Journal of Korea Multimedia Society Vol. 19, No. 2, February 2016(pp. 411-417)
http://dx.doi.org/10.9717/kmms.2016.19.2.411

A Case Study of the Base Technology for the Smart Grid Security: Focusing on a Performance Improvement of the Basic Algorithm for the DDoS Attacks Detection Using CUDA

Jun-Ho Huh[†], Kyungryong Seo[†]

ABSTRACT

Since the development of Graphic Processing Unit (GPU) in 1999, the development speed of GPUs has become much faster than that of CPUs and currently, the computational power of GPUs exceeds CPUs dozens and hundreds times in terms of decimal calculations and costs much less. Owing to recent technological development of hardwares, general-purpose computing and utilization using GPUs are on the rise. Thus, in this paper, we have identified the elements to be considered for the Smart Grid Security. Focusing on a Performance Improvement of the Basic Algorithm for the Stateful Inspection to Detect DDoS Attacks using CUDA. In the program, we compared the search speeds of GPU against CPU while they search for the suffix trees. For the computation, the system constraints and specifications were made identical during the experiment. We were able to understand from the results of the experiment that the problem-solving capability improves when GPU is used. The other finding was that performance of the system had been enhanced when shared memory was used explicitly instead of a global memory as the volume of data became larger.

Key words: ICT, CUDA, Smart Grid Network Intrusion Detection System, Java, GPU.

1. INTRODUCTION

The 'String Matching Algorithm' is one of the longest and the most prevalent issue in the computer science field. Today, we can observe the applications related to the string matching in many areas [1].

The string matching algorithm has a long history in the computational biology and it's begun with finding the similar proteins and gene sequences in the DNA database of gene sequences. The development of the technology to seek out similar sequences faster was motivated by an explosive increase in the data volume of available gene sequences during the 80's and 90's. Eventually, the parallel string matching algorithm and CUDA that utilize elaborate data structure called the suffix tree are bing used [2].

Meanwhile, the graphic cards mainly handle arithmetic operations of graphic-related floating-points required by computers. Since the introduction of 3D Voodoo graphic card in 1994 by 3DFX Interactive, the graphic processors continue to make rapid strides owing to the popularity of the multimedia and gaming fields.

Moreover, the developing speed of GPU's (Graphic Processing Units) has been much faster than that of CPU's since their development in 1999 and currently [3-8], the floating-points processing ability of GPU's is dozens or hundreds of times

※ Corresponding Author : Kyungryong Seo, Address: DSLab, Pukyong National University at Daeyeon, 45, Yongso-ro, Nam-gu, Busan, Republic of Korea, TEL : +82-51-629-6262, E-mail : krseo@pknu.ac.kr
Receipt date : Jan. 22, 2016, Approval date : Jan. 30. 2016

[†] Senior Research Engineer of SUNCOM Co., Republic of Korea (E-mail : 72networks@pukyong.ac.kr)
[††] Department of Computer Engineering, Pukyong National University at Daeyeon

faster than CPU's and costs less. Proposed program was implemented with Java. This paper is organized as follows: the related researches are described in Chapter 2; the experimental setup in Chapter 3; the performance times of CPU and GPU to run the implemented program are compared and discussed in Chapter 4; and finally, the conclusion and future work in Chapter 5.

2. RELATED RESEARCH

2.1 CUDA

CUDA (Compute Unified Device Architecture) is the GPU processing tool developed by Nvidia to conduct general purpose computing.

The CUDA program is composed with more than one phase and each phase is performed on the device such as the host (CPU) or GPU. The phase where there's almost no data parallelism exists will be implemented with the host code. On the other hand, if there are ample parallelism, the phase will be implemented with the device code. That is, the CUDA program is a source code in which the host code and device code have been integrated.

CUDA maintains the GPU memories by distinguishing them as the on-chips or the off-chips. Such memory-maintenance largely affect the composition of grids, blocks and threads that determine the program execution methods. It's preferable to use the on-chip memories in order to reduce transmission delay by the memory and to improve the execution speed, and the key is to maintain a small number of the on-chip memories efficiently. This is the opposite of the case where there are plenty of memories but the computing speed is relatively.

