GDPI: Signature based Deep Packet Inspection using GPUs

Nausheen Shoaib, Jawwad Shamsi, Tahir Mustafa, Akhter Zaman, Jazib ul Hasan, and Mishal Gohar
System Research Laboratory
Deaprtment of Computer Science
FAST-National University of Computer and Emerging Sciences
Karachi, Pakistan

***Abstract*—Deep Packet Inspection (DPI) is necessitated for many networked application systems in order to prevent from cyber threats. The signature based Network Intrusion and Detection System (NIDS) works on packet inspection and pattern matching mechanisms for the detection of malicious content in network traffic. The rapid growth of high speed networks in data centers demand an efficient high speed packet processing mechanism which is also capable of malicious packets detection. In this paper, we proposed a framework GDPI for efficient packet processing which inspects all incoming packet's payload with known signature patterns, commonly available is Snort. The framework is developed using enhanced GPU programming techniques, such as asynchronous packet processing using streams, minimizing CPU to GPU latency using pinned memory and zero copy, and memory coalescing with shared memory which reduces read operation from global memory of the GPU. The overall performance of GDPI is tested on heterogeneous NVIDIA GPUs, like Tegra Tk1, GTX 780, and Tesla K40 and observed that the highest throughput is achieved with Tesla K40. The design code of GDPI is made available for research community.**

***Keywords*—Packet processing; Graphic Processing Units (GPUs); deep packet inspection; network security; parallel computing; heterogeneity; CUDA**

I. Introduction

Deep Packet Inspection (DPI) is a challenging task which involves network packet filtering mechanism. The incoming packets specially the payload part is inspected for malicious content. The payload of packet is processed in order to determine the authenticity of the packet at application layer. This process of payload inspection needs to be effective and efficient in terms of speed. The packet payload inspection is performed repeatedly for every incoming packet. Considering the wide scale usage of DPI application, a significant high speed packet processing mechanism is needed.

Existing techniques of DPI systems use Graphic Processing Units (GPUs) [1]–[3] for performance improvement in terms of high throughput. Despite of its speedup and performance efficiency, deployment of DPI method over GPUs using CUDA C-programming platform is quite thought-provoking for developers. GPUs are also used in various computation-intensive applications like IP lookup, general packet classification [9] and pattern matching for DPI systems. Pattern matching consumes 70% [5] of the execution time which can be reduced by exploiting parallelism in GPUs.Various algorithms have been proposed for pattern matching [4]–[6] such as Rabin Karp [7], Knuth-Morris-Pratt (KMP) [8], [9] developed with CUDA C programming platform.

Most of the DPI systems such as Gnort [10], and other system are not open source [11] that can be used. There is an extensive need for efficient DPI mechanism which fully exploits GPU functionality using modern GPU programming techniques. The DPI system must consists of modular programming approach so newer pattern matching algorithms can be integrated to check the efficiency and should be available for further investigation.

In this paper, the research is motivated with above mentioned challenges and requirements. To this end,a framework is proposed named "GDPI" for deep packet inspection.The general architecture of GDPI is briefly shown in Fig. 1. The caching [12], [13] of incoming network packets boost the network performance as stream of packet is transferred from CPU memory to GPU memory. The asynchronous call of CUDA functions increases the transfer speed as CPU is not locked while streams are transferred to GPU memory for further packet processing.The framework used two common methods for packet transfer that is pinned memory and zero copy mechanisms. The packets are processed to identify malicious contents in the payload of packet using known patterns or signatures. We used an open source SNORT [14] database for known malicious patterns or signatures. The incoming packet's payload is inspected using open source CUDA based pattern matching algorithms, KMP and Rabin Karp. The framework is designed considering modular programming approach in a way that either of pattern matching algorithms,KMP or Rabin Karp can be selected for the patterns to be matched.The modular approach facilitates the integration of different pattern matching algorithms as per need.The memory coalescing technique is used for patterns residing in GPU shared memory to be matched with incoming packets, which also reduces the read operations and increases the overall packet processing speed. If packet payload contains malicious content, it is dropped or discarded otherwise forwarded to the next hop.Table 1 briefly elaborates the research challenges, goals and research contributions of this paper. The main contribution of this paper are:

- Developed an efficient and effective open source GPU based DPI solution for research community.
- Used a modular programming approach considering the selection of either of pattern matching algorithm

TABLE I. CHALLENGES, GOALS, RESEARCH CONTRIBUTIONS

S.No	Challenges	Goals	Research Contributions
1	High throughput	To develop network application based on hardware accelerators using GPUs which is capable of processing network traffic in an efficient way	The use of graphics processing units ensures the high level of parallelism
2	Prevention from malicious content	To inspect all incoming network traffic in order to prevent from cyber threats	Deep Packet Inspection to examine incoming packet payload
3	Use an efficient pattern matching algorithms for Deep Packet Inspection	To analyze the efficient pattern matching algorithm that can be integrated with the framework in order to observe the overall performance	Modular approach used to develop the GDPI
4	Parallel processing of packets using GPUs	To use enhance GPU programming techniques in a effective way	GPU programming techniques such as stream processing, zero copy,memory coalescing is incorporated
5	Availability of program code for GPU based DPI	Research community can access it for further investigations	Open Source

such as KMP or Rabin Karp in this case.

