ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 5, Issue 11, November 2017

# GPU Accelerated Pattern Matching Algorithm for Packet Data to Detect Virus Signature

Bhagyashri Chhagan Wadile, Prof.N.R.Wankhade

PG Student, Department of Computer Engineering, Late G.N.Sapkal College of Engineering, Maharashtra, India

H.O.D, Department of Computer Engineering, Late G.N.Sapkal College of Engineering, Maharashtra, India

**ABSTRACT:** The algorithm proposed In this paper is use for exact pattern matching on GPU. Among some famous algorithms, the Aho-Corasick Algorithm match multiple pattern simultaneously. a Traditional Aho-Corasick Algorithm matches the pattern by traversing state machine, known as Deterministic finite automata(DFA).Signature matching is important Technique in virus/worm detection, but Aho-corasick algorithm was developed only for string and virus/worm signature could be in regular expression . In this research work new guidelines are proposed for an efficient GPU adaptation of Aho-corasick algorithm for regular expression matching. Also several technique is introduced to optimization on GPU, including reducing global memory access, storage format for output table. To evaluate performance proposed system will use SNORT virus database. Proposed algorithm Implemented on NVIDIA GTX-680 Graphics card using CUDA.

**KEYWORDS**: Aho-Corasick , CUDA ,Graphics processing Unit, Pattern Matching.

## I. INTRODUCTION

String and pattern matching problems are fundamental to any computer application involving text processing. Pattern matching can be used in finding similar DNA or protein sequences, in Intrusion Detection Systems to find the presence of virus signatures in the incoming packets. Network Intrusion Detection Systems (NIDS) have been widely adopted to protect computer systems from several network attacks such as denial-of-service attacks, port scans, or malwares. The most critical operation affecting the performance of NIDS is to inspect packet content against thousands of attack patterns. Due to the ever increasing number of attacks, traditional sequential pattern matching algorithms are inadequate to meet the throughput requirements for high-speed networks. So we need a multi-pattern matching algorithm to meet the throughput requirements for high-speed networks. Snort [3] a Network Intrusion Detection System, uses an Aho-Corasick (AC) algorithm to match the input data with the known attack patterns. The large amount of incoming data cannot be handled by this traditional algorithm and it drops the packet data. Pattern matching routines in Snort account for up to 70% of total execution time [4]. To meet the throughput requirement for high-speed networks this algorithm proves to be inadequate.

## II. PAGE LAYOUT

In the past year their many approaches have been proposed to accelerate pattern matching process. This approaches are classified into logic architecture and Memory architecture [5][6]. In logic architecture the attack pattern are stored on the logic circuit that are implemented on FPGA i.e. Field programmable gate array.in memory architecture we draw a state machine of patterns and traverse the state machine to find the pattern. SNORT which is popular open source intrusion detection system also uses aAho-Corasick pattern matching algorithm for detection[3][8]. SNORT uses efficient pattern matching algorithm hence its run time is independent on pattern length and liner to length of target string. File carving is the process of reassembling computer files in the absence of file system metadata. In scalpel we use single pattern matching algorithm hence its run time linear to product of no of pattern and the target string length[7]. There are several attempt to improve performance of multi pattern matching using parallelism. For Example Xinyan Zha[12] compare performance of Aho-Corasick algorithm on CPU and GPU. Xinyan Zha also gives different parallel approaches of Aho-corasick algorithm depending on pattern storage. Huang et al.[13] implemented a Wu-

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(A High Impact Factor, Monthly, Peer Reviewed Journal)*

*Website: www.ijircce.com*

**Vol. 5, Issue 11, November 2017**

Membermultiple-pattern matching algorithm on GPUs and achieved speedup twice as fast as the traditional Wu-Member algorithm. Peng et al.[14 ] proposed GPU based web page matching system using advanced Aho-Corasick algorithm ,the proposed algorithm is 28 time faster than original Aho-Corasick algorithm which used in SNORT[8]. Mu et al[15] developed efficient GPU based router application and proposed GPU based routing table lookup solution whichis delivered higher throughput than previous CPU based solution. In this paper we proposed the Aho-Corasickmultipattern matching algorithmfor regular expression matching through use of GPU.Toefficiently utilized GPU power, proposed algorithm usages several optimization technique like reducing global memory accessing, CSR representation for storing state transition table of Aho-Corasick algorithm, more use of GPU shared memory for thread communication etc. The contributation of this work include :

- The implementation of GPU based Aho-Corasick Algorithm which support both ,string searching and regular expression matching.
- Improving Performance of GPU using different memory hierarchies that GPU provides.
- Use of CSR representation for storing AC machine state transition table.