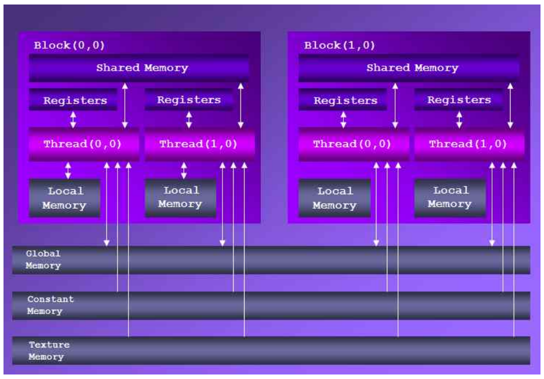


Fig. 1. Memory architecture of CUDA [4].

2.2 String Matching Algorithm

There are times when one wishes to find a certain word while working on the documents. It can be found right away inputting the data that he/she wants to look for using 'find'. In other words, the sting matching is a function that locates the desired word(s) in a given document. The string matching is being applied in the document processing field, as well as in the fields of pattern and voice recognitions, DNA sequencing analysis, etc.

2.3 BF (Brute Force) Algorithm

The BF algorithm is a string detection algorithm in which the pattern-match is examined for all conceivable locations and it does not require any pre-processing for the pattern recognition. There's a window on which the texts have been partitioned and those texts will be compared to the pattern. While the BF algorithm is easy to comprehend, there's a downside of taking more time to find the right pattern. An example is given as below Fig. 2, where the blocks indicate the comparisons made.

2.4 Quick Search Algorithm

The quick search algorithm was proposed by Sunday [6] and it uses worse-letter heuristic between the two by modifying it a little. However, this algorithm so simple that it shows quite an excellent performances.

As shown in Fig. 3, when a mismatching letter (i.e., x and a in the example) is found during the matching process, a new letter located further right to the pattern (i.e., b in the example) will be used and it moves to the far right location where the new letter appears in the patterns, subsequently

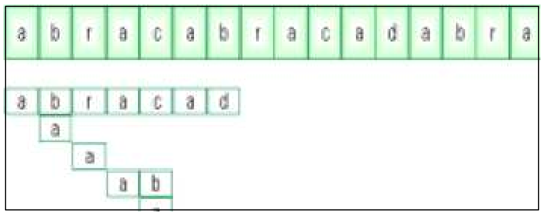


Fig. 2. An example of BF algorithm.

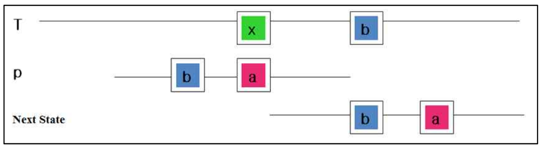


Fig. 3. Matching process of Quick Search algorithm.

resuming the comparisons.

To find a string "abracad" in the string "abra-cabracadabra" with the BF algorithm, the first string pattern "abracab" will be compared with the forementioned first string "abracad". If they match, the second string "bracad" and the second pattern "bracab" will be compared. If these do not match, then the first string "abracad" and the second pattern "bracab" will be compared. This process continues until the right string appears and is designated as the BF algorithm.

2.5 Intrusion Detection Systems (Firewall)

The types of intrusion detection systems (firewall) can be classified into Packet-filtering, Application Gateway, Stateful Inspection and Hybrid Firewall, depending on their operating modes.

Fig. 6. Nvidia Quadro Fx3800 used in this research.

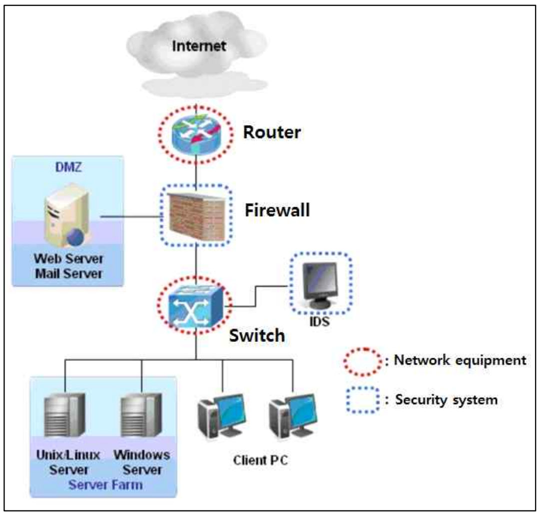


Fig. 4. An example of corporate network structure [9].

