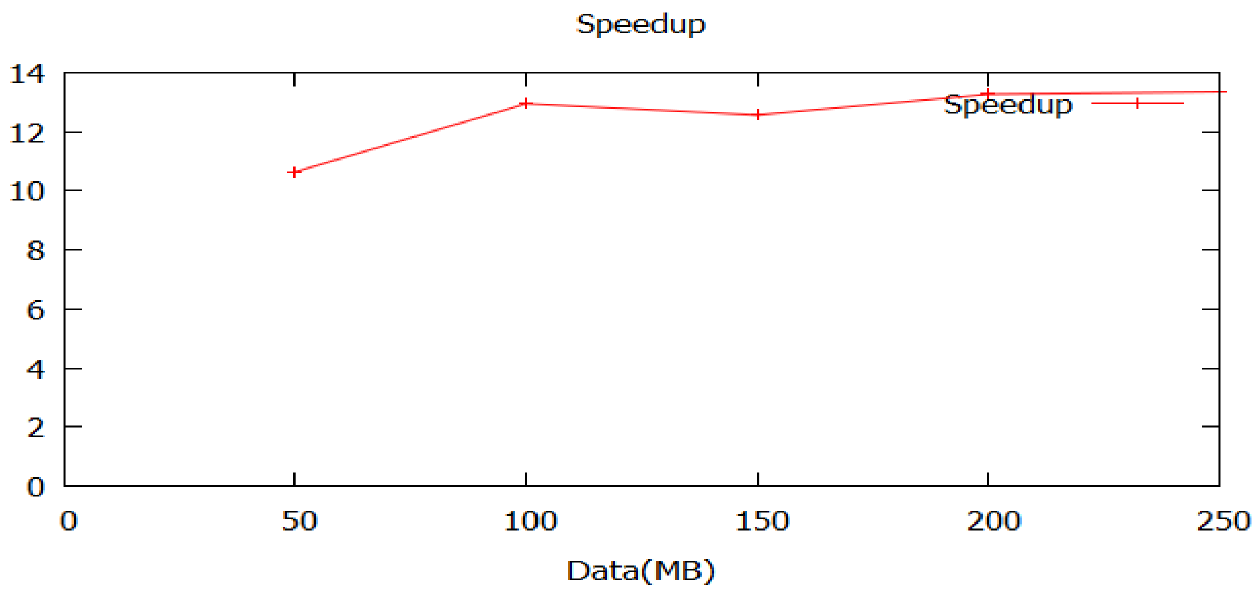


Fig 8: Speedup performance for 8 byte pattern

5. CONCLUSION

In this paper we have presented the enhanced Boyer-Moore algorithm based on the GPU.The enhanced Boyer-Moore algorithm accelerated the pattern searching process about 10x than serial version on CPU and almost 9x faster than the multithreaded version on CPU.As future work and it can try to do more research and more suitable application on the GPU such as text editor and virus scanning by the enhanced Boyer-Moore algorithm. The biggest challenge in any next step would be to scale the GPU application to handle larger data loads. Another challenge is the efficient use of the small on-chip memories.

6. ACKNOWLEDGMENT

This work was totally supported by the Guru Ghasidas Central University; Bilaspur (India).Special thanks are to be Dr. Manish Shrivastava. He gave to many suggestions.

7. REFERENCES

[1] Cheng-Hung Lin, Sheng-Yu Tsai, Chen-Hsiung Liu, Shih Chieh Chang, Jyuo-Min Shyu," Accelerating string matching using multi-threaded algorithm on GPU", IEEE Globecom 2010.

[2] Kenneth Ryan VeriSign Labs, "An Evaluation of GPUs for Accelerating a Selected String Searching Algorithm",21345 ridge top circle,Dulles,VA 20152.

[3] Charalampos S. Kouzinopoulos and Konstantinos G.Margaritis," String Matching on a multicore GPU

using CUDA", 2009 13th Pan-Hellenic Conference on Informatics.

[4] Jiangfeng Peng,Hu Chen,"CUgrep:A GPU-based high Performance Multi-String Matching System"2010 IEEE:V1-77 to V1-81

[5] Lingling yuan,"An improved algorithm for Boyer-Moore string Matching Chinese Information Processing" , IEEE.pp. 182-184,2011.

[6] Zhengda Xiong,"A composite Boyer-Moore Algorithm for the string Matching Problem" IEEE.pp. 492-496,2010.

[7] Xingxing Wang," A BM algorithm oriented on Network Security Audit System" IEEE.978-1-4244-5895, 2010.

[8] Yang Tong, Qiao Xiang-dong, "Analyze and Improvement of BM Algorithm" IEEE: 978-1-4244-3693, 2009.

[9] Prasad JC, Dr.K.S.M. Panicker,"Single Pattern Search Implementations in a Cluster Computing Environment" , IEEE.pp391-396,2010.

[10] Knuth,D.E,Morries,Jr.,J.H.,and pratt,V.B." fast pattern matching in string SIAM J.comptng.6,2(1977),pp323-350.

[11] Yuting Han, Guoai Xu "Improved algorithm of pattern matching based on BMHS", IEEE.pp238-241,2010.

[12] Zhu Yong giang,"Two enhanced BM algorithm in pattern matching", IEEE.pp341-346,2011.

[13] Yihui SHAN, Yuming JIANG, Shiyuan TIAN, "Improved Pattern Matching Algorithm of BMHS for Intrusion Detection". Computer Engineering, vol.35, 2009, pp.170-173

[14] Zhanjun REN, Quanzhu YAO, Xiaofeng WANG, Youjiao ZOU,"Application of Pattern Matching Algorithm in Intrusion Detection Technique". Modern electronic technology, vol.2, 2009, pp.63-67

[15] Baishuhong. Eason, An Improvement on BM Algorithm for Chinese, fujiandiannao, pp. 90–91, October. 2009.