International Journal of Electronics, Communication & Soft Computing Science and Engineering
ISSN: 2277-9477, Volume 3, Issue 3

Parallelization of Multipattern Matching on GPU

Mr.Gaurav K. Bhamare, Prof.Satish S. Banait

Abstract — **Pattern matching is a highly computationally intensive operation used in SNORT system but due to the increasingly storage capacity and the link speed the amount of data that need to be match against pattern is increased rapidly and traditional system is fail to match that data.GPU Computing Have attracted lots of attention due to their large amount of data processing. 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. Signature matching is important Technique in virus/worm detection, but traditional 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.**

Key Words — **Aho-Corasick , CUDA ,Graphics processing Unit, Pattern Matching, SNORT,DFA,regular expression**

I. INTRODUCTION

In Multipattern Matching algorithm, we have to report all occurrence of pattern in given string.Multipattern string matching use in number of application such as network intrusion detection, digital forensics, natural language processing[2]. For example, Snort is open source network intrusion detection system which contained thousands of pattern that are match against packet in network for virus/worm signature detection. In the case of Snort we have to search thousands of pattern per packet in very small time. Due to increasing number of attack this traditional sequential pattern matching technique is inefficient.

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 Aho-Corasick pattern matching algorithm for detection[3][8]. 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 ofmultipattern matching using parallelism. For Example XinyanZha[12] compare performance of Aho-Corasick algorithm on CPU and GPU.XinyanZha also gives different parallelapproaches of Aho-corasick algorithm depending on pattern storage.Huang et al.[13] implemented aWu-Manbermultiple-pattern matching algorithm onGPUs and achieved speedup twice as fast as the traditional Wu-Manber algorithm.Peng et al.[14] proposed GPU based web page matching system using advanced Ago-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 which is delivered higher throughput than previous CPU based solution.

In this paper we proposed the Aho-Corasick multipattern matching algorithm for regular expression matching through use of GPU.To efficiently 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 contribution 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 .

II. NVIDIA KEPLER ARCHITECTURE

Figure 1 shows NVIDA GTX 680 Keplerarchitecture.Kepler GTX 680 is the example of great performance/watt Streaming Multiprocessor, also known 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 GeForce GTX 680 to deliver revolutionary performance/watt when compared to GeForce GTX 580.

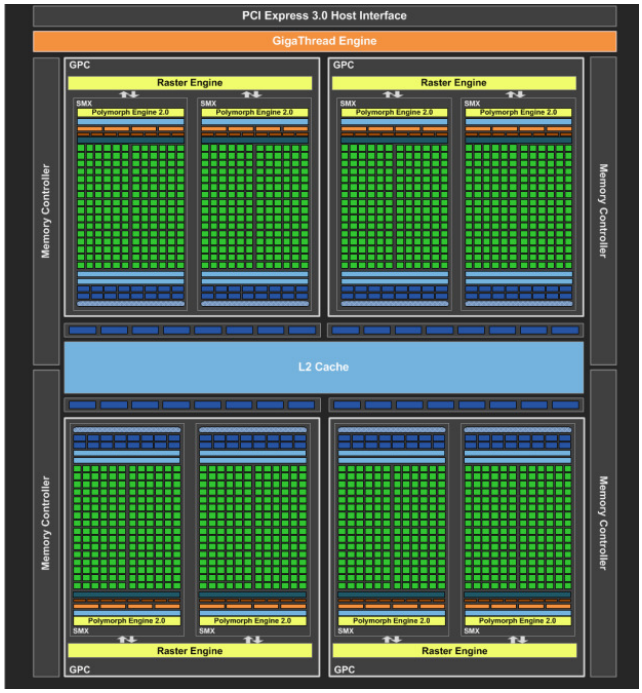


Fig 1:GTX 680 Architecture Block Digram[10]

CUDA is NVIDIA parallel computing archiecture which is used to increased the performance of GPU in processing.Using CUDA GPU is use for GPGPU computing i.e. genral purpose graphcis 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].

III. THE AHO-CORASICK ALGORITHM

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 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]. Figure 2 show AC state machine for pattern (ABED,ABCD,EABD,AB). As shown in Fig.2, 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. 2.