A base technology has been implemented in this paper to propose a method that utilizes CUDA focusing on the stateful inspection and considering cost-saving aspects in a multi-channel environment using GPU as a coprocessor for the CPU as it has more cores compared to CPU commonly used for the general-purpose computers.

Fig. 4 is an example of a corporate network structure using (stateful inspection method) as its

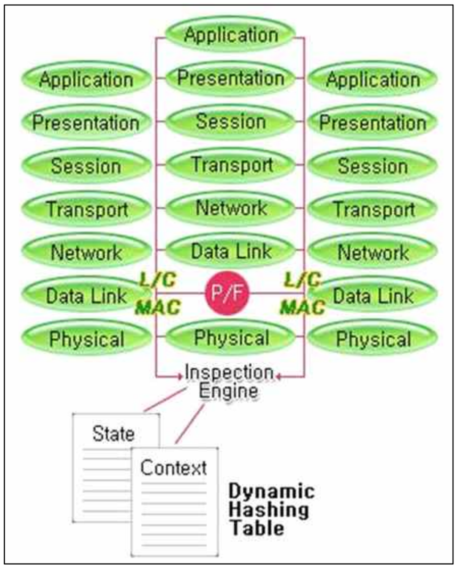
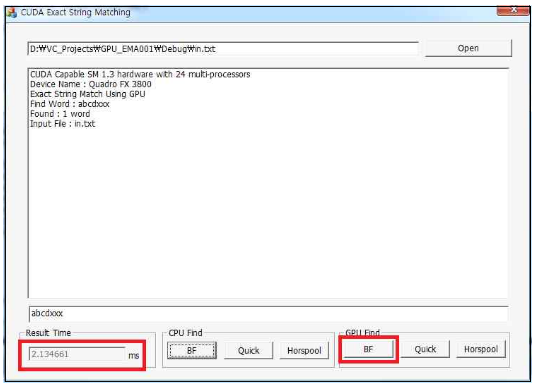


Fig. 5. A stateful inspection method.



```
D:₩VC_Projects₩GPU_EMA001₩Debug₩in.txt
```

```
CUDA Capable SM 1.3 hardware with 24 multi-processors
Device Name : Quadro FX 3800
Exact String Match Using GPU
Find Word : abcdxxx
Found : 1 word
Input File : in.txt
```

```
abcdxxx
```

Result Time: 2.134661 ms

Fig. 8. Computation using GPU.

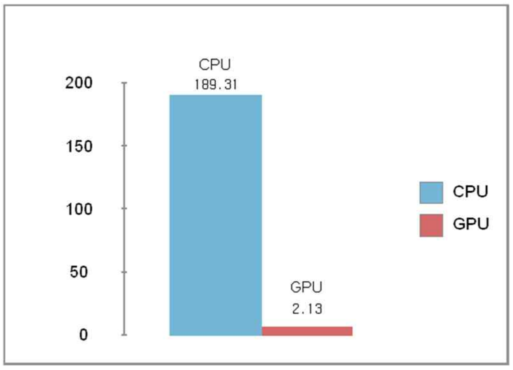


Fig. 9. Performance comparison between CPU and GPU.

firewall. Fig. 5 illustrates the stateful inspection method which shows a practical way to overcome the limitations of previous firewalls by providing an overall situational context for all the layers and maintaining a Client-Server model, using the packet-filtering method as its base. This method provides a more perfect inspection capability for the data and an additional session-tracking function compared to the packet-filtering such that the connectionless protocols such as UDP, RPC and others can be checked.