- Enhanced GPU programming techniques like asynchronous stream processing, zero copy, memory coalescing used to fully exploit parallelism.
- Performed extensive experiments using heterogeneous NVIDIA GPUs like Jetson Tk1, GTX 780, and Tesla K40.

The rest of this paper is organized as follows: Section II consists of related work which discusses the recent trends in high performance intrusion detection systems, regular expression matching methods for deep packet inspection, packet processing techniques on GPUs, pattern matching algorithms for DPI and machine learning methods used for DPI. Section III contains the proposed Methodology which includes packet capturing, packet transfer from CPU memory to GPU memory and pattern matching. Finally, Section IV includes the experiments performed using different NVIDIA GPUs. Section V follows with conclusion and future work.

II. RELATED WORK

We divide the related work in four broad categories.Each category briefly elaborates the approaches used in DPI systems. The categories are: high performance intrusion detection, regular expression matching for DPI, packet processing using GPUs, pattern matching algorithms over GPUs, and machine learning methods for DPI. Table 2 briefly explains the feature comparison of proposed framework with existing solutions.

A. High Performance Intrusion Detection

Network Intrusion Detection systems (NIDS) are well known source for monitoring inbound and outbound network traffics in order to prevent from DDOS and other types of attacks. Numerous solutions are available for the detection of malicious content in network monitoring systems like Snort [15], Suricata, BRO [16] are widely used NIDS.Due to continuous growth in network applications, high throughput is needed for the processing of network applications. The emergence of graphics processing units has increasingly in demand for every aspect in network processing [17] applications. A GPU based intrusion detection system, Gnort, Kargus [18] has been developed to accelerate packet processing speeds in NIDS.

The research is inspired with these existing solutions and designed a framework for signature based DPI using GPUs. Gnort and other GPU based DPI solutions are not open source,so we are not aware of its performance comparison with our framework which is open source and available for the research community. We developed it using modular approach which facilitates the integration of any pattern matching algorithm to improve its performance. Our framework is tested on different types of kepler architecture of NVIDIA GPUs like Tegra Tk1, GTX780, Tesla K40.

B. Regular Expression Matching for Deep Packet Inspection

In order to improve the efficiency of signature based NIDS, numerous automaton approaches has been adopted for its effectiveness. Deterministic finite automata (DFA) and nondeterministic finite automata (NFA) are two popular methods used in NIDS. In both approaches, regular expression string matching is done but with different performance and memory usage properties.Regular expression matching on deterministic finite automata (DFA) [19] is used for fast packet processing at wire speed.The researchers proposed DFA estimator and a regular expression based grouping algorithm. The DFA estimator calculates size based on regular expression and grouping algorithm without building actual DFA and showed improvement in terms of memory size and speed. For further enhancing speeds in string matching while controlling memory expense, a finite state machine (FSM) based scheme is developed which scans multiple characters in parallel by running small sized FSMs. This approach reduced the memory consumption cost and thus increased its efficiency. Tunable finite automata (TFA) [20] has solved state explosion and prediction performance problem. TFA allows multiple concurrent active states unlike DFA which allow only one active state, and achieved 98% reduction in memory consumption. The head body finite automaton [11] used multicore general purpose processors (GPP) for parallelism and single instruction multiple data (SIMD) operations. The head body matching is based on pre-defined DFA depth value and head size for partitioning and parallel processing and observed 58% significant increase in throughput.

The research is motivated with the generalized DPI methods [21] and used string matching based on hashing approach using general purpose graphics processing units (GPGPUs).The hashing method calculates the hash value of each string of currently inspected incoming payload. The hash

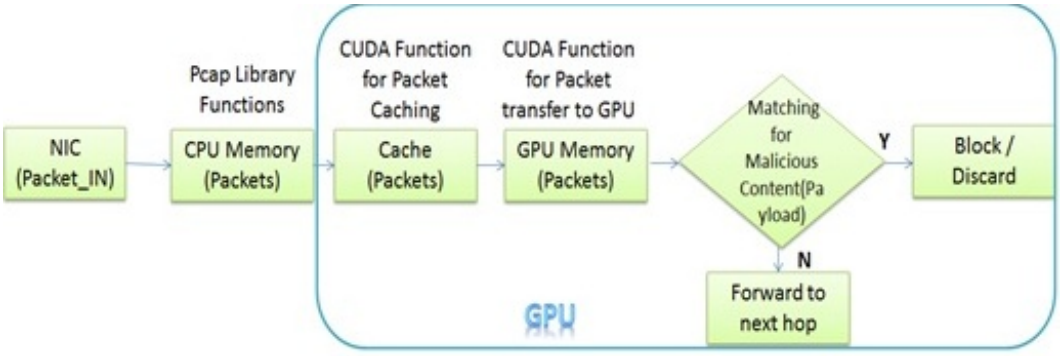

Fig. 1. General architecture of GDPI.