## III. THE NVIDIA KEPLER ARCHITECTURE

Figure 1 shows NVIDA GTX 680 Kepler architecture.Kepler GTX 680 is the example of great performance/watt Streaming Multiprocessor, also know as “SMX.” In GTX680 total no of CUDA core is 1536 .GTX Core work on Base Clock 1006 and Boost Clock 1058.Each CUDA core in GTX 680 Support 1024 number of active thread. The GTX 680 has 2048MB memory which work on 6 MHz Speed.. For improved power efficiency, the SMX now runs at graphics clock rather than 2x graphics clock but with 1536 CUDA cores in GK104, the GeForce GTX 680 SMX provides 2x the performance per watt of Fermi’s SM (GF110). Because of this eForce GTX 680 to deliver revolutionary performance/watt when compared to GeForce GTX 580. Level-3 Heading: A level-3 heading must be indented, in Italic and numbered with an Arabic numeral followed by a right parenthesis. The level-3 heading must end with a colon.

CUDA is NVIDIA parallel computing architecture which is used to increased the performance of GPU in processing.Using CUDA GPU is use for GPGPU computing i.e. general purpose graphics processing units.unlike CPU,GPU has parallel architecture in which we can execute number of threads parallel[10].

CUDA programmer is C language programmed which is written for HOST(CPU)[10]. C,Cpp extension are support by CUDA programming which is useful for transfer data from host to device memory (CPU to GPU) and also used to called Kernel function which execute on GPU cores[10].CUDA has access different types of memory's each having different caching properties,speed,capacity[10][11].The GPU CUDA memories are as follows:

- Device Memory of GTX680 :2GB of Device Memory available in GTX 680 graphics card.Device memory is accessible to all threads.each thread can read/write memory content directly.However latency time of device memory is high as compared to other memory.In GPU,thread thread scheduler hide this latency time by scheduling another thread if current thread requires some ip/op operation.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 5, Issue 11, November 2017

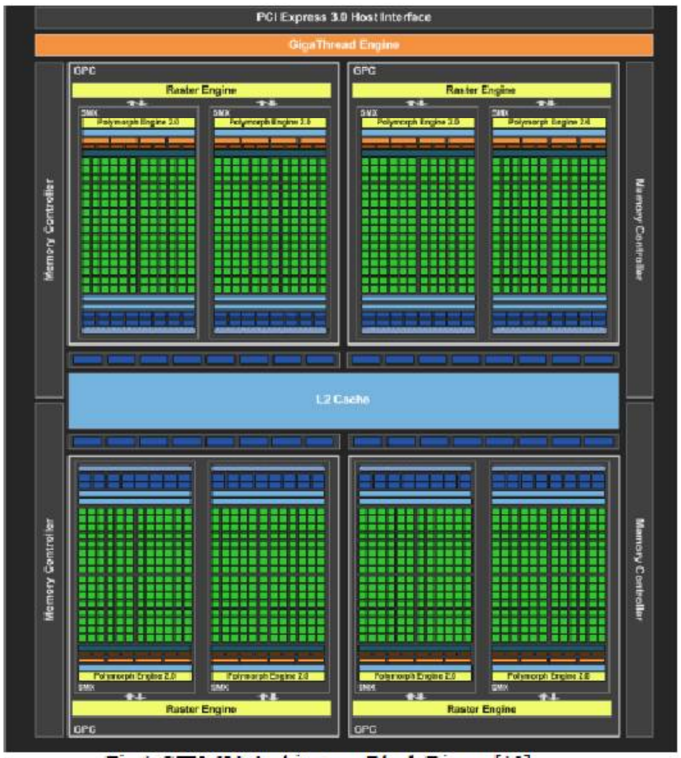


Fig 1:GTX 680 Architecture Block Digram[10]

Constant memory of GTX680:In GTX 680 constant memory size is 65536KB.In CUDA constant memory is used by all thread and it is read only memory.

