Network Functions Evaluation of Hardware Accelerated NFV Platform in View of 5G Requirements

Athanasia-Nancy Alonistioti
Department of Informatics & Telecommunications
National Kapodistrian University of Athens
Athens, Greece
email: nancy@di.uoa.gr

Abstract— **Network Function Virtualization (NFV) enables the implementation of "softwarized" network functions, using a shared hardware substrate. However, such functions are expected to offer performances comparable to native hardware environments. Therefore, increasing the performance of "software-based" Network Functions (NFs) that are executed on commodity hardware servers is an emerging challenge that should be addressed to fulfill the vision of NFV. Hardware accelerators, such as Many Integrated Core Architectures (MICs), Graphic Processor Units (GPUs) and Field Programmable Gate Arrays (FPGAs) are a valuable asset that can be paired with commodity servers to boost up network services performance. In this paper, we present an analysis of two commonplace network functions: a) Routing, b) Firewall and evaluate their implementation in an NFV middlebox combining software based network functions along with hardware accelerated ones in view of their applicability for 5G and interworking with Broadband Forum (BBF) network segments. Finally, we present detailed experimental results of CPU, GPU and FPGA solutions for these functions (routing, firewall and DPI) in order to evaluate their performance and verify their suitability for incorporation in hardware-accelerated NFV platforms.**

Keywords- 5G; Network Function Virtualization; Network services.

I. INTRODUCTION

Virtualization is the ability to implement a functionality separated from the hardware and simulated as a "virtual instance". Modern networks increasingly rely on advanced network processing functions for a wide spectrum of crucial functions ranging from security (firewalls, Intrusion Detection Systems (IDS), etc.), traffic shaping (rate limiters, load balancers), dealing with address space exhaustion (Network Address Translation (NAT)) or improving the performance of network applications (traffic accelerators, caches, proxies), to name a few.

Middleboxes are defined as intermediary boxes performing these network functions and represent the defacto approach for network evolution in response to changing performance, security, and policy compliance requirements. NFV brings a new era in network virtualization by involving the implementation of software-based middleboxes that implement such NFs in software and can run on standard server hardware. Such NFV middleboxes are suitable nodes for implementing a virtualization scheme in the basis of "Network Function as a Service (VNFaaS)" as defined in the NFV standardization process [1]. ICT research community seeks to address this issue [2] by shifting software solutions running on top of modern multi-core systems that have good flexibility and programmability, but they inherently lack high parallelism and quickly become the performance bottleneck for more compute-intensive applications, to a decoupling from dedicated hardware solutions. For example, even the best multi-core software algorithms today for routing and firewall can only achieve a rate of up to 30 Gbps [6], which is an order of much less than the widely deployed 100 Gbps links used in large Internet Service Provider backbones and Data Center networks. Moreover, taking into consideration that the performance vision in the era of 5G wireless networks is to achieve a 10Gbps individual user experience [3], the Internet Service Provider backbones and Data Center networks will boost their bandwidth needs in the scale of Tbps. Therefore, increasing the performance of "software-based" network services that are executed on commodity hardware servers is an emerging challenge that should be addressed to fulfill the vision of "software-based" network services through NFV. Modern commodity servers can be paired with hardware accelerators (Many Integrated Core Architectures (MICs), Graphic Processor Units (GPUs) and Field Programmable Gate Arrays (FPGAs) to boost up their performance. Such accelerators are a valuable asset and can be exploited to offload on demand the "software-based" network services that cannot comply with high performance demanding NFV environments. In this direction, the work presented in this paper explores the pairing of modern hardware accelerator (MICs, GPUs, and FPGAs) with commodity hardware to build a scalable hardware accelerated NFV platform to speed up NFs and support NF reallocation to different accelerators on demand. Important NFs to be considered in this paper are a) Routing and b) Firewall. These are very important NFs in view also of the needs for Broadband Forum (BBF) and 5G 3GPP interworking for the seamless application provision and QoS requirements between the involved network segments [14].

The rest of this paper is organized as follows. Section II presents related work and in parallel examines the considered NFs, their characteristics, their challenges and the applicability of hardware acceleration on them. Section III presents experimental results to evaluate and compare the performance of software and hardware assisted implementations of these use cases. Finally, Section IV concludes this paper.