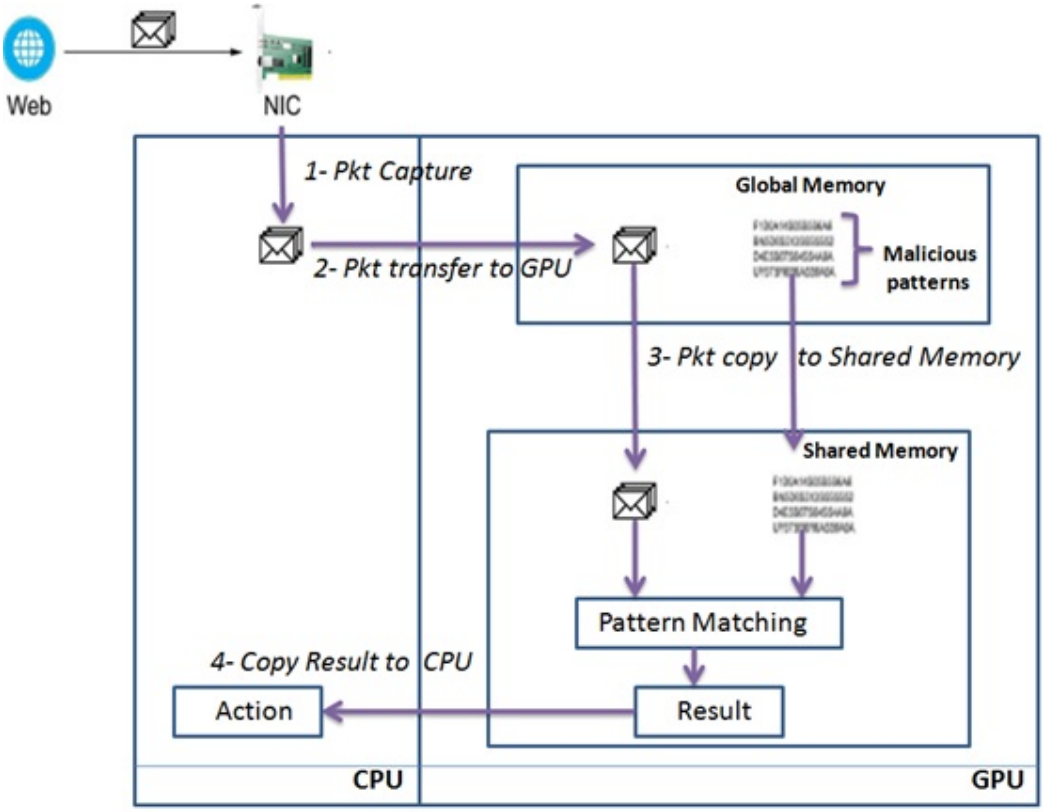

Fig. 2. Modular process flow of proposed framework GDPI.

value of payload is compared (binary search can be done for matching) byte by byte with the precomputed hash value of corresponding signature patterns. The other approach includes string search algorithm which looks for the occurrences of strings and compared with signature patterns.

C. Packet Processing using GPUs

GPU computing [22] has become an integral part of computing systems.There has been remarkable increase in throughput and performance capabilities of network applications as well. GPUs brought the new prospects for packet processing by offloading [23], [24] computation needs to the GPUs as it offers extreme thread level parallelism on hundreds of core. Packets are processed in a batch [25] to reduce per packet memory management overhead. Effective packet processing for network application has complex operations like TTL decrement, checksum recalculation, broadcast management, handling of IP options like ICMP or ARP. The network operations needed modular pipeline processing [26] for synchronous and asynchronous packets with in-order execution by leveraging GPUs. The growth of virtualization for network appliances lead the use of software based virtual switches. These virtual switches needed high packet iO rate and a classification scheme for high throughput as the size and dimension of forwarding table is continuously growing in case of SDN. The packet classifiers attained high throughput by ensuring the coalesced memory [27] access in order to reduce latency for off chip memory.

The research is highly inspired with enhanced GPU based CUDA programming techniques which can be used for fast and effective packet processing. We used CUDA function APIs for batch processing of packets. This helped us to increase the packet transfer rate from CPU memory to GPU memory. The batch of packets are sent to GPU memory instead of sending single packet per thread improves the overall performance. The packet offloading from CPU to GPU memory using page locked or pinned memory via asynchronous CUDA functions increased the transfer rate. Another CUDA technique for CPU to GPU transfer is Zero Copy [28], [29] in which space is allocated to both CPU and GPU and does have to transfer the data, reduces CPU cycles and save memory bandwidth.The memory accesses for pattern matching is coalesced which decreased memory access latency and speedup the packet processing capabilities of the framework.