- Shared memory of GTX680: Threads in one block uses some common memory which is not accessible to threads in other block such memory is called shared memory. shared memory per block in GTX 680 is 49152.
- Texture memory: Texture memory *is* global memory. The only difference is that textures are accessed through a dedicated read-only cache, and that the cache includes hardware filtering which can perform linear floating point interpolation as part of the read process. The cache, however, is different to a conventional cache, in that it is optimized for spatial locality (in the coordinate system of the texture) and not locality in memory. For some applications, this is ideal and will give a performance advantage both because of the cache and the free FLOPs you can get from the filtering hardware, but for others, it won't and textures can be *slower* because access involves a cache miss penalty in addition to the global memory read, and interpolation is not required.
- Pinned memory: This is part of the host memory. Data transfer between pinned and device memory is faster than between page able host memory anddevice memory. Also, this data transfer can be done concurrent with kernel execution.

## III. MATHEMATICAL MODEL

In pattern matching we have to report all pattern in given text. A deterministic finite automata represent machine which detect the pattern. A deterministic finite automata represented by 5 tuple ($Q$, $\Sigma$, $\delta$, $q0$, $F$),where : $\Sigma$ represent as alphabet ,$Q$ is represent set of states ,T is transition function,q0 is initial state and F is the final state .Given input string as p1,p2,…pN,DFA processing as follows :at step 0,the DFA is in state q0.in next step i it will be in si=T(si-1,Ii).
S={P,F,T,ACSM,Rnfa,Ndfa,RD}
Where, P: patterns in datasets , F: functions, T:input text, ACSM: Aho-Corasick state machine ,Rnfa: Regular expression

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

*(A High Impact Factor, Monthly, Peer Reviewed Journal)*

*Website: www.ijircce.com*

Vol. 5, Issue 11, November 2017

to Non deterministic finite automata Ndfa:NFA to Deterministic faint automata, RD: result data.
Let P={p1,p2,p3…..pn}
F={f1,f2,f3,f4..fn}
T={t1,t2,t3….tn}
RD={rd1,rd2,rd3…rdn}
Function f1 :
It read pattern database and return no of pattern in given database.
F1(P)={p1,p2, p3…pn}
Function f2:
It read pattern and classifies it according to string or Regular expression
F2(pi)={ RE |a-z|A-Z|0|1}
Function f3:
Takes R.E as input and produce Non-deterministic automata.
F3(RE)={Rnfa1, Rnfa2, Rnfa3…. Rnfa(n)}
Function f4:
It take Non-deterministic finite automata as input as gives Deterministic finite automata.
F4(NFA)={DFA1,DFA2,DFA3….DFAn}
Function f5:
It read normal string pattern and convert it into Deterministic finite automata.
F5(pi): {DFA1,DFA2,DFA3….DFAn}
Function f6:
It reads all input text T and gives one byte of text stream,
F6(T)={T1,T2,T3,T4...Tn}
Function f7:
It is executed by each thread on GPU.it take one byte of t1 as input as perform pattern matching on DFA and if pattern match gives index position.
F(Ti)={pos1,pos2,pos3…posn} reference items consecutively in square brackets (e.g. [1]). When referring to a reference item, please simply use the reference number, as in [2].