II. Network Functions: Related work Identification and NFV challenges

In this section, we present the two aforementioned NFs a) Routing, b) Firewall. An overview of algorithmic schemes, applicability of hardware acceleration, advantages, related work and NFV tradeoffs are discussed for each use case.

A. *Routing*

IP Router functionalities are grouped into two categories, packet processing and packet forwarding. Routing table lookup is the most critical packet processing algorithm and has been extensively discussed in the literature, as it is considered the main bottleneck in router performance [4]. Therefore, we focus on routing table lookup process, which is the main performance bottleneck and we omit discussing updating process. Key metrics used to evaluate routing table lookup performance are searching time, dynamic updates and memory requirements. In software routing table lookup schemes [4], tries are the dominant solution for the organization of IP Router Forward Information Base (FIB) and the service of IP lookup requests in virtualized environments [5]. Multibit tries are a subcategory of tries that achieve a speedup of the lookup process as multi-bit tries examine strides of bits at a time and lower total memory accesses, as shown in Fig. 1.

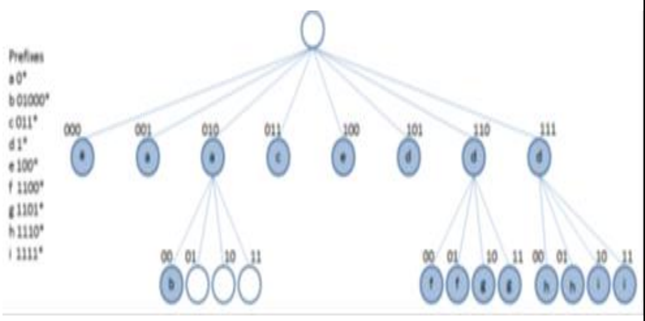


Figure 1. Multi-bit trie representation of a FIB.

Moreover, further speedup can be achieved by the processing of “IP request” in large batches if parallelism in lookup process can be implemented. Towards this direction, GPUs can instantiate thousands or threads in parallel that can be exploited to boost performance in trie lookup process. In addition, there are three characteristics in IP table lookup process that make GPU implementations a tempting alternative [6], [7].

•IP requests in large batches: In real routers, IP requests can form batches of more than 216 concurrent requests. However, the bigger the size of a batch, the better is the utilization of GPU computing capabilities.

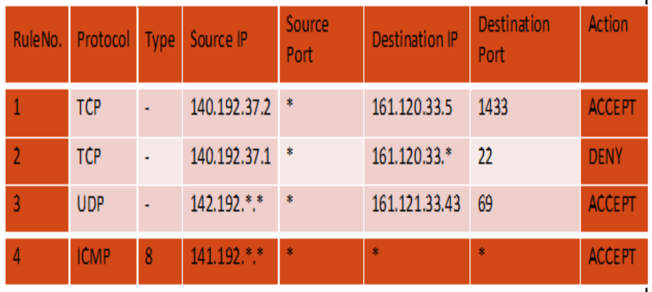
•IP requests locality: In real traffic conditions, IP requests present repetition (e.g., UDP flows, TCP bursts). This feature boosts performance in GPU, since accessing adjacent memory positions is extremely fast in GPU oriented implementations.

•GPU memory coalescence: extremely large FIBs (i.e., more levels in a trie) do not always mean more references in memory and thus lower lookup speed. Due to coalescence in GPU memory, the size of each level of the trie also affects the performance in parallel lookup.

On the other hand, one of the deficiencies of GPU is that host memory and GPU are separate units and thus transfer of data between host and device memory is inevitable. In order to alleviate time consuming I/O transactions, batch of requests can be divided into several groups executed on independent kernels on different streams in order to achieve I/O transactions pipelining [8]. Hardware-based routing table lookup schemes [9] fall in two categories: TCAM-based (TernaryContent Addressable Memory) and SRAM-based (Static Random Access Memory) solutions.

B. *Firewall*

Firewalls usually respond in synergy with IP routers. Firewall architectures have the same unique scope; to examine packet headers under a hierarchical set of rules and either discard or accept them. An indicative example of firewall policy rules is presented in Fig. 2. Without loss of generality, we will focus on the most used but simplest architecture of firewalls, the stateless firewall, to present the algorithmic schemes for purely software and hardware accelerated approaches along with their advantages and tradeoffs.