D. Pattern Matching Algorithms over GPUs

String matching is an important technique for various applications.Traditional string matching algorithms requires backtracking and the comparison process [30] repeatedly effects the efficacy of the algorithm.There are various algorithms for Pattern matching such Aho-Corasick [31], Boyer-Moore [32]. Middlebox services for network applications typically inspects for known patterns which can be present anywhere in payload. The Aho-Corasick algorithm constructs a DFA and provides an optimal performance as it frequently refers memory and causes large number of cache misses. DFC algorithm [17] resolved this problem by minimizing CPU stalls and maximizing CPU level parallelism. The computation time for processing large number of patterns has been sufficiently decreased by using high end parallel SIMD architecture of GPUs. KMP and Rabin Karp, etc. showed the speedup for pattern matching when integrated with these high end GPUs.

Another GPU based algorithm HMPA [33] outperformed CPU only and GPU only implementations as it adopted the hybrid approach. The packet filtering and full pattern matching is performed by CPU and GPU respectively, but overall performance is limited. For this purpose, a variant of HMPA is designed known as CHMPA [3] which estimates CPU and GPU processing capabilities and self allocates the process at runtime.The varying payload length also become a bottleneck which is resolved by LHMPA [11].It restricts those packets whose payload length exceeds predefined length bound. The variable length payload for pattern matching is resolved by using a probabilistic data structure bloom filter [1] using multiple hash functions.

The research is encouraged with the recent and efficient algorithms for pattern matching. We used CUDA based KMP and Rabin Karp algorithm. KMP is a single pattern matching algorithm but we developed it as a variant of KMP for multi-pattern matching.The hashing method for DPI is fulfilled by Rabin Karp algorithm. The GDPI framework used a modular approach that either of algorithm can be selected to identify the overall performance and efficacy of the framework.

E. Machine Learning Methods for DPI

Network traffic classification has become significantly important with rapid growth of current internet network and online applications. Advance Machine Learning techniques provide a new dimension for detection of attacks at various levels. The machine learning techniques such as naive bayes, support vector machine (SVM), C4.5 decision trees have been used for the classification of attacks and observed high accuracy factor with C4.5 [34] classifier. The flow based anomaly detection systems adopted a deep learning approach know as deep neural network [35] and observed 75.5% accuracy and has high potential in software defined network (SDN) environment. The network traffic characterization and application identification can also be achieved through convolutional neural network (CNN) [36] with both encrypted and unencrypted network traffic.

We are highly motivated with this new paradigm for classification of attacks from cyber threats prevention. In this paper, we have not used any machine learning or deep learning techniques but we can incorporate these techniques in our future work.

III. Methodology

This section explains development design of GDPI using CUDA C programming platform. CUDA is a parallel computing application platform developed by Nvidia to use CUDA enabled graphics processing unit.

A. Packet Capturing at NIC

The packets are generated by an open source network traffic generator tool Ostinato at Network Interface Card (NIC). Fig. 2 shows that the stream of packets are created using cudaStream() API. The API creates stream of packets at CPU memory. The stream of packets is transferred to GPU memory for further processing and reduces the overhead of sending single packet per thread. It increased the packet transfer rate and also decreased the CPU to GPU latency due to asynchronous CUDA function operations.

B. Packet Transfer from CPU to GPU Memory

The CUDA based application framework manages concurrency by executing asynchronous functions for stream processing.The stream of incoming packets is transferred from CPU memory to GPU memory in two ways: the first method is to transfer by using page locked host memory also called pinned memory. The CUDA API is cudaMemcpyAsync() which is a asynchronous transfer and helps improved the transfer rate and reduce the latency from CPU to GPU memory. The second method is to transfer the stream of packets using Zero copy mechanism. The Zero Copy is a way to map host memory and access it directly over PCIe without doing an explicit memory transfer as it allows CUDA kernel to directly access host memory, instead of reading data from GPU global memory. The CUDA API for zero copy is cudaHostGetDevicePointer() creates a memory to access the data between CPU and GPU. We used both methods to transfer the incoming packets and open source signatures to the GPU memory.

C. Pattern Matching

The open source SNORT signature's database is used for the detection of malicious content in packet payload.The

TABLE II. FEATURE COMPARISON OF EXISTING SOLUTIONS WITH GDPI

S.No	Feature Set	Snort	Bro	Suricata	Kargus	Gnort	GDPI
1	GPU based system	-	-	✓	✓	✓	✓
2	Open source	✓	✓	-	-	-	✓
3	Modular approach	-	-	N/A	-	N/A	✓
4	Enhance GPU programming techniques	N/A	-	N/A	-	N/A	✓
5	Evaluation on heterogeneous GPUs	N/A	-	N/A	-	-	✓

