VNF-AAP: Accelerator-aware Virtual Network Function Placement

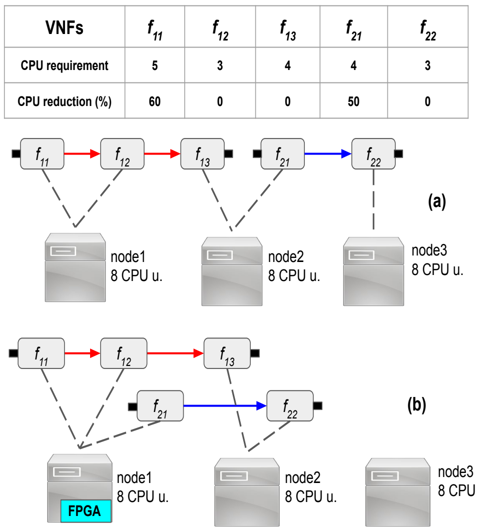
Gourav Prateek Sharma, Wouter Tavernier,
Didier Colle, Mario Pickavet
IDLab, Department of Information Technology
Ghent University - IMEC,
Email: {gouravprateek.sharma, wouter.tavernier, didier.colle, mario.pickavet}@ugent.be

***Abstract*—Network Function Virtualization aims to migrate packet-processing tasks from special-purpose appliances to Virtual Network Functions (VNFs) running on x86 or ARM servers. However, achieving the line-rate packet-processing for VNFs running on CPUs can be a challenging task. External hardware accelerators can be used to offload heavy-lifting tasks (e.g. en/decryption and hashing) from performance-critical VNFs. State-of-the-art VNF placement algorithms only consider compute resources while assigning VNFs on server nodes. We propose a placement algorithm which takes into consideration hardware accelerator resources in addition to compute resources. For evaluation, we compare the performance of our approach with the Integer Linear Program (ILP) method and also with the state-of-the-art best-fit method of VNF placement.**

***Index Terms*—NFV, hardware accelerator, FPGA, placement algorithm, allocation**

I. Introduction

With Network Function Virtualisation (NFV), telecom operators are making a transition in the manner with which network services are created and managed. Network services are normally composed of multiple network functions (NFs) chained together in a specific topology, each NF performing a specific packet-processing task. NFs are traditionally implemented using specialized application-specific integrated circuits (ASICs) called middleboxes. However, network services based on middleboxes result in inflexibility and high CAPEX and OPEX [1]. But significant cost-savings can be achieved by performing packet-processing tasks using virtual network functions (VNFs) running on commercial-off-the-shelf (COTS) servers (e.g. x86 or ARM) instead of doing it using expensive middleboxes. However, softwarization of network functions accompanied by virtualization overhead leads to performance degradation of several network functions, e.g. processing latency of a firewall VNF can go up to 10x as of hardware firewall [1]. Use of external hardware accelerators (e.g. FPGAs, NPUs, GPUs), for offloading CPU intensive tasks from VNFs, has therefore been proposed. The massive parallelism available in hardware accelerators results in a significant throughput and latency improvement. Moreover, hardware accelerators are more efficient in terms of (Number of Operations)/Watt as compared to CPUs.
Placement of network functions across the multiple server-nodes of NFV infrastructure (NFVI) in a fast and scalable manner has been a major research challenge. Various models have been proposed to describe the VNF placement problems [1] [2]. These models consider parameters such as available computational and network resources along-with bandwidth and latency requirements of VNF chains.
With the availability of hardware accelerators in NFVI, VNF placement has made the placement problem even more complicated. This is illustrated via a simple example. Fig. 1 shows a scenario when it is required to place two VNF chains on three server-nodes. CPU requirements (cores) and percentage CPU reduction of VNFs are provided. Let us assume each server-node has 8 units of CPU resources available. All three nodes will be required for the placement of VNFs in case (a) when no acceleration is available on any node. In case (b), we assume the first node is attached with an FPGA board over PCIe bus. VNFs f_{11} and f_{21} now can offload their computational tasks to dedicated accelerators running on the FPGA. Cost-saving can, therefore, be achieved if VNF placement of (b) is followed instead of (a).
In order to optimize the use of resources in NFVI, placement algorithms must take into consideration the presence of accelerators in NFVI nodes. We refer to this problem as VNF-



VNFs	f_{11}	f_{12}	f_{13}	f_{21}	f_{22}
CPU requirement	5	3	4	4	3
CPU reduction (%)	60	0	0	50	0

Fig. 1. Comparison of VNF placement without and without accelerator availability.

AAP, the accelerator-aware VNF placement problem. First, we describe the Integer Linear Programming (ILP) model for VNF-AAP problem in section II. Next, we propose an algorithm based on the best-fit method to solve the VNF-AAP problem in section III. Last, we compare the performance of our algorithm with ILP and the usual best-fit method in section V.