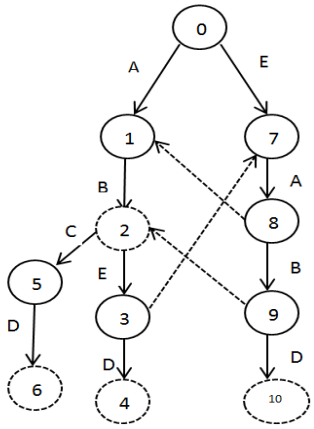


Fig 2: AC State Machine For Pattern ABED,ABCD,EABD,AB

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 length n[9].

IV. PROPOSED WORK

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 $\{P_1,P_2,P_3..P_n\}$, Input Text T.

Output: Locations where the patterns occur In T

Begin

- Declare n thread one for each byte.
- Curent_state=0
- Pattern_length=0
- Number_of_ pattern=0
- **For** cursor=start_of_ string To end_of_ string
- **If** (DFAtable[current_ state][T[cursor]].next state ≠0) then
    - **If**(DFAtable[current_state][T[cursor]].isFinal=0) then
    - current_state=DFAtable[current_state][T[Cursor]].next state
    - pattern_length=pattern_length +1
    - **else**
    - match_Position=cursor-pattern length
    - match_ state=current_state
    - num_Pattern=num_Pattern +1
    - **else**
    - pattern_length=0
    - cursor_state=0
- 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 Figure 4.for each byte thread execute above proposed algorithm. The algorithm switches current state to 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.

C. Optimization of Device Memory

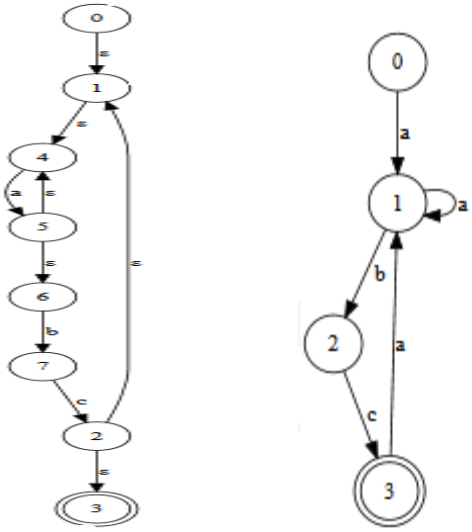
Two important task in DFA matching is reading the input data and fetching next state from state table. this memory transfer can take lots of time.in general memory latency is hide by using several threads in parallel .multiple thread can utilized memory by overlapping data with computation. In traditional 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. a matrix 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.

D.Regular expression matching on GPU

A regular expression is set of character that specifies pattern.It is easy to find a string or word in given text. Every editor on our computer system can do this, but regular expression are more powerful and flexible. using regular expression one can find word of certain size .You also search word which containing four or more vowel in it and it ends with 'Z'.Numbers, punctuation characters, you name it, a regular expression can find it. For example (c*)cc represent infinite number of possibilities like {cc,ccc,cccc}.

1. Converting Regular expression to Finite Automata

As Aho-Corasick algorithm work on finite automata(DFA) that's why we convert regular expression to DFA and then used that DFA for pattern matching. The most common approach to convert regular expression to DFA is first convert regular expression to NFA i.e. non deterministic finite automata then convert NFA to DFA. Purposed work uses same approach first it convert regular expression to NFA using Thompson algorithm[16] then convert NFA to DFA by using subset construction Algorithm[17] [18].Figure 3 is example of a DFA for regular expression "(abc+)+" ,to draw DFA from RE we need two algorithms, first is Thompson algorithm and second one is subset construction algorithm .figure 3.a shows the NFA which is the result of Thompson algorithm and figure 3.b shows the final DFA which is the result of sub set construction algorithm. Figure 8 is the proposed model for regular expression matching on GPU.The task divide in two section CPU and GPU and shows no of operation Perform on both .Initially all pattern are compile to DFA state table .the compilation process is done offline by the cpu,usually at start of application. Then state table is copied and store to the memory space of GPU.at the searching phase each thread search different position of input data. If pattern is match then index position of that pattern is stored and result are send back to CPU.



(a) (b)

Fig 3.The NFA state machine(a) and The DFA state machine(b) for regular expression(abc+)+

E. Optimized Aho-corasickAlgorithm :

Input : DFA state transition Table, Set of patterns $\{P_1,P_2,P_3..P_n\}$, Input Text T.