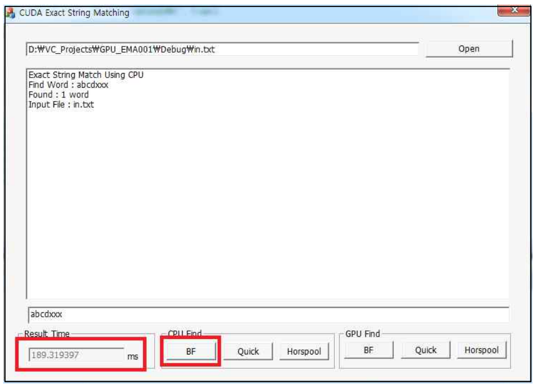
3. EXPERIMENTAL ENVIRONMENT

The experimental setup is as in the following. For the computing process, All the system restrictions and specifications have been made identical except CPU's and GPU's. Intel Core i5 CPU, 4GB RAM, Nvidia Quadro Fx3800, and Widow 7 OS were used for the experiment. Fig. 6 shows Nvidia Quadro Fx3800 used in this research. Especially, the graphic card used here has 192 multiprocessors and can have 192 warps at the same time.

Meanwhile, among the graphic cards which support CUDA, Nvidia GeForce 8800GTX has 16 multiprocessors whereas GeForce GTX 280 has 30. A single multiprocessor can execute 32 threads at a time physically and this unit is called a 'warp'. The threads within the warp carry out the same command. Also, a single multiprocessor can simultaneously have up to 24 warps(Nvidia GeForce 8800GTX) and 32 warps (GeForce GTX 280), respectively. Actually, only one warp can process computing at a time but all of the warps placed on the multiprocessors are in the active state so that while one of the warps executes the computation, others work on the tasks like waiting for the data from the global memory, etc. Additionally, only one block can be performed on a single multiprocessor such that the execution process cannot be shared among several multiprocessors.

4. BASIC ALGORITHM FOR THE STATEFUL INSPECTION TO DETECT DDoS ATTACKS USING CUDA

In the string matching, the suffix tree could be



```
D:₩VC_Projects₩GPU_EMA001₩Debug₩in.txt
```

```
Exact String Match Using CPU
Find Word : abcdxxx
Found : 1 word
Input File : in.txt
```

```
abcdxxx
```

Result Time: 189.319397 ms

Fig. 7. Computation using CPU.

constructed time-proportionally against the length of the corpus and provides time-proportional and exact matching result against the length of the query regardless of the size of the corpus.

In this experiment, we've implemented a Basic algorithm for the stateful inspection to detect DDoS attacks Using CUDA and compared the search speeds between CPU and GPU used.

As shown in Fig. 7 below, it takes about 189.31ms when the 100kb text file is checked with the suffix tree searching the BF computation algorithm which has utilized CPU whereas, as in Fig. 8, it took approximately 2.13ms as the same method has been applied using GPU.

GPU computation time shows much difference compared to that of CPU and we can observe that the computation utilizing GPU is more efficient for the large volume searching, even the algorithm is a simple one. Fig. 9 shows Performance comparison between CPU and GPU. Also, as a result of the suffix tree searching speed comparison between CPU and GPU, we recognized that the computing speed of GPU was roughly 89 times faster and showed more efficiency.