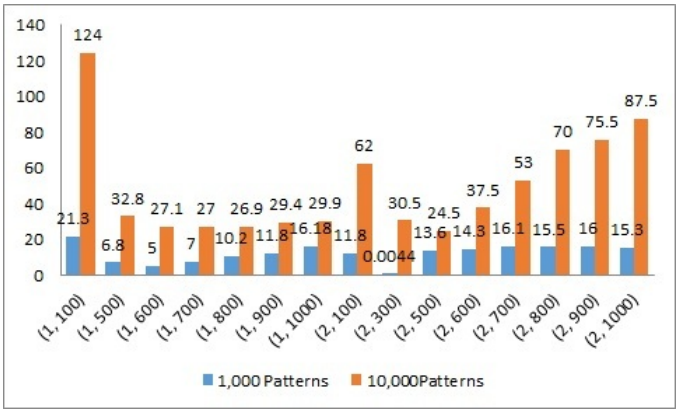

Fig. 3. Packet processing time with varying no. of threads and blocks on group of pattern sets.

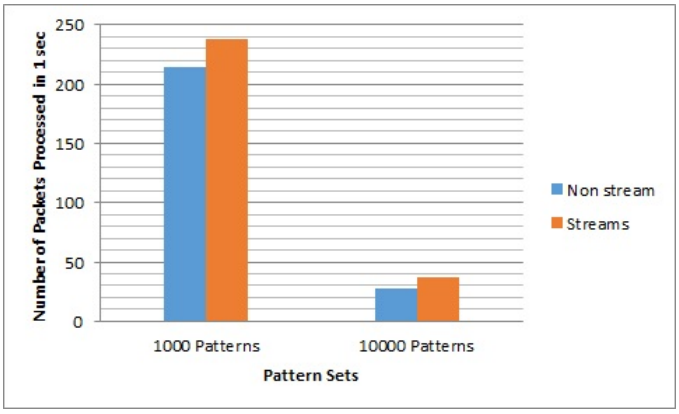

Fig. 4. Number of packets processed for pattern matching on group of pattern sets.

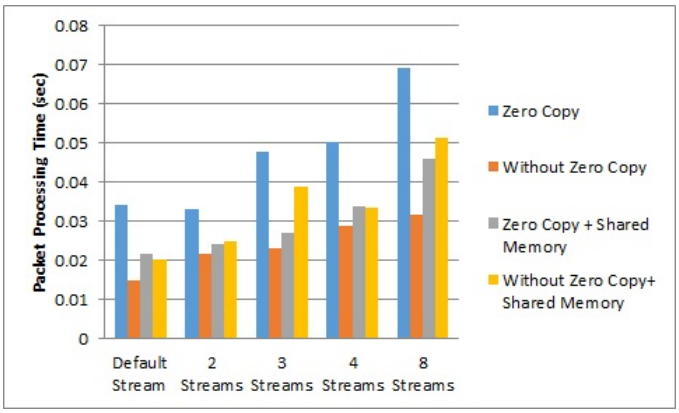

Fig. 5. Packet processing time with and without zero copy and shared memory.

hexadecimal format of malicious signature is loaded into CPU memory.The string matching algorithms used for pattern matching are KMP and Rabin Karp. The preprocessing includes the computation of prefix table and hash values of signature over GPU, while using KMP or Rabin Karp, respectively. The modular approach is used and either of the algorithms can be selected for pattern matching.By copying preprocessed signature patterns to GPU's shared memory, reduces read operations while searching using memory coalescing mechanisms. For patterns p1,.....pn, and GPU grid consisting of threads t1.....tk, each thread i iterates j times, assigned pattern pi+(j*n) which then applies the searching algorithm on it.

The packet payload is compared by using string matching algorithms developed for GPUs such Rabin Karp and KMP. After the packet is transferred to the GPU memory, a kernel is launched with 4 Blocks of 512 threads by default (unless configured otherwise). Each thread in a grid scans for a specific malicious signature using the selected algorithm KMP or Rabin Karp and then returns the action to be performed on the packet. The performance is increased due to coalesced memory access technique together with shared memory. The results are copied back to host. The respective stream managing thread then analyze the result. In case of malicious content found in payload of the packet, the connection is closed by sending a reset control signal TCP_RST and TCP_FIN to both end points to finally close the connection.

IV. EXPERIMENTS

The framework is initially tested on NVIDIA Tegra TK1, a mobile embedded system with 192 cores and 2GB of memory. We also evaluated GDPI on commodity hardware with Ubuntu server 14.0.4 with linux kernel installed on it.The supported programming platform CUDA 7.5 is installed for GPU programming. These machines are equipped with NVIDIA GTX 780 with 2304 cores and NVIDIA Tesla K40c with 2880 cores. The memory installed in these commodity hardware is 8GB.

A. Packet Streams

Initially, the incoming packets is sent to GPU using single packet per thread by varying grid configuration (blocks, threads). Fig. 3 shows the packet processing time with different number of blocks and threads configuration of a grid. We observed the time taken for packet processing on signature pattern set of 1000 and 10,000. Initially, the grid is launched with single block and 100 threads and then gradually the grid configuration is varied. The packet processing speed is gradually increased and showed improved performance as shown in Fig. 3.