Output: Locations where the patterns occur In T

- Begin
- calculate index location as index=blockDim.x *blockIdx.x+threadIdx.x
  if (index < no of packets){
    - get the length of packet as plen=packets[index].len
    - set cursor=0
      while(cursor <plen){

```
if(nextState!=null && DFA[index]!=null){
      matches_state=nextstate-1
      matches_location=cursor
      break;
       }
if(match[nextstate]!=null)
nextstate ++
else
nextstate =matcher[nextstate].failure
   }
}
```

- end

V. RESULT

We test the performance of Aho-Corasick serial implementation on Intel Ci5 with 8gb RAM and parallel Aho-Corasick algorithm using CUDA GTX 680 graphics card on Ci5 computer with variable number of packets and having same packet length. After this we check the performance of Optimized Aho-Corasick algorithm against optimized and Serial Aho-Corasick algorithm with different number of pattern and having same number of packet.

Table I:: serial vs parallel Aho-corasick time comparison

Number of Packet	Serial Algorithm	Parallel Algorithm
500	7.971ms	59.21us
5000	9.603ms	1.995ms
50000	796.228ms	16.014ms

Table II: :Aho-Corasick Serial vsUn-optimized parallel vsParallel on CUDA time performance table

No of pattern	Pattern MinLength	Serial Algorithm	Parallel Un-optimized	Parallel Optimized	No of Packets
100	2	1.422ms	86.672us	95.393us	500
100	20	9.012ms	6.834ms	5.77ms	500
1000	2	3.267ms	879.232us	92.96us	500
5000	2	2.919ms	85.56us	42.81us	500

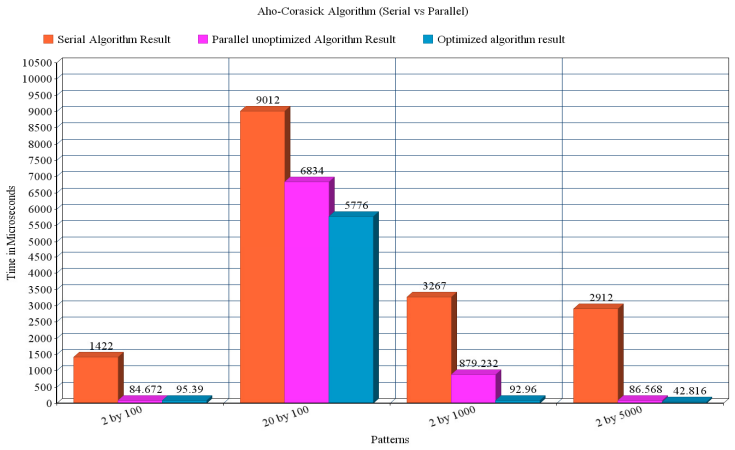


Fig 4 :Aho-Corasick algorithm time chart(serial algorithm vs Parallel Un-optimized vs Optimized parallel)

ACKNOWLEDGMENT

We gratefully acknowledge the support of NVIDIA Corporation with the donation of the Kepler GTX 680 GPU for this research.

CONCLUSION

GPU provide high computing power then CPU. In this paper we present an efficient DFA implementation of both String searching and regular expression matching using GPU architecture .we evalut our proposed algorithm on different memory hierarchy provided by the GPU. we also uses several optimization technique for efficiently implementing matching algorithm. Parallel Implementation of Aho-Corasick algorithm reduces the time required for pattern matching on large datasets.. 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/.
[9]. 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, 200
[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 Processing 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.
[17]. John E. Hopcroft and Jeffrey D. Ullman, Introduction to Automata Theory, 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.

AUTHOR'S PROFILE

Mr.Gaurav K.Bhamare is post graduate student of computer engineering at K.K Wagh college of engineering ,Nashik under University of Pune.he complete his undergraduate course of engineering from Pune University. His areas of interest include Parallel computing ,Image processing .

Prof Satish S. Banait completed his post-graduation from Government Engineering college Aurangabad,BAMU university Maharashtra. Presently he is working at K.K.Wagh college of engineering ,Nashik,Maharashtra,India as a Assistant professor .He has presented papers at National and International conferences and also published paper in national and international journals on various aspect of the computer engineering and Data mining and Data Warehouse .His research of interest include Data Warehouse,Software testing and quality assurance ,Software design and architecture,parallel computing.