Examples of reference items of different categories shown in the References section include:

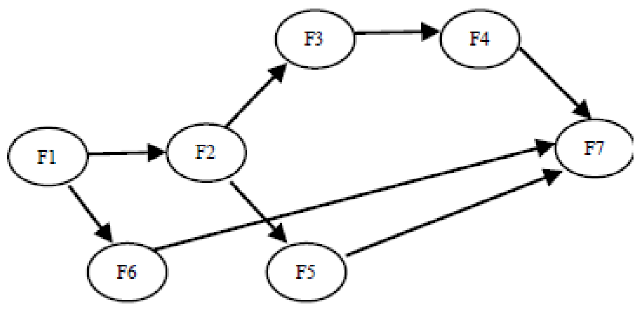


Figure 2:Functional Dependency

## IV. AHO-CORASICKALGORITHM

The Aho-Corasick algorithm was proposed in 1975 by Alfred V. Aho and Margaret J.Corasick[1] , and this is the most effective multi pattern matching algorithm. Aho- Corasick (AC) is a multi-string matching algorithm, meaning it matches the input against multiple strings at the same time. Multi-string matching algorithms generally pre-process the set of strings, and then search all of them together over the input text. The algorithm works in two parts. The first part is the building of the tree from keywords you want to search for, and the second part is searching the text for the keywords using the previously built tree (state machine). Searching for a keyword

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

**International Journal of Innovative Research in Computer and Communication Engineering**

*(A High Impact Factor, Monthly, Peer Reviewed Journal)*

*Website: www.ijircce.com*

**Vol. 5, Issue 11, November 2017**

is very efficient, because it only moves through the states in the state machine. If a character is match, goto() function is executed otherwise it follows fail() function. The pattern matching machine is constructed by starting from the root node and inserting each pattern one after another[1][4]. The algorithm works asfollow:

- Start at Root Node(0)
- For each pattern in P do
  - If the path end before the pattern ,then continue adding edge and state inpattern.
  - Once the pattern is identified then mark its as finalstate.

The time taken to search pattern is linear to pattern length .the search algorithm is as follow :

- Start at root node (0)
- For each character in i put string follow the path of constructedtree
  - If we reach to final state node then pattern is present.
  - If path terminated before reaching to end then that pattern is notpresent.

The Aho-Corasick Algorithm uses three function :

- The goto() function g(state, input symbol) is the next state from current state on receiving inputsymbol.
- The failure function f(q). for $q \neq 0$, is the next state in case of a mismatch i.e. Failure link .
- Output function out(q) gives set of pattern that found at stateq

Figure 3 show AC state machine for pattern (ABED,ABCD,EABD,AB). As shown in Fig.3, states 2, 4 and 6,10 are the final states of the patterns “AB,” “ABED,” and “ABCD,” “EABD ”while states 4 represent the final state of the pattern “ABED .” The internal state 2 becomes a final state because the pattern “AB” is a suffix of the pattern “ABED.” Therefore, the state machine matches the pattern “AB” when the state machine reaches state 4. For example, consider the case where we wish to match an input stream containing “MNPSQABEABD” from the AC state machine in Fig.3.

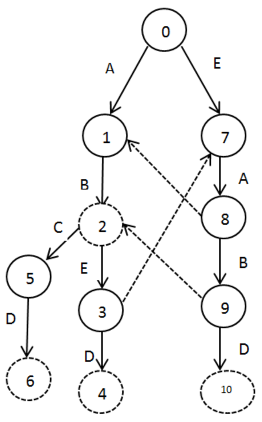


Fig 3: AC State Machine For Pattern ABED,ABCD,EABD,AB

The AC machine starts from state 0, and remain in that state for input ‘M’,’N’,’P’,’S’,’Q’ because AC machine don’t have any valid transition from 0 state for these input. For next input ‘A’ it travels to state 1, and then reaches state 2 for input ‘B’ that is the final state of the pattern “AB”. Then for next input ‘E’ it goes to state 3, for next input ‘A’ state 3 don’t have any valid transition hens AC machine take failure transition and goes to state 7,next it goes to state 8 for ‘A’, for input ‘B’ it goes to next state 9,and for next input ‘D’ it goes to state 10 which is final state for pattern ‘EABD’.Hence we get the pattern ‘AB’,’EABD’. In summary, the AC algorithm matches all patterns in O(n) time for processing an input stream of lengthn[9].