After that we implemented stream processing through asynchronous CUDA function API cudaStream(). The API created the stream of packets and then transfer to GPU memory.

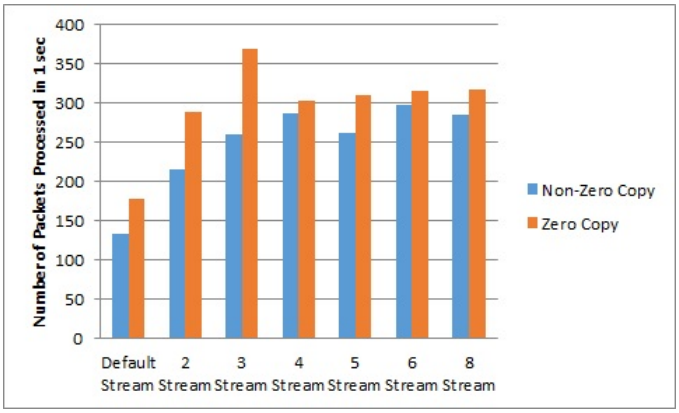

Fig. 6. No. of packets processed with coalescing memory.

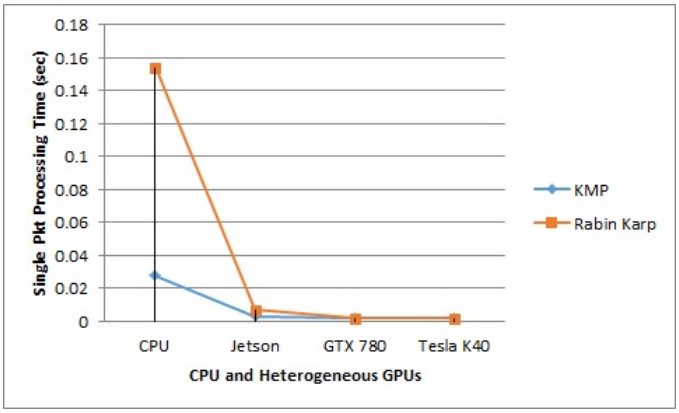

Fig. 7. GDPI performance on CPU and Heterogeneous GPUs.

The stream processing of packets showed the performance improvement when compared with single packet per thread. Fig. 4 shows the number of packets processed per second on pattern set of 1,000 and 10,000, respectively. The processing time of packet streams reduced the overhead of sending single packet per thread.The packet streams has clearly showed significant increase in packet processing time.

B. Packet Transfer using Pinned Memory and Zero Copy

The incoming packets stream transferred from CPU to GPU memory using pinned memory and zero copy mechanism. The packet stream transfer rate is improved using zero copy.The incoming packets and pattern sets are in shared memory of GPU. Fig. 5 shows that multiple streams are sent to the GPU shared memory for comparison with pattern sets.The packets processing time using zero copy with shared memory showed an increase in performance as compared to non-zero copy.

C. Pattern Matching using Memory Coalescing

The signature patterns were copied from device global memory to shared memory for patterns to be compared with incoming packet streams. When malicious signature patterns copied to shared memory. Fig. 6 shows that the memory coalescing technique together with zero copy and patterns reside in shared memory increased the overall performance of packet processing.

D. DPI Module Performance on Hetrogeneous GPUs

Fig. 7 shows the single packet processing time is evaluated on CPU cores and different NVIDIA GPUs kepler architecture like, Jetson Tk1, GTX780 and Tesla K40 with pattern matching algorithms, KMP and Rabin Karp. GDPI framework showed improved results on Tesla GPU with Rabin Karp algorithm.

V. Conclusion and Future Work

This paper proposed a framework GDPI for signature based deep packet inspection using GPUs. The framework GDPI scans incoming packet for the detection of malicious content in the payload of packet using Snort signatures. The framework is designed using a modular programming approach so any of the pattern matching algorithm can be integrated to observe the performance improvements.The implementation is mainly focused on the recent GPU programming techniques such as stream processing, memory overlapping methods zero copy, etc. and observed speedup in packet transfer from from CPU to GPU memory. The asynchronous nature of CUDA operations violates the serialization process. The use of shared memory where patterns and packets are copied reduced read operations from global memory. The memory coalescing technique also resulted in reduction of bandwidth and thus speedup the matching process which is considered as main bottleneck in DPI systems. The framework is tested on heterogeneous kepler architectures of NVIDIA GPUs such as Tegra Tk1, GTX 780 and Tesla K40. The experiment results achieved maximum speed of packet processing when tested over Tesla K40 with Rabin Karp algorithm. The GDPI framework is open source for research community for further investigation.

