科技部補助專題研究計畫成果報告
期末報告

使用高度平行演算法加速開源的網路入侵偵測系統SNORT

計 畫 類 別 ： 個別型計畫
計 畫 編 號 ： MOST 106-2221-E-003-012-
執 行 期 間 ： 106年08月01日至107年07月31日
執 行 單 位 ： 國立臺灣師範大學電機工程學系（所）

計 畫 主 持 人： 林政宏
共 同 主 持 人： 張世杰
計畫參與人員： 碩士班研究生-兼任助理：謝政宏
碩士班研究生-兼任助理：林永鑫
碩士班研究生-兼任助理：李易穎
碩士班研究生-兼任助理：吳振豪
碩士班研究生-兼任助理：謝亦信

報 告 附 件 ： 出席國際學術會議心得報告

中 華 民 國 107 年 09 月 17 日

中 文 摘 要 ： 目前網路入侵偵測系統大多採用多重樣式字串比對的方式，是否含有網路攻擊與異常的封包，透過比對數以千計的攻擊特徵來偵測封包內容。隨著大數據時代的來臨，網路速度與攻擊活動的增加，多重樣式字串比對面臨效能與吞吐量的不足，導致許多封包沒有處理且遺失。為了改善網路入侵偵測系統的效能與吞吐量，本計畫提出階層式平行架構，利用多張圖形處理器(Multi-GPU)與三種不同層面的平行技術加速網路入侵偵測系統。
階層式平行架構由三層不同的平行技術所組成，從上至下來看，第一層實現資料平行(Data Parallelism)於多張圖形處理器；第二層將管線化排程(Pipeline Schedule)實現於個別圖形處理器中，屬於任務平行(Task Parallelism)；第三層則是採用資料平行的技術，優化 Aho-Corasick 演算法。本計畫實驗結果顯示，採用四張圖形處理器 Nvidia Titan X 實現於階層式平行架構，總系統吞吐量可高達 70 Gbps，與傳統使用於Snort 中的 Aho-Corasick 演算法相比，可高達四十倍的改善倍率。當圖形處理器的數量增加，總系統的吞吐量也會隨之增加。除此之外，本計畫採用完美雜湊(Perfect Hashing)的方法，壓縮傳統 Aho-Corasick 的狀態機，減少在Snort 中99.2%多重樣式字串比對的記憶體使用量，最後本計畫文將提出的階層式平行架構實現於開源網路入侵偵測系統 Snort。

中 文 關 鍵 詞 ： 網路入侵偵測系統、圖形處理器、多重樣式字串比對、Aho-Corasick 演算法

英 文 摘 要 ： Multi-string matching has been widely used in NIDS to detect network attacks and malicious network packets by matching packet contents with thousands of attack patterns. Due to the rapid increase of growing network attacks and network speeds, multistring matching faces the challenges for limited performance and insufficient throughput. In order to improve the performance and throughput of multi-string matching, this thesis presents a novel hierarchical parallelism that can accelerate multi-string matching on multiple GPUs. The hierarchical parallelism consists of three layers of parallelism. From top to bottom, the first layer is the data parallelism on multiple GPUs; The second layer is the task parallelism on a single GPU; The last layer is the data parallelism of the Aho-Corasick algorithm. Experimental results show that the hierarchical parallelism on a machine featured with four Nvidia Titan X GPUs can achieve 70 Gbps of throughput which is 40 times faster than the Aho-Corasick algorithm used in Snort. As the number of GPUs increase, the throughput of the hierarchical parallelism will also increase. In addition, the proposed approach adopts the perfect hashing to construct state machines that can achieve memory reduction on Snort up to 99.2%. Finally, the proposed hierarchical parallelism is implemented in the open source network intrusion detection system using Snort.

英文關鍵詞：Network intrusion detection systems, graphics processing units, multiple string matching, Aho-Corasick algorithm.

科技部補助專題研究計畫成果報告

（期末報告）

使用高度平行演算法加速開源的網路入侵偵測系統SNORT

計畫類別：個別型計畫
計畫編號：MOST 106-2221-E-003 -012
執行期間：106 年 8 月 1 日至 107 年 7 月 31 日

執行機構及系所：國立台灣師範大學電機工程學系

計畫主持人：林政宏副教授
共同主持人：張世杰教授
計畫參與人員：謝政宏、林永鑫、謝亦信、吳振豪、李易穎

成果報告類型(依經費核定清單規定繳交)：■精簡報告　□完整報告

本計畫除繳交成果報告外，另須繳交以下出國心得報告，共一份：
■出席國際學術會議心得報告

中 華 民 國 107 年 七 月 卅一 日

一、 中文摘要

目前網路入侵偵測系統大多採用多重樣式字串比對的方式，是否含有網路攻擊與異常的封包，透過比對數以千計的攻擊特徵來偵測封包內容。隨著大數據時代的來臨，網路速度與攻擊活動的增加，多重樣式字串比對面臨效能與吞吐量的不足，導致許多封包沒有處理且遺失。為了改善網路入侵偵測系統的效能與吞吐量，本計畫提出階層式平行架構，利用多張圖形處理器(Multi-GPU)與三種不同層面的平行技術加速網路入侵偵測系統。