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 5, Issue 11, November 2017

## V. SYSTEM ARCHITECTURE AND OVERVIEW

GPU GTX 680 consist of 1024 number of thread per block. The device code is launched by host which organized as grid which is one or to dimensional array of block. using these threads we achieve parallelism in Aho-Corasick algorithm.

### *A. parallel Aho-Corasick Algorithm on GPU*

Proposed work modifies the traditional Aho-corasick algorithm .
Algorithm:
**Input** : DFA state transition Table, Set of patterns {P1,P2,P3..Pn} , Input Text T.
**Output**: Locations where the patterns occur In T.
**Begin**

```
o  Declare n thread one for each byte.
o  Curent_state=0
o  Pattern_length=0
o  Number_of_ pattern=0
o  For cursor=start_of_ string To end_of_ string

o  If (DFAtable[current_ state][T[cursor]].next state ≠0) then
o  If(DFAtable[current_state][T[cursor]].isFina l=0) then current_state=DFAtable[current_state][T[Cursor]].next state
o  pattern_length=pattern_length +1
o  else
o  match_Position=cursor-pattern length
o  match_ state=current_state
o  num_Pattern=num_Pattern +1
o  else
o  pattern_length=0
o  cursor_state=0
o   end
```

### *B. Pattern Matching on GPU*

During the reading of input stream the algorithm moves over input data stream one byte at a time i.e. each thread will performs searching operation on one byte of data as shown Figure4.for each byte thread execute above proposed algorithm. The algorithm switches current stateto next state according to state transition table. Figure 4 represent AC pattern Machine process on GPU kernel.as show in figure we get a input as one byte at a time, then we read each character from one byte, and passes to AC state machine which decide next state for that input and when pattern is matched we stored output in output array which contained index position of pattern in string.
We take the advantage of available streaming processor on GPU and use them to create multiple data processing threads and assign one byte of data to each thread. Because of this approach thread operate independently. Figure 5 is multithread pattern matching approach.as shown in figure input stream is divide to one byte of data .Their are 'n' no of threads, each thread work on one byte of data and follows instruction in AC state machine. Each thread execute AC state machine function because GPU operate on SIMD(Single instruction multiple data) principal. Each thread may access different location of State transition table .

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

**International Journal of Innovative Research in Computer and Communication Engineering**

*(A High Impact Factor, Monthly, Peer Reviewed Journal)*

*Website: www.ijircce.com*

**Vol. 5, Issue 11, November 2017**

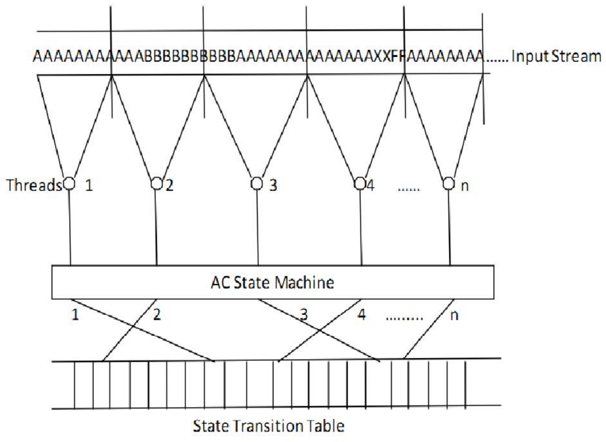

AAAAAAAAAAAABBBBBBBBBBBAAAAAAAAAAAAAAAAXXFFAAAAAAAA...... Input Stream
Threads
1
2
3
4
......
n
AC State Machine
1
2
3
4
.........
n
State Transition Table