As network security is of vital concern for cloud and bigdata applications.The research community is continuously working on recent methods and practices for deep packet inspection. Machine learning (ML) and deep learning (DL) techniques providing a new paradigm to Network Intrusion Detection Systems. ML and DL techniques are used for the classification of attacks in order to prevent from cyber threats. Network intrusion detection systems performance needs improvement in terms of accuracy, which can be achieved with the use of deep learning techniques for intrusion or fraud detection. These systems can be implemented and tested in real time SDN environment to evaluate the effectiveness of the system in terms of accuracy, latency, and throughput. The proposed GDPI framework is not using any machine leaning or deep learning classification techniques such as support vector machine and deep convolutional neural network (DNN). The incorporation of these techniques can significantly increase the effectiveness of the proposed framework in real time environment.

Acknowledgment

This work was supported by NVIDIA Teaching and Research Center. The work is also supported by Higher Education Commission (HEC) Pakistan under the grant number NRPU-5946.

REFERENCES

[1] Hung, Che-Lun, Chun-Yuan Lin, and Po-Chang Wu. "An Efficient GPU-Based Multiple Pattern Matching Algorithm for Packet Filtering." Journal of Signal Processing Systems 86, no. 2-3 (2017): 347-358

[2] Lee, Chun-Liang, and Tzu-Hao Yang. "A Flexible Pattern-Matching Algorithm for Network Intrusion Detection Systems Using Multi-Core Processors." Algorithms 10, no. 2 (2017)

[3] Lin, Yi-Shan, Chun-Liang Lee, and Yaw-Chung Chen. "A capability-based hybrid CPU/GPU pattern matching algorithm for deep packet inspection." International Journal of Computer and Communication Engineering 5, no. 5 (2016): 321.

[4] Diwate, Mr Rahul B., and Satish J. Alaspurkar. "Study of Different Algorithms for Pattern Matching." International Journal 3, no. 3 (2013).

[5] Lin, Cheng-Hung, Chen-Hsiung Liu, Lung-Sheng Chien, and Shih-Chieh Chang. "Accelerating pattern matching using a novel parallel algorithm on GPUs." IEEE Transactions on Computers 62, no. 10 (2013): 1906-1916.

[6] Bellekens, X., I. Andonovic, R. C. Atkinson, C. Renfrew, and T. Kirkham. "Investigation of GPU-based pattern matching." In The 14th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNet2013). 2013.

[7] Sharma, Jyotsna, and Maninder Singh. "CUDA based Rabin-Karp Pattern Matching for Deep Packet Inspection on a Multicore GPU." International Journal of Computer Network and Information Security 7, no. 10 (2015): 70.

[8] Lin, Kuan-Ju, Yi-Hsuan Huang, and Chun-Yuan Lin. "Efficient parallel knuth-morris-pratt algorithm for multi-GPUs with CUDA." In Advances in Intelligent Systems and Applications-Volume 2, pp. 543-552. Springer Berlin Heidelberg, 2013.

[9] Rasool, Akhtar, and Nilay Khare. "Parallelization of KMP string matching algorithm on different SIMD architectures: Multi-core and GPGPU's." International Journal of Computer Applications 49, no.

[10] Vasiliadis, Giorgos, Spiros Antonatos, Michalis Polychronakis, Evangelos Markatos, and Sotiris Ioannidis. "Gnort: High performance network intrusion detection using graphics processors." In Recent Advances in Intrusion Detection, pp. 116-134. Springer Berlin/Heidelberg, 2008.

[11] Lin, Yi-Shan, Chun-Liang Lee, and Yaw-Chung Chen. "Length-bounded hybrid CPU/GPU pattern matching algorithm for deep packet inspection." Algorithms 10, no. 1 (2017): 16.

[12] Cui, L., Yu, F.R. and Yan, Q., 2016. When big data meets software-defined networking: SDN for big data and big data for SDN. IEEE Network, 30(1), pp.58-65.

[13] Zhang, W., Wood, T., Ramakrishnan, K.K. and Hwang, J., 2014, June. SmartSwitch: Blurring the Line Between Network Infrastructure & Cloud Applications. In HotCloud.

[14] Jiang, Haiyang, Guangxing Zhang, Gaogang Xie, Kav Salamatian, and Laurent Mathy. "Scalable high-performance parallel design for network intrusion detection systems on many-core processors." In Proceedings of the ninth ACM/IEEE symposium on Architectures for networking and communications systems, pp. 137-146. IEEE Press, 2013.

[15] Roesch, Martin. "Snort: Lightweight intrusion detection for networks." In Lisa, vol. 99, no. 1, pp. 229-238. 1999.

[16] Bhosale, Dhanashri Ashok, and Vanita Manikrao Mane. "Comparative study and analysis of network intrusion detection tools." In Applied and Theoretical Computing and Communication Technology (iCATccT), 2015 International Conference on, pp. 312-315. IEEE, 2015.

[17] Choi, Byungkwon, Jongwook Chae, Muhammad Jamshed, Kyoungsoo Park, and Dongsu Han. "DFC: Accelerating String Pattern Matching for Network Applications." In NSDI, pp. 551-565. 2016

[18] Jamshed, Muhammad Asim, Jihyung Lee, Sangwoo Moon, Insu Yun, Deokjin Kim, Sungryoul Lee, Yung Yi, and KyoungSoo Park. "Kargus: a highly-scalable software-based intrusion detection system." In Proceedings of the 2012 ACM conference on Computer and communications security, pp. 317-328. ACM, 2012.