階層式平行架構由三層不同的平行技術所組成，從上至下來看，第一層實現資料平行(Data Parallelism)於多張圖形處理器；第二層將管線化排程(Pipeline Schedule)實現於個別圖形處理器中，屬於任務平行(Task Parallelism)；第三層則是採用資料平行的技術，優化 Aho-Corasick 演算法。本計畫實驗結果顯示，採用四張圖形處理器 Nvidia Titan X 實現於階層式平行架構，總系統吞吐量可高達 70 Gbps，與傳統使用於Snort 中的 Aho-Corasick 演算法相比，可高達四十倍的改善倍率。當圖形處理器的數量增加，總系統的吞吐量也會隨之增加。除此之外，本計畫採用完美雜湊(Perfect Hashing)的方法，壓縮傳統 Aho-Corasick 的狀態機，減少在Snort中99.2%多重樣式字串比對的記憶體使用量，最後本計畫文將提出的階層式平行架構實現於開源網路入侵偵測系統 Snort。

關鍵字：網路入侵偵測系統、圖形處理器、多重樣式字串比對、Aho-Corasick 演算法

二、 英文摘要

Multi-string matching has been widely used in NIDS to detect network attacks and malicious network packets by matching packet contents with thousands of attack patterns. Due to the rapid increase of growing network attacks and network speeds, multistring matching faces the challenges for limited performance and insufficient throughput. In order to improve the performance and throughput of multi-string matching, this thesis presents a novel hierarchical parallelism that can accelerate multi-string matching on multiple GPUs. The hierarchical parallelism consists of three layers of parallelism. From top to bottom, the first layer is the data parallelism on multiple GPUs; The second layer is the task parallelism on a single GPU; The last layer is the data parallelism of the Aho-Corasick algorithm. Experimental results show that the hierarchical parallelism on a machine featured with four Nvidia Titan X GPUs can achieve 70 Gbps of throughput which is 40 times faster than the Aho-Corasick algorithm used in Snort. As the number of GPUs increase, the throughput of the hierarchical parallelism will also increase. In addition, the proposed approach adopts the perfect hashing to construct state machines that can achieve memory reduction on Snort up to 99.2%. Finally, the proposed hierarchical parallelism is implemented in the open source network intrusion detection system using Snort.

Keywords: ***Network intrusion detection systems, graphics processing units, multiple string matching, Aho-Corasick algorithm.***

三、 計畫緣由與目的

Due to the increasing popularity of the Internet today, network security is becoming more and more important. Network Intrusion Detection Systems (NIDS) are designed to monitor malicious activity or policy violations on networks or systems to detect malicious network attacks. Because of the exponential growth of the number and type of network attacks, the load of the detection system is getting larger as the rule set increases. At the same time, advances in hardware have also increased network bandwidth, and NIDS must improve the overall performance of the system to meet current network requirements. Snort [2] is a powerful and lightweight open source network intrusion detection system with good scalability and portability. It has the ability to analyze instantaneous network traffic, log packets on the network, perform analysis of network protocols including TCP, UDP, and ICMP, and search for attack patterns. It can detect a variety of attacks, real-time alerts on attacks such as buffer overflows, port scans, CGI attacks, SMB probes, and attempts to probe fingerprints of the operating system. Since its publication in 2002, Snort has used the Aho-Corasick (AC) [1] algorithm to match multiple string patterns. According to OSEC (Open Security Evaluation Criteria) verification report [3], Snort's performance is about 750 Mbps, of which the execution time of multiple string matching accounted for 41.68%. In other words, Snort faces a big challenge to accommodate the increasing number of attack patterns and meet the network speed requirements. Therefore, many hardware approaches have been proposed, including FPGA [4] [10], ASIC [5], and GPUs [6] to improve the performance of Snort. In our past researches, we have had some preliminary results in accelerating string matching using GPUs. We propose the Parallel Failureless Aho-Corasick (PFAC) [7] [8] algorithm to accelerate the traditional Aho-Corasick algorithm using GPUs. We also propose using a perfect hash to compress the state transition table and achieve significant memory reduction. In this project, we extend our preliminary study and propose a hierarchical parallelism mechanism to optimize Snort and reduce its memory requirements. The hierarchical parallelism consists of three layers of parallelism. From top to bottom, the first layer is the data parallelism on multiple GPUs, the second layer is the task

parallelism on a single GPU, and the third layer is the data parallelism of the Aho-Corasick algorithm. Experimental results show that the hierarchical parallelism on a machine with four Titan X GPUs achieves up to 70 Gbps of throughput, more than 40 times faster than the traditional AC algorithm used in Snort. As the number of GPUs increases, the throughput of hierarchical parallelism will increase. In addition, Snort uses linked lists to construct state machines, leading to inefficient traversal of the AC state machines and memory requirements for storing pointers. We propose to use perfect hashing [9] to store the state machines and achieves up to 99.2% of memory reduction on Snort. Finally, the proposed hierarchical parallelism is implemented in Snort, an open source network intrusion detection system