Fig 3:Multi threading Approach

*C.Optimization of Device Memory*

Two important task in DFA matching is reading the input data and fetching next state from state table. this memory transfer can take lots of time.in general memory latency is hide by using several threads in parallel .multiple thread can utilized memory by overlapping data with computation. In traditional Aho-Corasick algorithm matrix is used to store state transition table. In parallel approach of Aho-Corasick algorithm, proposed algorithm use CSR representation of Sparse matrix. In Aho-Corasick algorithm the state transition table is stored in matrix, this matrix is sparse matrix i.e. amatrix where most of entries are zero.it is very inefficient use of computer memory(it is not useful to store may zero values in computer memory) and more important is computer programed need more time to run this code i.e. unnecessary computation is required ,because of these reason proposed system use Compress sparse row format for storing state transition table in parallel AC algorithm . In this research proposed system uses The NVIDIA CUDA Sparse Matrix library (cuSPARSE).In cuSPARSE there are collection of basic linear algebra subroutines which use for sparse matrix and using this subroutines the matrix provide up to 8x faster performance.

$$\begin{pmatrix} 0 & 0 & 1 & 8 \\ 0 & 4 & 0 & 0 \\ 0 & 0 & 0 & 2 \\ 0 & 3 & 5 & 0 \end{pmatrix}$$

(a)

| Index | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Val | 1 | 8 | 4 | 2 | 3 | 5 |
| Col_in | 3 | 4 | 2 | 4 | 2 | 3 |
| Row_in | 1 | 3 | 4 | 5 | | |

(b)

*D.Host Memory Optimization*

Time required to transfer the data between CPU and GPU is more, proposed system reduce number of transaction between CPU and GPU by using page lock memory. The page lock memory offer better performance as swapping is not their.it also access directly at GPU using DMA.hence by using page lock memory improve overall performance by reducing cost of data transfer to and from GPU.

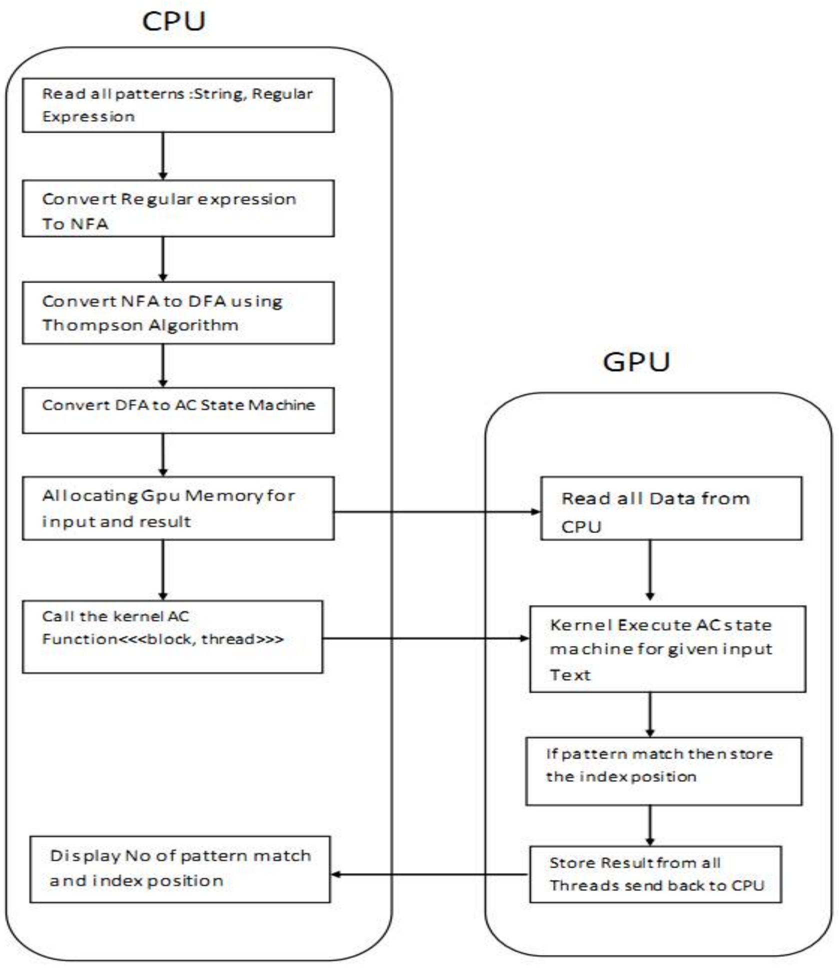
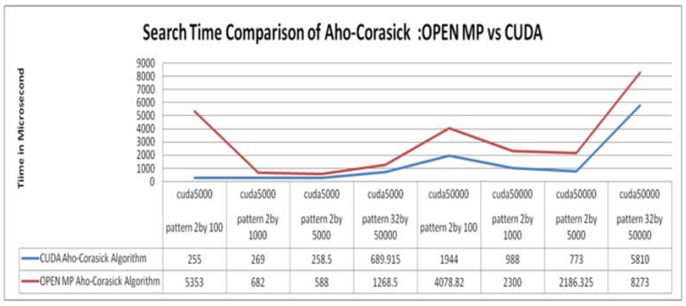