II. Problem Formulation

We formulate the accelerator-aware VNF placement problem as an integer linear program (ILP). Various parameters and decision variables involved in the ILP formulation are described in Table I.

TABLE I
Description of parameters and decision variables

Notation	Description
C	set of all requested chains.
$\mathcal{F}^c$	set of all VNF constituting chain $c \in C$
N	set of all available server nodes.
$\mathcal{R}_{cpu}(n)$	maximum CPU resources (cores) available on $n \in N$.
$\mathcal{R}_{acc}(n)$	maximum accelerator resources (logic elements) available on $n \in N$.
$\mathcal{R}_{bus}(n)$	maximum bus bandwidth (Mbps) on $n \in N$.
A	a set of all the available accelerator types.
$c_0(f^c)$	CPU requirement (cores) of VNF f of chain c.
$c_i(f^c)$	CPU reduction (cores) for VNF f of chain c.
$atype(f^c)$	Type of accelerator required for offloading VNF f^c.
t_c	throughput demand (Mbps) of chain c.
$r(a)$	accelerator resource (logic elements) requirement of the accelerator $a \in A$.
x_n	1 iff atleast one VNF is placed on n
$\alpha^n_{f^c}$	1 iff VNF f of chain c is placed on n.
$\beta^n_{f^c}$	1 iff VNF f of chain c is accelerated on n.
δ^n_a	1 iff accelerator type a is instantiated on the node n

$$obj : \min \sum_{n \in N} x_n \tag{1}$$

$$\sum_{c \in C, f^c \in \mathcal{F}^c} \alpha^n_{f^c} c_0(f^c) - \beta^n_{f^c} c_i(f^c) \leq \mathcal{R}_{cpu}(n) \quad \forall n \in N \tag{2}$$

$$\sum_{a \in A} r(a)\delta^n_a \leq \mathcal{R}_{acc}(n) \quad \forall n \in N \tag{3}$$

$$\sum_{c \in C, f^c \in \mathcal{F}^c} 2\beta^n_{f^c} t_c \leq \mathcal{R}_{bus}(n) \quad \forall n \in N \tag{4}$$

$$\beta^n_{f^c} = 0 \quad \forall n \in N, \forall c \in C, \forall f^c \in \mathcal{F}^c : \text{if } atype(f^c) \notin A \tag{5}$$

$$\sum_{n \in N} \alpha^n_{f^c} = 1 \quad \forall c \in C, \forall f^c \in \mathcal{F}^c \tag{6}$$

$$\beta^n_{f^c} \leq \alpha^n_{f^c} \quad \forall n \in N, \forall c \in C, \forall f^c \in \mathcal{F}^c \tag{7}$$

$$\delta^n_a = \begin{cases} 1, & \text{if } \sum\limits_{\substack{\forall c \in C, \forall f^c \in \mathcal{F}^c, \\ a = atype(f^c)}} \beta^n_{f^c} \geq 1 \\ 0, & \text{otherwise} \end{cases} \quad \forall n \in N, \forall a \in A \tag{8}$$

$$x_n \leq \sum_{c \in C, f^c \in \mathcal{F}^c} \alpha^n_{f^c} \quad \forall n \in N \tag{9a}$$

$$\sum_{c \in C, f^c \in \mathcal{F}^c} \alpha^n_{f^c} \leq M x_n \quad \forall n \in N \tag{9b}$$

$$x_n, \alpha^n_{f^c}, \beta^n_{f^c}, \delta^n_a \in \{0, 1\} \quad \forall n \in N, \forall c \in C, \forall f^c \in \mathcal{F}^c \tag{10}$$

The objective (1) of the ILP is to minimize the total number of server-nodes utilized to place a set of given VNF chain requests. The first constraint (2) ensures that the sum of the effective CPU demands of all VNFs placed on any node does not surpass its maximum CPU capacity. The constraint in (3) indicates the availability of the finite amount of hardware accelerator resources for the instantiation of accelerators. The rate of communication between VNFs and accelerators running is bounded by the maximum bandwidth of the PCIe bus, as indicated in (4). A VNF can be accelerated only if a corresponding accelerator implementation is available in A as represented by (5). Each VNF of every chain is placed exactly once as indicated by (6). Constraint (7) is a consequence of the fact that a VNF f^c can be accelerated on a node n only if it is placed on it. Constraint (8) ensures that an accelerator of a particular type is instantiated if a non-zero number of VNFs are using that accelerator type. The pair of constraints (9a - 9b) forces x_n to be equal to 1 iff a non-zero number of VNFs are placed on n. Constraint (10) ensures variables $x_c, \alpha^n_{f^c}, \beta^n_{f^c}, \delta^n_a$ can only take binary (0 or 1) values.
The solution to the above ILP results in the maximum consolidation of VNFs on a given set of server nodes. However, the VNF-AAP problem with a relatively larger number of chains, VNFs and nodes is challenging problem to solve using the ILP approach. Moreover, for online placement of VNF chains wherein chain requests are arriving dynamically, ILP cannot be used as it requires all the placement requests before running. In the next section, we propose a scalable heuristic method for the placement and accelerator allocation of VNFs in an online fashion.