四、 研究方法

The original Snort processes one packet at a time, and all packets are processed sequentially. This strategy does not apply to GPUs as it is not cost-effective to process single and small data packets on GPUs. Since GPUs can handle large amounts of data at the same time, we propose that batch technology put the packets in buffers and then process each buffer on a batch basis. In order to achieve batch processing, we propose a hierarchical parallelism architecture which adopts multiple GPUs to optimize Snort with data and task parallelism. Fig. 1 shows the proposed hierarchical parallelism architecture which consists of three levels of parallelism.

4.1 Data parallelism on multiple GPUs

The first level is the data parallelism on multiple GPUs where each data buffer is allocated to its associated GPU. Different protocol network packets are sent to a specific buffer. In this stage, we record the order of the packets, the location of the packet, to facilitate the subsequent post-processing action. When the buffer is full or the maximum elapsed time is up, the contents of the buffer will be moved to the GPU. To meet real-time requirements, the buffer size is set to 300 MB and the maximum elapsed time is set to 0.45 seconds.

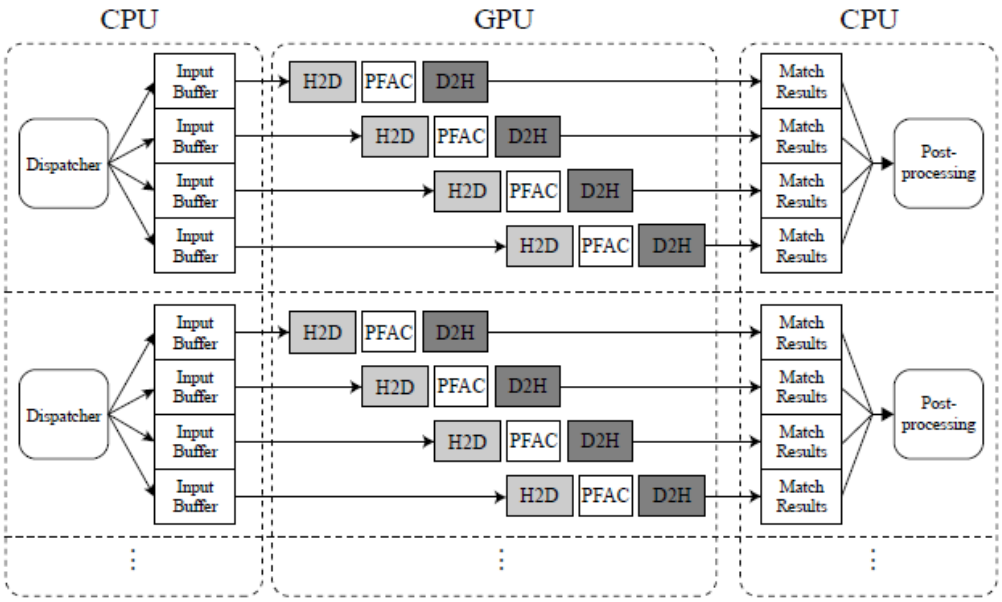

Figure 1. The hierarchical parallelism mechanism

4.2 Task parallelism on a single GPU

The second level is the task parallelism which implements pipelining to utilize the resources of GPUs. A typical GPU process consists of three tasks, moving data from CPU to GPU, GPU kernel function, and moving data from GPU to CPU. CUDA provides a streaming technology that splits the input into multiple streams and sends them one by one to the GPU, which is so called pipelining.

4.3 Data parallelism of the Aho-Corasick algorithm

Snort adopts the Aho-Corasick algorithm to match thousands of attack patterns in parallel. The third level is the data parallelism of the Parallel Failure-less Aho-Corasick (PFAC) [7][8] algorithm on GPUs. PFAC is a multi-threaded algorithm for enhancing the Aho-Corasick algorithm. In Snort, attack patterns are grouped by their protocols and the patterns in the same group are compiled into a state machine. An Aho-Corasick state machine contains both valid and failure transitions. For example, Fig. 2 shows the Aho-Corasick state machine used to match the four patterns of “he”, “she”, “his”, and “her”, where solid lines represent valid transitions and dashed lines represents failure transitions. Then, multiple string matches are done by traversing the Aho-Corasick state machine. Take the input string “shers” as an example. The state machine starts from the initial state

0. After obtaining “s”, the state machine enters state 3. After that, the input character “h” is acquired, and the state machine moves from state 3 to state 4. Next, the input character “e” is obtained and the state machine moves to the final state 5 which represents the pattern “she” is matched. Then, the input character “r” is obtained. Because there is no valid transition for “r”, the state machine moves to state 2 by the failure transition. Meanwhile, state 2 represents the pattern “he” is matched. After reaching state 2, the state machine takes the input character “r” again and moves to state 8. Finally, after getting “s”, the state machine reaches the final state 9 which represents the pattern “hers” is matched.