CPU
Read all patterns :String, Regular Expression
Convert Regular expression To NFA
Convert NFA to DFA using Thompson Algorithm
Convert DFA to AC State Machine
Allocating Gpu Memory for input and result
Call the kernel AC Function<<<block, thread>>>
Display No of pattern match and index position
GPU
Read all Data from CPU
Kernel Execute AC state machine for given input Text
If pattern match then store the index position
Store Result from all Threads send back to CPU

Fig 8:System Architecture for pattern matching on GPU

## VI. RESULT AND ANALYSIS

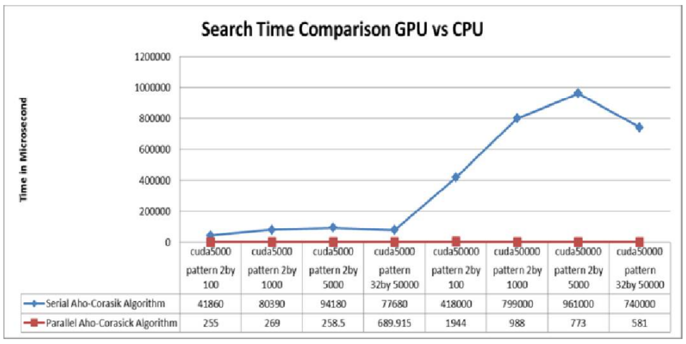
The GPU is specializedfor compute-intensive, highly parallelcomputationMore transistors are devoted to data processing rather than data caching and flow control .The fast-growing video game industry exerts strong economic pressure that forces constant innovation.

We test the performance of serial implementation ofAho- corasick algorithm and Parallel Version of Un-optimized algorithm with fixed number of pattern. Here we use SNORT virus signature database for FSM and randomly generated patterns and payloads.



| | cuda5000 pattern 2by 100 | cuda5000 pattern 2by 1000 | cuda5000 pattern 2by 5000 | cuda5000 pattern 32by 50000 | cuda50000 pattern 2by 100 | cuda50000 pattern 2by 1000 | cuda50000 pattern 2by 5000 | cuda50000 pattern 32by 50000 |
|---|---|---|---|---|---|---|---|---|
| CUDA Aho-Corasick Algorithm | 255 | 269 | 258.5 | 689.915 | 1944 | 988 | 773 | 5810 |
| OPEN MP Aho-Corasick Algorithm | 5353 | 682 | 588 | 1268.5 | 4078.82 | 2300 | 2186.325 | 8273 |

Table I: : Aho-Corasick Result Analysis(Serial vs. Open MP)



| | cuda5000 pattern 2by 100 | cuda5000 pattern 2by 1000 | cuda5000 pattern 2by 5000 | cuda5000 pattern 32by 50000 | cuda50000 pattern 2by 100 | cuda50000 pattern 2by 1000 | cuda50000 pattern 2by 5000 | cuda50000 pattern 32by 50000 |
|---|---|---|---|---|---|---|---|---|
| Serial Aho-Corasik Algorithm | 41860 | 80390 | 94180 | 77680 | 418000 | 799000 | 961000 | 740000 |
| Parallel Aho-Corasick Algorithm | 255 | 269 | 258.5 | 689.915 | 1944 | 988 | 773 | 581 |

Table II : Search Time Comparison GPU vs CPU

## VII. CONCLUSION AND FUTUREWORK

GPU provide high computing power then CPU. Parallel Implementation of Aho-Corasick algorithm reduces the time required for pattern matching on large datasets. Proposed algorithm works on String and regular expression. proposed work also uses various optimization technique on host and device. Serial implementation result are provided for two pattern matching algorithm Aho-Corasick and Brute Force algorithm.