RuleNo.	Protocol	Type	Source IP	Source Port	Destination IP	Destination Port	Action
1	TCP	-	140.192.37.2	*	161.120.33.5	1433	ACCEPT
2	TCP	-	140.192.37.1	*	161.120.33.*	22	DENY
3	UDP	-	142.192.*.*	*	161.121.33.43	69	ACCEPT
4	ICMP	8	141.192.*.*	*	*	*	ACCEPT

Figure 2. Firewall Rules.

The main pitfall in software-based firewall solutions is the linear search of the ruleset for every packet (ruleset may exceed 1K rules in many cases). Even though, linear search is a non-sophisticated searching algorithm, it is followed by open source and commercial products. Firewall rules usually are dominated by TCP protocol rules. For example, a typical distribution with regards to the protocols is:

75% TCP, 14% UDP, 4% ICMP, 6%, 1% other [13]. Taking into consideration that firewall rules are examined sequentially and UDP rules are usually positioned at the bottom of the hierarchical list, the performance of firewall degrades sharply. Towards this direction research efforts propose several optimization approaches such as rule reordering, multilevel filtering, statistical analysis and data mining techniques to predict the rules best suit to every packet, etc. Since firewalls usually work in synergy with IP routers, GPU is a promising alternative for firewall as well and firewall implementations can exploit the GPU features that are also exploited for routing schemes to speed up their performance.

Hardware-based firewall schemes are also either TCAM-based or SRAM-based solutions. TCAM solutions can perform fast parallel searches over all entries in one clock cycle but are not scalable to bigger rulesets. SRAM-based solutions are more scalable but cannot perform parallel searches.

However, when combined with pipelining and bit vector (BV) splitting algorithms [11] they can perform parallel searches on individual fields of a packet header to increase performance. Finally, when firewall schemes combine BV splitting algorithms, deep pipelining and multiport RAMs in modern FPGAs, these hardware implementations can achieve high performance by processing up to two packets per clock cycle.

III. EXPERIMENTAL RESULTS

In this section, we examine the three considered NFs (routing, firewall, DPI) that are included in the NFV middlebox. We evaluate the achieved performance both in CPU, GPU and FPGA implementations of the above described algorithms for these NFs. We have utilized the implementations of [7] for routing and firewall CPU and GPU implementations, enhanced by essential modification to address the challenges that are discussed in Section 3. Our purpose is to evaluate the throughput of IP lookup and firewall rule traversal algorithms independently, without examining additional overheads from NIC packet buffering. Considered implementations are composed of three consecutive phases: i) pre-shading, ii) shading and iii) post-shading. Pre- and post-shading phases run on CPU worker threads and perform the actual packet batching, host-to-device and device-to-host batch transfers (in case of GPU implementation) and other miscellaneous tasks. Shading process, which realizes the actual NFs functionality is executed either on a single core CPU implementation, or is offloaded to GPU and performed by GPU master threads. Our FPGA routing implementation is based on the architecture that is proposed in [9]. The routing FPGA architecture implements a full binary search trie (BST) for the IP look up process by exploiting FPGA's embedded memories. Pipelining is used to increase the throughput. Our firewall FPGA implementation is based on the architecture that is proposed in [11]. A memory optimized Field Split Bit Vector (FSBV) was implemented that utilize TCAM/CAM for rule fields with a very small number of unique values compared with the ruleset size. Moreover, multiple bits (k bit stride) inspection was exploited to reduce pipeline length and increase the achieved performance [12].

In order to achieve realistic case study conditions, we have exploited real network traffic measurements so as to highlight the benefits and peculiarities of each of the three NFs and their provided solutions, either software or hardware based. Regarding the IP routing NF, our purpose was to examine the effectiveness of FIB storage and FIB traversal. For these reasons, we have exploited CAIDA FIB records so as to construct FIB tries with varying size. In case of software and GPU solutions, multi-bit tries guarantee balanced tries with minimum depth extension, and FIB entries storage in coalescenced memory positions. Regarding the firewall NF, our purpose was to construct firewall rules that conform to typical protocol distribution [13] and examine realistic firewall rules cardinality. Moreover, DefCOM provide us with various traffic patterns (TCP/UDP intensive, TCP/UDP balanced), so as to examine firewall performance under normal and malicious traffic conditions (DoS). In Fig. 3 and Fig. 4 respective results are presented.