[19] Liu, Tingwen, Alex X. Liu, Jinqiao Shi, Yong Sun, and Li Guo. "Towards fast and optimal grouping of regular expressions via DFA size estimation." IEEE Journal on Selected Areas in Communications 32, no. 10 (2014): 1797-1809.

[20] Xu, Yang, Junchen Jiang, Rihua Wei, Yang Song, and H. Jonathan Chao. "TFA: A tunable finite automaton for pattern matching in network intrusion detection systems." IEEE Journal on Selected Areas in Communications 32, no. 10 (2014): 1810-1821.

[21] Xu, Chengcheng, Shuhui Chen, Jinshu Su, S. M. Yiu, and Lucas CK Hui. "A Survey on Regular Expression Matching for Deep Packet Inspection: Applications, Algorithms, and Hardware Platforms." IEEE Communications Surveys & Tutorials 18, no. 4 (2016): 2991-3029.

[22] Owens, John D., Mike Houston, David Luebke, Simon Green, John E. Stone, and James C. Phillips. "GPU computing." Proceedings of the IEEE 96, no. 5 (2008): 879-899.

[23] Renart, Eduard G., Eddy Z. Zhang, and Badri Nath. "Towards a GPU SDN controller." In Networked Systems (NetSys), 2015 International Conference and Workshops on, pp. 1-5. IEEE, 2015.

[24] Han, Sangjin, Keon Jang, KyoungSoo Park, and Sue Moon. "PacketShader: a GPU-accelerated software router." In ACM SIGCOMM Computer Communication Review, vol. 40, no. 4, pp. 195-206. ACM, 2010.

[25] Rogora, Daniele, Michele Papalini, Koorosh Khazaei, Alessandro Margara, Antonio Carzaniga, and Gianpaolo Cugola. "High-Throughput Subset Matching on Commodity GPU-Based Systems." system 20, no. 40M (2017): 212M.

[26] Sun, Weibin, and Robert Ricci. "Fast and flexible: parallel packet processing with GPUs and click." In Proceedings of the ninth ACM/IEEE symposium on Architectures for networking and communications systems, pp. 25-36. IEEE Press, 2013.

[27] Varvello, Matteo, Rafael Laufer, Feixiong Zhang, and T. V. Lakshman. "Multilayer packet classification with graphics processing units." IEEE/ACM Transactions on Networking 24, no. 5 (2016): 2728-2741.

[28] Vasiliadis, Giorgos, Lazaros Koromilas, Michalis Polychronakis, and Sotiris Ioannidis. "Design and Implementation of a Stateful Network Packet Processing Framework for GPUs." IEEE/ACM Transactions on Networking (TON) 25, no. 1 (2017): 610-623.

[29] Vasiliadis, Giorgos, Lazaros Koromilas, Michalis Polychronakis, and Sotiris Ioannidis. "GASPP: A GPU-Accelerated Stateful Packet Processing Framework." In USENIX Annual Technical Conference, pp. 321-332. 2014.

[30] Faro, Simone, and Thierry Lecroq. "The exact online string matching problem: A review of the most recent results." ACM Computing Surveys (CSUR) 45, no. 2 (2013): 13.

[31] Aho, Alfred V., and Margaret J. Corasick. "Efficient string matching: an aid to bibliographic search." Communications of the ACM 18, no. 6 (1975): 333-340.

[32] Boyer, Robert S., and J. Strother Moore. "A fast string searching algorithm." Communications of the ACM 20, no. 10 (1977): 762-772.

[33] Lee, Chun-Liang, Yi-Shan Lin, and Yaw-Chung Chen. "A hybrid CPU/GPU pattern-matching algorithm for deep packet inspection." PloS one 10, no. 10 (2015): e0139301.

[34] Shafiq, Muhammad, Xiangzhan Yu, Asif Ali Laghari, Lu Yao, Nabin Kumar Karn, and Foudil Abdessamia. "Network Traffic Classification techniques and comparative analysis using Machine Learning algorithms." In Computer and Communications (ICCC), 2016 2nd IEEE International Conference on, pp. 2451-2455. IEEE, 2016.

[35] Tang, Tuan A., Lotfi Mhamdi, Des McLernon, Syed Ali Raza Zaidi, and Mounir Ghogho. "Deep Learning Approach for Network Intrusion Detection in Software Defined Networking." In Wireless Networks and Mobile Communications (WINCOM), 2016 International Conference on, pp. 258-263. IEEE, 2016.

[36] Lotfollahi, Mohammad, Ramin Shirali, Mahdi Jafari Siavoshani, and Mohammdsadegh Saberian. "Deep Packet: A Novel Approach For Encrypted Traffic Classification Using Deep Learning." arXiv preprint arXiv:1709.02656 (2017).