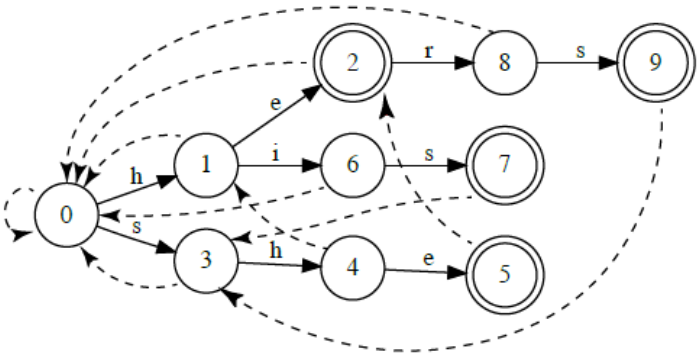

Figure 2. The hierarchical parallelism mechanism

The Aho-Corasick algorithm uses one thread to scan from the beginning of the input string to the end, by traversing the Aho-Corasick state machines. To accelerate the Aho-Corasick algorithm, we have proposed a fine-grained data parallel algorithm called Parallel Failure-less Aho-Corasick (PFAC) algorithm [7][8]. As shown in Fig. 3, the PFAC algorithm assigns a separate thread for each character position, with each thread responsible for traversing a specific PFAC state machine. The PFAC requires n threads for an input string of length n. Each thread only needs to be responsible for matching any of the signature rules that match the beginning of the character at its location. In other words, each thread only needs to traverse valid transitions. When there is no valid transition, the thread terminates immediately without taking failure transitions. Therefore, the traditional Aho-Corasick state machine can be simplified by removing all failure transitions and the loop-back transition in the initial state. Fig. 4 shows the PFAC state machine, which compiles four

patterns, “he”, “her”, ”his” and ”her”. Compared to Fig. 2, all failure transitions as well as the loop-back transition in the initial state are all removed.

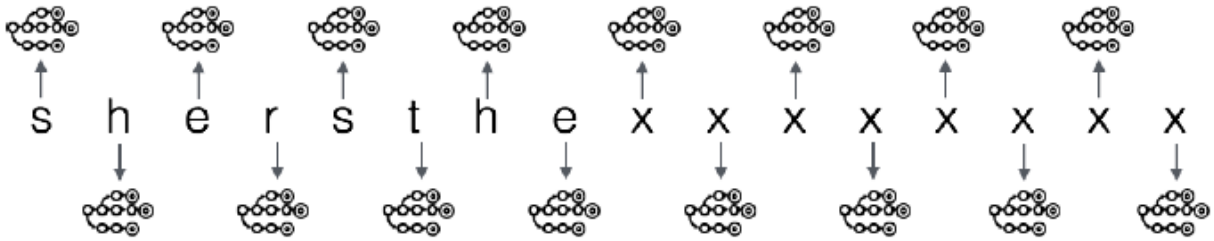

Figure 2. Parallel Failure-less Aho-Corasick (PFAC) Algorithm

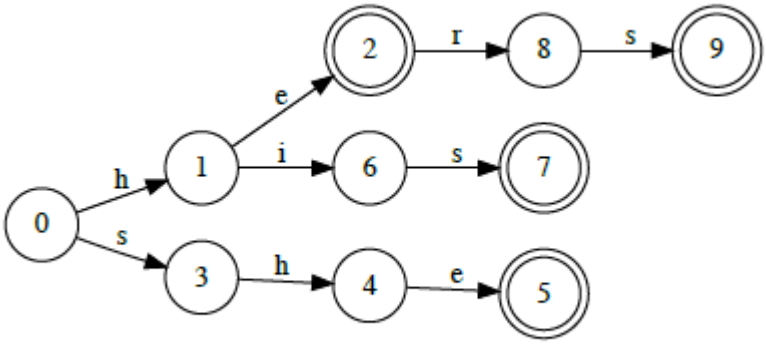

Figure 3. PFAC state machine

Fig. 5 shows an example of using PFAC to find “she”, “he” and “her”. Consider an input string containing the substring“shers”. Thread n is located at the character “s” and thread n+1 is located at the character “h”. After obtaining“s”, “h”, and “e”, thread n will reach state 5 which represents the pattern “she” is found. On the other hand, thread n+1 will reach state 2 after taking “h” and “e” and will reach state 9 after taking “r” and “s”. States 2 and 9 indicate that “he” and “hers” are found, respectively. Different from the traditional Aho-Corasick algorithm, the PFAC adopts multiple threads to search patterns in parallel. PFAC is a fine-grained parallel algorithm that is suitable for implementation on GPUs. Although PFAC initially requires thousands of threads, most threads terminate very early. The time complexity of PFAC is O (1) to O (m) where m is the length of the longest pattern. Therefore, the overall performance is much better than the traditional Aho-Corasick algorithm implemented on CPUs. We have released the source code of PFAC on Github (https://github.com/pfaclib/PFAC).