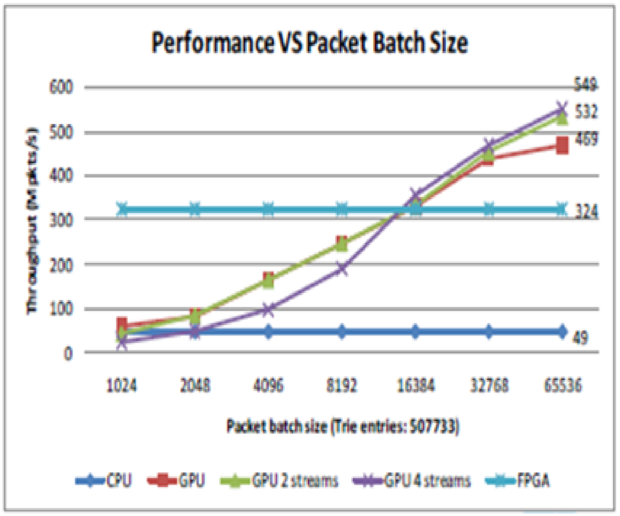


Figure 3. Routing scheme performance analysis: scaling packet batch size.

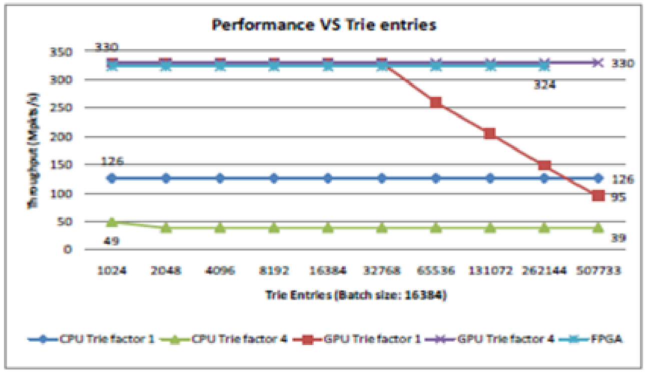


Figure 4. Routing scheme performance analysis: scaling trie entries

In case of routing NF, Figure 3 shows that scaling in batch size boosts up GPU configurations and FPGA and CPU present stable performance. GPU routing implementations achieve a peak performance of more than 500Mpkts/s and FPGA routing implementations can achieve a peak performance of 330 Mpkts/s while CPU-only implementations can achieve a performance of 49Mpkts/s. Moreover, GPU implementation that utilizes 2 streams has better performance compared to the single stream GPU implementation especially in case of heavy traffic (~12% improvement). GPU implementations outperform CPU ones in all cases, presenting up to ~10x acceleration in heavy workloads (65K packets per batch). The use of 4 GPU streams is not efficient in low workload conditions and achieve slightly better throughput (compared to GPU with 2 streams) in traffic congestion. Finally, FPGA implementations have significantly higher throughput in most traffic conditions but are outperformed from GPU implementations in extremely heavy traffic conditions (65K packets per batch). In Figure 4, we also examine the achieved performance in correlation with the FIB entries. There is a 3x performance acceleration in GPU implementation when compared with the CPU implementation when large FIBs are utilized regardless of the trie factor (either 1-stride or 4-stride multi-bit tries in both use cases). Additionally, GPU implementation that exploits 4-stride trie has better performance compared to GPU implementation that utilizes 1-stride due to memory coalescence. In case of CPU implementation, the performance degrades for more complex trie structures. Finally, for FPGA implementation, the performance is stable even when large FIBs are utilized.

We have evaluated the memory utilization for both GPU and FPGA implementation. The GPU implementation utilizes up to 50% of the available memory in our GPU card even for the larger FIB set that we have examined. In contrary, the FPGA routing implementation dominates our chip in large FIB sets since the 256K FIB set needs about 95% of the chip embedded RAM.