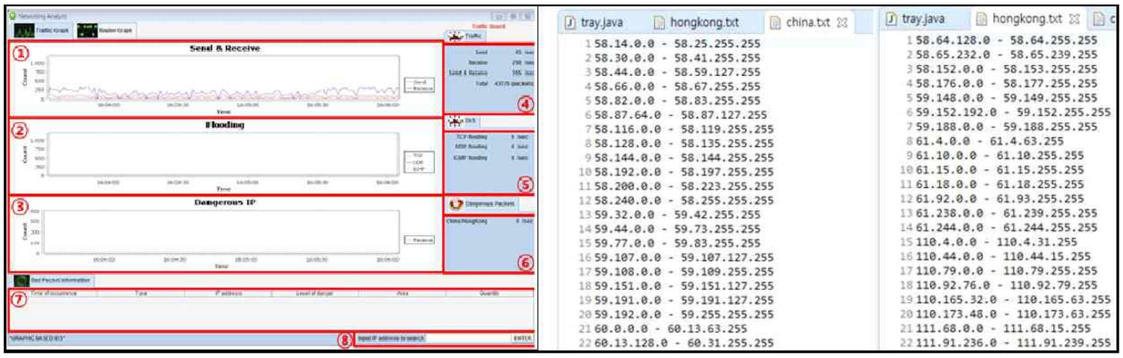
5. CONCLUSION AND FUTURE WORK

It's will be quite efficient to use GPU computation if anyone wish to attempt a large-scale and high-performance computation with less cost unless data is small so that computing work is not complicated. Due to the characteristics of the GPU computation process, the data on the main memory must be copied by the graphic card memory and subsequently copied by the main memory again so that we needed to consider overheads.

As the results of experiment show, it was clear that the problem-solving capability had improved when GPU was used. By using the shared memory explicitly instead of the global memory, system's computing performance can be improved in proportion to data volume increase. Based on this implementation, We are to design and implement a light-weight Smart Grid intrusion detection system based on this implementation as our future task.

APPENDIX

The draft of experiment in this paper is the product of the Team Project carried out at the "Mobile Multimedia" class (Graduate school curriculum) which was led by Prof. Young-Bong Kim of Dept. of IT Convergence and Application Engineering Pukyong National University at Daeyeon. In this research paper, performance evaluation and additional descriptions together with more experiments have been conducted for the draft. We'd like to express our gratitude to Prof. Young-Bong Kim and the Ph.D student Hongseok Choi and Ph.D student Sugarbayar Otgonchimeg and (Prof. Munkhbaatar Doyoddorj) for their in-



```
58.14.0.0 - 58.25.255.255
58.30.0.0 - 58.41.255.255
58.44.0.0 - 58.59.127.255
58.66.0.0 - 58.67.255.255
58.82.0.0 - 58.83.255.255
58.87.64.0 - 58.87.127.255
58.116.0.0 - 58.119.255.255
58.128.0.0 - 58.135.255.255
58.144.0.0 - 58.144.255.255
58.192.0.0 - 58.197.255.255
58.200.0.0 - 58.223.255.255
58.240.0.0 - 58.255.255.255
59.32.0.0 - 59.42.255.255
59.44.0.0 - 59.73.255.255
59.77.0.0 - 59.83.255.255
59.107.0.0 - 59.107.127.255
59.108.0.0 - 59.109.255.255
59.151.0.0 - 59.151.127.255
59.191.0.0 - 59.191.127.255
59.192.0.0 - 59.255.255.255
60.0.0.0 - 60.13.63.255
60.13.128.0 - 60.31.255.255
```

```
58.64.128.0 - 58.64.255.255
58.65.232.0 - 58.65.239.255
58.152.0.0 - 58.153.255.255
58.176.0.0 - 58.177.255.255
59.148.0.0 - 59.149.255.255
59.152.192.0 - 59.152.255.255
59.188.0.0 - 59.188.255.255
61.4.0.0 - 61.4.63.255
61.10.0.0 - 61.10.255.255
61.15.0.0 - 61.15.255.255
61.18.0.0 - 61.18.255.255
61.92.0.0 - 61.93.255.255
61.238.0.0 - 61.239.255.255
61.244.0.0 - 61.244.255.255
110.4.0.0 - 110.4.31.255
110.44.0.0 - 110.44.15.255
110.79.0.0 - 110.79.255.255
110.92.76.0 - 110.92.79.255
110.165.32.0 - 110.165.63.255
110.173.48.0 - 110.173.63.255
111.68.0.0 - 111.68.15.255
111.91.236.0 - 111.91.239.255
```

Fig. 10. Sample of light-weight Smart Grid Network Intrusion Detection System'GUI [10-11].

volvement in the Team Project. Our gratitude extends to 5 alumni who remained anonymous during the Team Project Presentation.

The second draft of this paper was presented Poster Session in Fall Conference of Korea Multimedia Society, Nov 6-7 (2015)[12]. I am grateful to 2 anonymous commentators who have contributed to the enhancement of the paper's completeness with their valuable suggestions at the Conference.