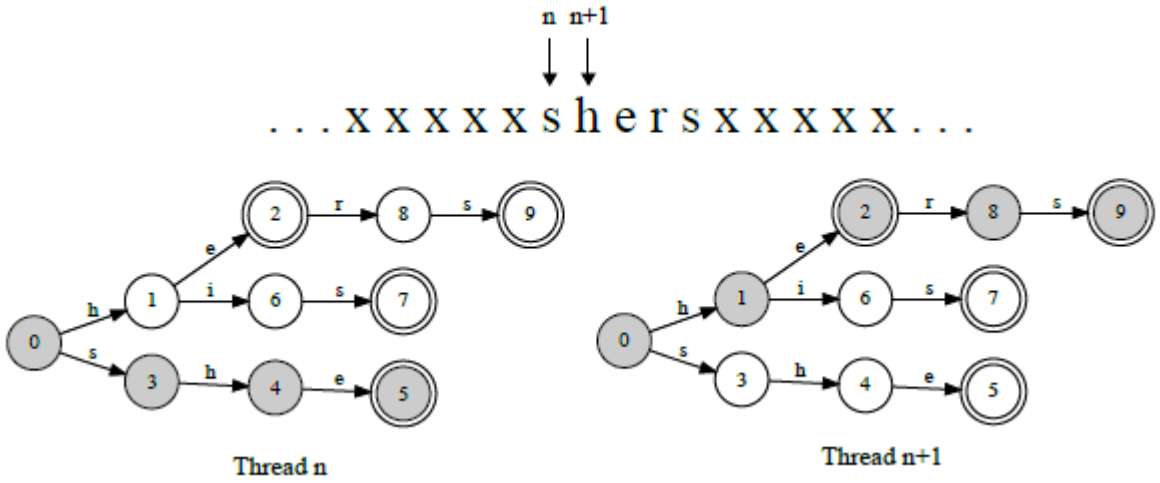

Figure 3. Example of PFAC

4.4 Construction of State Machine

In terms of building the AC state machine, Snort uses linked lists to store the AC state table. There are two disadvantages of the linked lists, one is the inefficiency of traversing the state machine and the other is the memory requirement for storing pointers. In our approach, we use perfect hashing [9] to store the PFAC state machine. Fig. 6 shows an example of using perfect hashing to compress a 2D table. The perfect hashing algorithm first places the 11 valid keys in the 4x8 table in size order. Then, the perfect hashing algorithm moves the four rows to the appropriate places, with no two keys in the same column. The offset of rows are recorded in the row table. Finally, the 2D table can be compressed into a one-dimensional vector stored in the hash key table.