Experimental results for the firewall NF when scaling both the packet batch size and the firewall ruleset size are shown in Fig. 5 and Fig. 6, respectively. GPU firewall implementations achieve a peak performance of more than 35Mpkts/s and FPGA firewall implementations can achieve a peak performance of 32 Mpkts/s while CPU-only implementations can achieve a performance of only 2.48 Mpkts/s. The experimental results show a significant performance boost in GPU implementation when large packet batches are utilized. Moreover, the utilization of multiple GPU streams further improves the achieved performance. FPGA firewall implementation has stable performance that is not correlated with packet batch size scaling and is significantly higher than the CPU and GPU implementations in small batch sizes .In large batch sizes, the GPU implementation outperforms the FPGA implementation. When the firewall ruleset increases, the achieved performance is seriously affected in both CPU and GPU implementations as depicted in Fig. 6. Finally, since the FPGA implementation builds a parallel hardware structure for every rule, its performance is not affected by the number of the firewall rules. Finally, in case of heavy traffic load (65K packets per batch) and large input batches GPU outperforms FPGA implementation for real traffic.

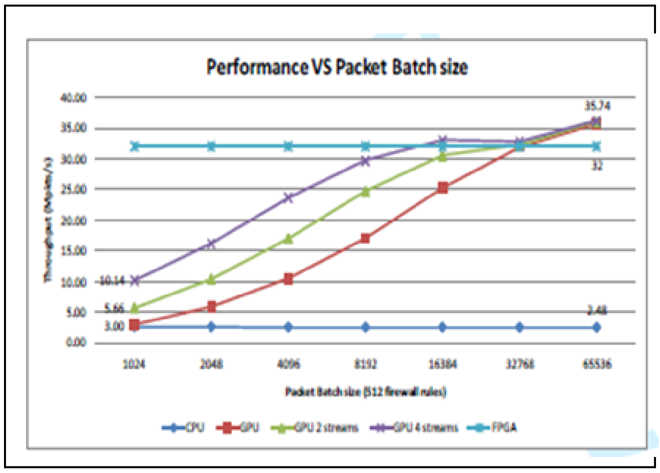


Figure 5. Firewall scheme performance analysis: scaling packet batch size.

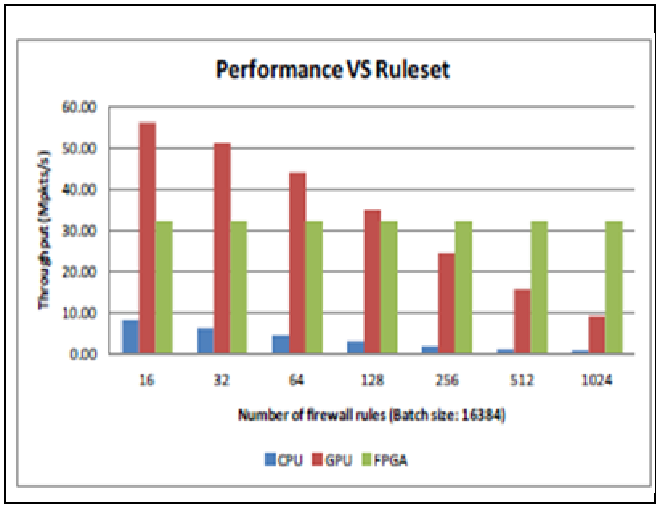


Figure 6. Firewall scheme performance analysis: scaling firewall ruleset size.

Concerning the resource utilization of these implementations, we have evaluated the memory utilization for both GPU and FPGA implementation. The GPU implementation utilizes almost 2% of the available memory in our GPU card even for the larger input batches that we have examined. The FPGA FSBV firewall implementation needs about 18% of the chip embedded RAM even for the larger implemented ruleset (1024 rules) since the chosen architecture is optimized for memory utilization.

The acquired results confirm the feasibility to consider the virtualization of such NFs in the 5G domain, both in the terms of operational performance, scalability but also in terms of their reconfigurability and dynamicity such implementations entail, enabling transparent integration and service continuity between 5G 3GPP and BBF network segments.