As a part of future work one can test performance of Aho- Corasick algorithm on various memory Hierarchy provide by GPU.

## REFERENCES

[1]. A.V. Aho and M.J. Corasick. Efficient String Matching: An Aid to Bibliographic Search. Communications of the ACM, 18(6):333–340, 1975.
[2]. X. Chen, B. Fang, L. Li, and Y. Jiang. WM+: An Optimal Multi-pattern String Matching Algorithm Based on the WM Algorithm. Advanced Parallel Processing Technologies, pages 515–523, 2005.
[3]. GNU Grep. Webpage containing information about the gnu grep search utility.Website, 2012.http://www.gnu.org/software/grep/.
[4]. G. Navarro and M. Raffinot. Flexible Pattern Matching in Strings: Practical On-line Search Algorithms for Texts and Biological Sequences. Cambridge University Press, 2002.
[5]. M. Crochemore, A. Czumaj, L. Gasieniec, T. Lecroq, W. Plandowski, and
W. Rytter. Fast Practical Multi-pattern Matching. Information Processing Letters, 71(3-4):107 – 113, 1999.
[6]. M. Crochemore and W. Rytter. Text Algorithms.Oxford University Press, Inc.,1994.
[7]. Z. Zhou, Y. Xue, J. Liu, W. Zhang, and J. Li. MDH: A High Speed Multi-phase Dynamic Hash String Matching Algorithm for Large-Scale Pattern Set. Information and Communications Security,4861:201–215, 2007 . [8].Snort. Webpage containing information on the snort intrusion prevention and detection system.Website, 2010.http://www.snort.org/.
[sss9]. S. Dori and G.M. Landau. Construction of AhoCorasick Automaton in Linear Time for Integer Alphabets. Information Processing Letters, 98(2):66– 72, 2006.
[10]. CUDA Zone. Official webpage of the nvidiacudaapi. Website, http://www.nvidia.com/object/cuda home.html.
[11]. N Wilt. The CUDA Handbook: A Comprehensive Guide to GPU Programming. Addison-Wesley Professional, 2013.
[12].XinyanZha and SartajSahni," Multipattern String Matching On AGPU".Communications of the ACM, 20(10):762–772, 1977.
[13]. N. Huang, H. Hung, S.Lai et al, A GPU-based Multiple-pattern Match- ing Algorithm for Network Intrusion Detection Systems, The 22nd International Conference on Advanced Information Networking and Applications, 2008

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(A High Impact Factor, Monthly, Peer Reviewed Journal)

Website: www.ijircce.com

Vol. 5, Issue 11, November 2017

[14]. J.F. Peng, H. Chen, and S.H. Shi, "The GPU-Based String Matching
System in Advanced AC Algorithm," Proc. Int'l Conf. Computer and Information Technology (CIT '10), pp. 1158-1163, 2010.
[15]. S. Mu, X. Zhang, N. Zhang, J. Lu, Y.S. Deng, and S. Zhang, "IP Routing sProcessing with Graphic Processors," Proc. Design, Auto-mation Test in Europe Conf. Exhibition (DATE '10), pp. 93-98, 2010.
[16].KEN THOMPSON," PROGRAMMING TECHNIQUES: REGULAR EXPRESSION SEARCH ALGORITHM", JUNE 1968
[17]. John E. Hopcroft and Jeffrey D. Ullman, Introduction to AutomataTheory, Languages, and Computation, Addison-Wesley Publishing, Reading Massachusetts, 1979.
[18]. M. O. Rabin and D. Scott. Finite automata and their decision problems. IBM Journal of Research and Development, 3(2):114–125, 1959.