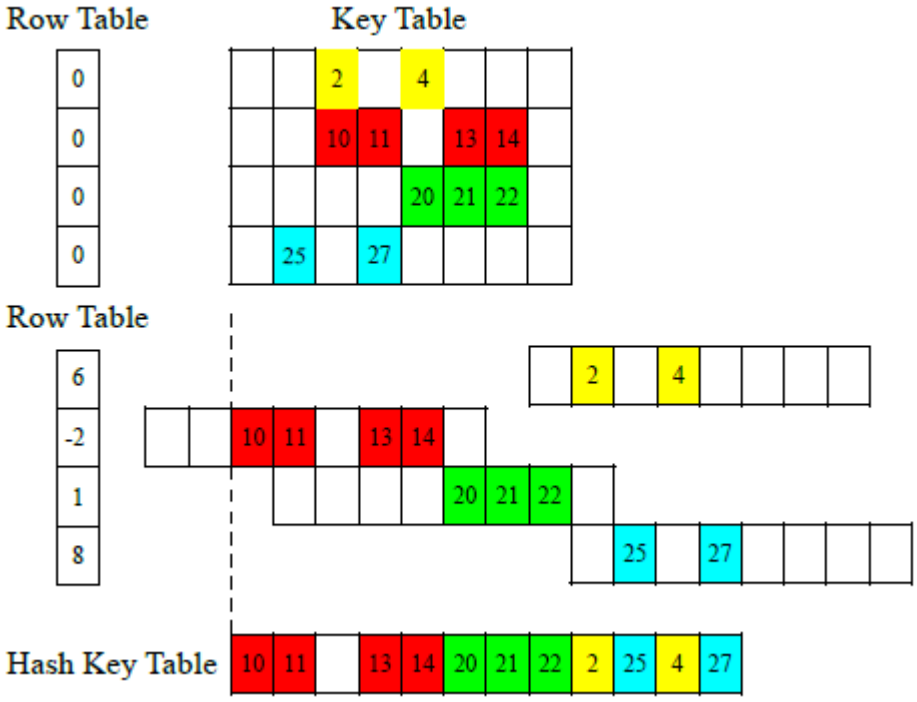

Figure 3. An example of perfect hashing construction

4.5 Post-processing

As discussed in Section 2, Snort processes individual packets, performs string matching through the Aho-Corasick algorithm [1] in the Detect module, and outputs the result as to where the packet has been successfully matched and sends a warning message to the user. Since it is not effective to process a single packet on GPUs, we propose putting packets into buffers and batch processing buffered packets. When the buffer is full or reaches its maximum elapsed time, the batch of packets is moved to GPUs for string matching. Then, the GPU will return match results, reporting the locations of matched patterns. The problem is that given a match pattern location, how to quickly know which packet contains the pattern. In this project, we propose to create a balanced binary search tree to record the packet's ID and its location in the buffer. The time complexity of searching a balanced binary tree is O (log n), where n is the number of nodes.

4.6 Experimental Results

The experimental environment is equipped with a CPU host and 4 GPU devices. The host uses the Intel Xeon E5-2683 CPU, which includes 16 cores operating at 2.1GHz and two hardware threads per core. The device uses NVIDIA GeForce GTX Titan X GPU, which includes 3,840

CUDA cores and 12GB DDR5X DRAM. The operating system is Xubuntu 16.04 LTS. CUDA [11] and OpenMP [12] are adopted to implement the proposed hierarchical parallel approach. The compiler includes GCC-5.4.0 and CUDA-8.0s NVCC. First, we compare the proposed hierarchical PFAC with the traditional AC algorithm used in Snort. We use the packet generator to generate packets of different throughput as input packets. The input size is 208MB, with a minimum of 2.4 Gbps and a maximum of 20 Gbps, respectively, as shown in Table 1. We extract the Blacklist rules from Snort 2.9.x as the pattern rule file. The blacklist rules consist of 1,000 rules, 29,949 characters, a total of 30K bytes. The PFAC with 4 GPUs achieves an average throughput of 70 Gbps, which is more than 40 times faster than the traditional AC that achieves average throughput of only 1.63 Gbps. Table 1 shows that the PFAC achieves significant performance improvements over the traditional AC that does not meet packet throughput requirements. Table 2 shows the comparison with the related approaches. Our proposed method shows better performance than the other methods. In terms of memory requirements, for the blacklist rules, Snort needs a link list of 21.8 MB, whereas we only need a perfect hash table of 0.1725 MB. The proposed approach achieves up to 99.2% of memory reduction on Snort.

TABLE 1: A comparison of AC and PFAC in different packets throughput with four GPUs

Version	Packets number	Packets throughput	AC throughput	PFAC throughput
V1	220978	2.40 Gbps	1.50 Gbps	71.61 Gbps
V2	285044	3.10 Gbps	1.55 Gbps	68.93 Gbps
V3	217913	4.30 Gbps	1.51 Gbps	69.59 Gbps
V4	240717	5.80 Gbps	1.70 Gbps	68.77 Gbps
V5	222444	6.35 Gbps	1.52 Gbps	69.36 Gbps
V6	222431	6.40 Gbps	1.69 Gbps	70.55 Gbps
V7	218164	8.40 Gbps	1.69 Gbps	70.20 Gbps
V8	217510	9.80 Gbps	1.71 Gbps	71.50 Gbps
V9	217135	16.0 Gbps	1.76 Gbps	70.14 Gbps
V10	214913	20.0 Gbps	1.71 Gbps	72.25 Gbps

TABLE 2: Comparison with other methods

Paper	Platform	Algorithm	System throughput
Para-Snort [5]	CPU	AC	0.843 Gbps
Gnort [6]	GPU	AC	2.300 Gbps
Z.K. Baker [10]	FPGA	KMP	6.400 Gbps
A. Mitra [4]	FPGA	AC	12.90 Gbps
Our method	GPU	PFAC	70.29 Gbps

五、 結論與建議

我們提出了一種分層平行結構，它由三個平行級別組成，以加速開源NIDS Snort。分層平行結構在性能和記憶體需求減少方面取得了顯著進步。

六、 參考文獻

[1] Alfred V. Aho , Margaret J. Corasick, “Efficient string matching: an aid to bibliographic search,” Communications of the ACM, v.18 n.6, p.333-340, June 1975

[2] M.Roesch. Snort-lightweight intrusion detection for networks. In the 13th USENIX Conference on System Administration, 1999.

[3] OSEC - http://osec.neohapsis.com

[4] Abhishek Mitra , Walid Najjar , Laxmi Bhuyan, “Compiling PCRE to FPGA for accelerating SNORT IDS,” Proceedings of the 3rd ACM/IEEE Symposium on Architecture for networking and communications systems, December 03-04, 2007, Orlando, Florida, USA

[5] X. Chen, Y. Wu, L. Xu, Y. Xue, and J. Li., “Para-Snort: A Multithread Snort on Multi-core IA Platform.,” In Proceedings of Parallel and Distributed Computing and Systems (PDCS), 2009.