IV. Conclusions

We have presented an analysis of two commonplace network functions: a) Routing, b) Firewall in the context of NFV and function “softwarization”. Experimental results for CPU, GPU and FPGA solutions for these functions have been evaluated for their performance to verify the abovementioned functions suitability for incorporation in hardware-accelerated NFV platforms. The results have shown that there are several benefits in scaling, reconfiguration and operational efficiency without compromising performance for the virtualization of NFs studied in the presented work, especially considering the requirements for 5G. As the studied NFs are very important n view also of the needs for Broadband Forum (BBF) and 5G 3GPP interworking for the seamless application provision and QoS requirements between the involved network segments, the results confirm the applicability of such solutions in the 5G domain.

Acknowledgment

The author gratefully acknowledges the NKUA and HUAWEI collaboration project in SDN and the fruitful inputs and comments from the relevant colleagues related to the presented results.

References

[1] ETSI GS NFV 001,"Network Functions Virtualisation (NFV):Use Cases". [Online]. Available from: http://www.etsi.org/technologiesclusters/technologies/nfv [retrieved 11,2012].

[2] V. Sekar, N. Egi, S. Ratnasamy, M.K. Reiter, and G. Shi, "Design and implementation of a consolidated middlebox architecture", Proc. 9th USENIX conference on Networked Systems Design and Implementation (NSDI), 2012, p.24-38.

[3] “*5G: A Technology Vision*”, Position paper, Huawei, [Online]. Available from: www.huawei.com/ilink/en/download/HW_314849 [retrieved: 9,2017].

[4] M. A. Ruiz-Sanchez, E. W. Biersack, and W. Dabbous, "Survey and Taxonomy of IP Address Lookup Algorithms", IEEE Network: The Magazine of Global Internetworking archive, vol. 15, no. 2, pp 8-23, March 2001.

[5] S. Haoyu, M. Kodialam, H. Fang, and T. V.Lakshman, "Efficient Trie Braiding in Scalable Virtual Routers," IEEE/ACM Transactions on Networking, vol. 20, no. 5, pp. 1489-1500, Oct. 2012.

[6] M. Dobrescu, N. Egi, K. Argyraki, B. G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, S. Ratnasamy, "RouteBricks: exploiting parallelism to scale software routers", ACM Symposium on Operating Systems Principles (SOSP), 2009, p. 15-28, ISBN: 978-1-60558-752-3.

[7] S. Han, K. Jang, K. Park, and S. Moon, "PacketShader: a GPU-accelerated software router", ACM SIGCOMM Computer Communication conference (SIGCOMM), pp. 195-206, 2010.

[8] W. Sun and R. Ricci, "Fast and flexible: parallel packet processing with GPUs and click", 9th ACM/IEEE symposium on Architectures for networking and communications systems (ANCS), pp. 25-36, 2013.

[9] H. Le and V. K. Prasanna, "Scalable High Throughput and Power Efficient IP-Lookup on FPGA", 17th IEEE Symposium on Field Programmable Custom Computing Machines (FCCM), April 2009, p. 167- 174 , ISBN: 978-0-7695-3716-0.

[10] U. Mustafa, M. M. Masud, Z. Trabelsi, T. Wood, and Z.Al.Harthi, "Firewall performance optimization using data mining techniques," 9th International Wireless Communications and Mobile Computing Conference (IWCMC), pp. 934-940, 2013.

[11] T. Ganegedara and V. K. Prasanna, "Stridebv 400g+ single chip packet classification", IEEE Conference on High Performance Switching and Routing (HPSR), pp. 1-6, 2012.

[12] H. J. Jung, Z. K. Baker, and V. K. Prasanna, "Performance of FPGA implementation of bit-split architecture for intrusion detection systems", 20th International Parallel and Distributed Processing Symposium (IPDPS), 2006, p. 124-124, DOI: 10.1109/IPDPS.2006.1639434.

[13] D. Rovniagin and A. Wool, "The geometric efficient matching algorithm for firewalls", IEEE Transactions on Dependable and Secure Computing, vol. 8, no. 1, pp. 147–159, 2011.

[14] Broadband Forum TR-203 “*Interworking between Next Generation Fixed and 3GPP Wireless Networks*” Issue: 1 Issue Date: August 2012, [Online]. Available from: https://www.broadband-forum.org/technical/download/TR-203.pdf .