REFERENCES

[1] J. Peng and Hu Chen, "A GPU-based High Performance Multi-string Matching System," *Proceeding of IEEE International Conference on Future Computer and Communication*, Vol. 1, pp. 66-81, 2010.

[2] M. Schatz and C. Trapnell, "Fast Exact String Matching on the GPU," *Citeseer*, pp. 1-6, 2007.

[3] C.H. Lin, S.Y. Tsai, C.H. Liu, and S.C. Chang, "Accelerating String Matching Using Multi-Threaded Algorithm on GPU," *Proceeding of IEEE Global Telecommunications Conference,* pp. 1-5, 2010.

[4] NVIDIA, *NVIDIA CUDA Programming Guide 2.0*, NVIDIA Corporation, 2008.

[5] D. Knuth, J. Morris, and V. Pratt, "Fast Pattern Matching in Strings," *SIAM Journal on Computing*, Vol. 6, No. 2, pp. 323-350, 1977.

[6] D.M Sunday, "A Very Fast Substring Search Algorithm," *Communications of the ACM*, Vol. 33, No. 8, pp. 132-142, 1990.

[7] L. Yang, B.J. Jang, S.H. Lim, K.C. Kwon, S.H. Lee, and K.R. Kwon, "Weather Radar Image Generation Method Using Interpolation Based on CUDA," *Journal of Korea Multimedia Society*, Vol. 18, No. 4, pp. 473-482, 2015.

[8] NVIDIA, *CUDA C programming Guide V6.0*, NVIDIA Corporation, 2014.

[9] Jongsu Park, http://m.dbguide.net/about.db?cmd=view&boardConfigUid=19&boardUid=125803 (accessed Jun., 7, 2006).

[10] J.H. Huh, M.H. Hong, J.M. Lee, and K.R. Seo, "Implementation of DDoS Botnet Detection System On Local Area Network," *Journal of Korea Multimedia Society*, Vol. 16, No. 6, pp. 678-688, 2013.

[11] J.H. Huh, D.H. Lee, and K.R. Seo, "Implementation of Graphic Based Network Intrusion Detection System for Server Operation," *International Journal of Security and Its Applications*, Vol. 9, No. 2, pp. 37-48, 2015.

[12] J.H. Huh, N.J. Kim, and K.R. Seo, "Implementation of String Matching Program for Finding Query Strings Using CUDA," *Proceedings of Fall Conference of the Korea Multimedia Society*, Vol. 18, No. 2, pp. 688-691, 2015.

Jun-Ho Huh

finished the Cooperative Marine Science and Engineering Program, Texas A&M University at Galveston, United States of America in Aug. 2006.
Received B.S. in Science Degree from Department of Major of Applied Marine Sciences [Marine Aquaculture, Oceanography, Marine Life Sciences (Currently Faculty of Marine Biomedical Sciences)], B.S. in Engineering Degree (Double Major) from Department of Major of Computer Engineering from Jeju National University at Ara, Jeju, Republic of Korea in Aug. 2007.
M.A. in Education Degree from Department of Major of Computer Science Education, Graduate School of Education, Pukyoug National University at Daeyeon, Busan, Republic of Korea in Aug. 2012.
Ph.D. in Engineering Degree from Department of Major of Computer Engineering, Graduate School, Pukyoug National University at Daeyeon, Busan, Republic of Korea in Feb. 2016. He received the Best Paper Award from Korea Multimedia Society twice (Nov. 2014, May. 2015). His research interests are Green IT, Smart Grid, Network Security, Curriculum of Computer, High Availability Computing.

Kyungryong Seo

received B.S. in Engineering Degree from Department of Major of Electrical Machinery Engineering from Pusan National University, Busan, Republic of Korea in Feb. 1983.
M.S. in Engineering Degree in Electrical Engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea in Feb. 1990.
Received the Ph.D. Engineering Degree in Electrical Engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Republic of Korea in Aug. 1995.
Currently he is a Full Professor of Computer Engineering Departments, Pukyong National University at Daeyeon, Busan, Republic of Korea.
His research interests are High Speed Computer Network, Network Security, High Availability Computing.