[6] Vasiliadis G, Antonatos S, Polychronakis M, Markatos EP, Ioannidis S,“Gnort: high performance network intrusion detection using graphics processors,” In: 11th international symposium on recent advances in intrusion detection, Boston, MA, USA, 2008, pp. 116-134.

[7] C.-H. Lin, J.C. Li, C.H. Liu, S.C. Chang, “Perfect hashing based parallel algorithms for multiple string matching on graphic processing units.,” IEEE Trans. Parallel Distrib. Syst. 99, 1 (2017)

[8] C.-H. Lin, C.-H. Liu, L.S. Chien, S.C. Chang, “Accelerating pattern matching using a novel parallel algorithm on gpus.,” IEEE Trans. Comput. 62(10), 19061916(2013)

[9] C.-H. Lin, C.-H. Liu, S.-C. Chang, and W.-K. Hon, “Memory- efficient pattern matching architectures using perfect hashing on graphic processing units,” in Proc. 31st Annu. IEEE Int. Conf. Com- put. Commun., 2012, pp. 19781986.

[10] Z. K. Baker and V. K. Prasanna, “A methodology for synthesis of efficient intrusion detection systems on FPGAs.,” Proc. FCCM, 2004.

[11] NVIDIA: CUDA Zone (2016). https://developer.nvidia.com/cudazone

[12] The OpenMP API specification for parallel programming (2016). http://www.openmp.org/

科技部補助專題研究計畫項下出席國際學術會議心得報告

日期： 107 年 8 月 16 日

計畫編號	106-2221-E-003 -012		
計畫名稱	使用高度平行演算法加速開源的網路入侵偵測系統 SNORT		
出國人員姓名	林政宏	服務機構及職稱	國立台灣師範大學電機系
會議時間	Aug 21, 2017 - Aug 23, 2017	會議地點	Helsinki, Finland
會議名稱	ICA3PP 2017 : 17th International Conference on Algorithms and Architectures for Parallel Processing		
發表論文題目	A Novel Parallel Dual-Character String Matching Algorithm on Graphical Processing Units		

一、參加會議經過

這是第一次參與 ICA3PP 國際研討會，ICA3PP 從 1995 年開始舉辦，今年是第 17 屆，主要聚焦於平行處理的算法和架構。今年在芬蘭赫爾辛基舉辦，為期三天，Aug 21, 2017 - Aug 23, 2017。

ICA3PP 現在被公認為全球的重要會議，今年的接受率為 29.03%，研究範疇涵蓋了平行算法和架構等許多方面，包括以下十七個 track:

Track 1: Parallel and Distributed Architectures
Track 2: Software Systems and Programming Models
Track 3: Distributed and Network-based Computing
Track 4: Big Data and its Applications
Track 5: Parallel and Distributed Algorithms
Track 6: Applications of Parallel and Distributed Computing
Track 7: Service Dependability and Security in Distributed and Parallel Systems
Track 8: Internet of Things and Cyber-Physical-Social Computing
Track 9: Performance Modeling and Evaluation

二、與會心得

參與本次會議主要是口頭發表我們的部分研究成果，題目為「A Novel Parallel Dual-Character String Matching Algorithm on Graphical Processing Units」，本論文成果主要運用於網路入侵偵測系統，提升病毒比對核心效能，發表後受到許多研究者的重視與正面回饋，並與我們進行交流。此外，會議進行的這三天，我們也聆聽多場論文發表，獲得許多研究方向與方法上的啟發。

三、發表論文摘要

Aho-Corasick algorithm has been widely used in network intrusion detection system to inspect network packets against thousands of attack patterns. To improve the performance of network intrusion detection systems, many variations of Aho-Corasick algorithm are proposed to accelerate multiple string matching on GPUs or dedicated hardware. One of the proposed variations is to increase the number of characters that are processed per cycle. However, increasing the number of characters processed per cycle will encounter two major problems. The first problem is the input alignment problem while the second problem is the large increase of memory required for storing the state transition table. The two problems cause the multi-character approach become less feasible. In this paper, we propose a novel parallel dual-character string matching algorithm on graphical processing units. In order to solve the two major problems, the proposed algorithm presents a new state machine to solve the input alignment problem, and compresses the state transition table using perfect hashing to solve the memory explosion problem. The experimental results show that the proposed algorithm is superior to the state-of-the-art approaches in terms of performance and memory requirements.

四、建議

至歐美參加高等級會議，所費不貲，感謝科技部的補助，得以減輕出國發表論文的壓力。

五、攜回資料名稱及內容

會議論文集電子檔案

圖一：學生口頭發表論文

圖二：大會晚宴

106年度專題研究計畫成果彙整表

計畫主持人：林政宏	**計畫編號：**106-2221-E-003-012-
計畫名稱：使用高度平行演算法加速開源的網路入侵偵測系統SNORT	

成果項目					量化	單位	質化 （說明：各成果項目請附佐證資料或細項說明，如期刊名稱、年份、卷期、起訖頁數、證號...等）
國內	學術性論文	期刊論文			0	篇	
		研討會論文			0		
		專書			0	本	
		專書論文			0	章	
		技術報告			0	篇	
		其他			0	篇	
	智慧財產權及成果	專利權	發明專利	申請中	0	件	
				已獲得	0		
			新型/設計專利		0		
		商標權			0		
		營業秘密			0		
		積體電路電路布局權			0		
		著作權			0		
		品種權			0		
		其他			0		
	技術移轉	件數			0	件	
		收入			0	千元	
國外	學術性論文	期刊論文			0	篇	
		研討會論文			2		初步成果分別發表於2017 ICA3PP與2018 ICASI 國際研討會，其中一篇獲得2018 ICASI 最佳論文獎。 Chung-Yu Liao and Cheng-Hung Lin, "A Novel Parallel Dual-Character String Matching Algorithm on Graphical Processing Units", in Proc. 17th International Conference on Algorithms and Architectures for Parallel Processing (ICA3PP-2017), Helsinki, Finland, August 21-23, 2017. (Acceptance rate: 29.03%) Cheng-Hung Lin and Cheng-Hung Hsieh, "A Novel Hierarchical Parallelism for Accelerating NIDS Using GPUs, " in Proc. IEEE International Conference on Applied System Innovation (IEEE ICASI 2018), Chiba, Tokyo, Japan, April

						13-17, 2018. (Best Conference Paper Award)
		專書			0	本
		專書論文			0	章
		技術報告			0	篇
		其他			0	篇
	智慧財產權及成果	專利權	發明專利	申請中	0	件
				已獲得	0	
			新型/設計專利		0	
		商標權			0	
		營業秘密			0	
		積體電路電路布局權			0	
		著作權			0	
		品種權			0	
		其他			0	
	技術移轉	件數			0	件
		收入			0	千元
參與計畫人力	本國籍	大專生			0	人次
		碩士生			5	本計畫共5名研究生參與，接受訓練包括C/C++, Python程式設計、字串比對演算法、Linux系統程式開發、GPU CUDA程式設計
		博士生			0	
		博士後研究員			0	
		專任助理			0	
	非本國籍	大專生			0	
		碩士生			0	
		博士生			0	
		博士後研究員			0	
		專任助理			0	
其他成果 （無法以量化表達之成果如辦理學術活動、獲得獎項、重要國際合作、研究成果國際影響力及其他協助產業技術發展之具體效益事項等，請以文字敘述填列。）						

科技部補助專題研究計畫成果自評表

請就研究內容與原計畫相符程度、達成預期目標情況、研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性）、是否適合在學術期刊發表或申請專利、主要發現（簡要敘述成果是否具有政策應用參考價值及具影響公共利益之重大發現）或其他有關價值等，作一綜合評估。

1. 請就研究內容與原計畫相符程度、達成預期目標情況作一綜合評估
   ■達成目標
   □未達成目標（請說明，以100字為限）
   　□實驗失敗
   　□因故實驗中斷
   　□其他原因
   說明：

2. 研究成果在學術期刊發表或申請專利等情形（請於其他欄註明專利及技轉之證號、合約、申請及洽談等詳細資訊）
   論文：■已發表　□未發表之文稿　□撰寫中　□無
   專利：□已獲得　□申請中　■無
   技轉：□已技轉　□洽談中　■無
   其他：（以200字為限）
   本研究之成果將以開源軟體方式，開放於Github，供研究社群使用。

3. 請依學術成就、技術創新、社會影響等方面，評估研究成果之學術或應用價值（簡要敘述成果所代表之意義、價值、影響或進一步發展之可能性，以500字為限）
   本研究提出了一種分層平行結構，它由三個平行級別組成，實現於圖形處理器，以加速網路入侵偵測系統Snort。本計畫實現提出的分層平行架構於四張圖形處理器 Nvidia Titan X，總系統吞吐量可高達 70 Gbps，與傳統使用於 Snort 中的 Aho-Corasick 演算法相比，可獲得高達四十倍的改善。此外本架構具有良好擴充性，當圖形處理器的數量增加，總系統的吞吐量也會隨之增加。此外，本計畫採用完美雜湊(Perfect Hashing)的方法，壓縮傳統 Aho-Corasick 的狀態機，減少在Snort 中99.2%多重樣式字串比對的記憶體使用量。本計畫成果可以用於資訊安全相關產業，提升網路攻擊掃描的效能。本計畫相關之研究成果，皆以開源方式，釋放於Github供研究社群參考。

4. 主要發現
   本研究具有政策應用參考價值：■否　□是，建議提供機關
   （勾選「是」者，請列舉建議可提供施政參考之業務主管機關）
   本研究具影響公共利益之重大發現：□否　□是
   說明：（以150字為限）