III. Proposed Algorithm

The algorithm proposed for solving VNF-AAP problem is based on the best-fit strategy, which is usually employed for solving the bin-packing type of problems. The pseudo-code for the algorithm is shown in Alg. 1. Every time a request arrives for the VNF chain placement, Alg. 1 is executed and each VNF of the chain is placed on a node $n \in N$. This algorithm also tries to give access of hardware accelerators to VNFs.

Algorithm 1: VNF placement and accelerator allocation

```
Input: $c$, $N$
Output: $\alpha, \beta$
1  $vnfsList \leftarrow$ sorted list (descending) of $f^c \in \mathcal{F}^c$ in CPU usage;
2  $nodeList \leftarrow$ sorted list (ascending) of $n \in N$ in available CPU resources;
3  for $vnf_c$ in $vnfsList$ do
4      $plc, acc \leftarrow False$;
5      for $n$ in $nodeList$ do
6          if $atype(f^c) \in A$ then
7              if accelVNF ($f^c$,$n$) $== True$ and $\mathcal{R}_{cpu}(n) \geq c_0(f^c) - c_i(f^c)$ then
8                  $plc, acc \leftarrow True$;
9                  break;
10             else
11                 $f_{sel}^{c'} \leftarrow \min_{\forall c, f^c;\ \beta[f^c]=n} c_i(f^c)$;
12                 if Swap($f^c, f_{sel}^{c'}$) $== True$ then
13                     $plc, acc \leftarrow True$;
14                     break;
15                 end
16         end
17         if $\mathcal{R}_{cpu}(n) \geq c_0(f^c)$ then
18             $plc \leftarrow True$;
19             break;
20         end
21     end
22     if $plc == True$ then
23         updateCpuRes ($n$);
24         $\alpha[f^c] \leftarrow n$;
25         if $acc == True$ then
26             $\beta[f^c] \leftarrow n$;
27             updateAccRes ($n$);
28         end
29     else
30         removeVNFs ($c$);
31     end
32 end
```

The input to the algorithm is a placement request c which comprises of a set of VNFs $\mathcal{F}^c$ along-with associated parameters ($c_0(f^c), c_i(f^c)$, $atype(f^c)$) and a set of server nodes N. The first step is to sort all VNFs in $\mathcal{F}^c$ according to their CPU requirements in the descending order (1). Similarly, all the server nodes are ascendingly sorted according to the available CPU resources (2). Placement of VNFs in $vnfsList$ is tried in succession. VNFs for which no accelerator available in A, placement is done using the best-fit manner (17-19). In usual best-fit based placement, a node with least but sufficient CPU resources is selected for the placement of the VNF.
For a VNF f^c, with an available accelerator type in A, the availability of accelerator resources is first checked by the function `accelVNF`. If a sufficient amount of accelerator resources (e.g. logic elements, bus-bandwidth, etc) and CPU is available on n, f^c is placed on n along-with the allocation of an accelerator type $a = atype(f^c)$ (7-9).
In case of insufficient accelerator resources, accelerator access for $f_{sel}^{c'}$ can be swapped with f^c. $f_{sel}^{c'}$ is a VNF running on n with a minimum reduction in CPU requirement $c_i(f^c)$ (11).
If f^c cannot be placed on any of the nodes, other VNFs of chain c are removed from all nodes (30). Otherwise, CPU and accelerator resources are updated accordingly (22-28).

IV. Related Works

The use of hardware accelerators to improve the performance of various VNFs has proposed. In [3], a significant improvement in throughput along-with a reduction in CPU utilization was demonstrated by offloading en/decryption and authentication tasks from the IPSec traffic to a look-aside accelerator. Similarly, the dynamic allocation of accelerators to SSH-tunnels based on CPU usage of VNFs was presented in [4]. OpenANFV is a platform built to support hardware acceleration for VNFs in an elastic and flexible fashion [5]. VNFs such as Deep Packet Inspection (DPI), network deduplication (DeDup) and Network Address Translation (NAT) were shown to be accelerated in terms of throughput.
. VNF placement problem has been an interesting area of research since the early adoption of NFV. Various system models have been proposed which can be used to describe VNF placement problem [cite models] and developing heuristics for efficient solutions. However, none of these models consider resources other than general-purpose resources (e.g. CPU, memory, bandwidth, etc). In [6], authors have proposed an architecture called Uniform Hardware Acceleration Deployment (UHAD) to ease the integration of hardware accelerators present in forwarding nodes (switches and routers) and compute nodes (servers) in the general NFVI. An algorithm is also proposed for allocating accelerator resources to VNFs. However, the placement of VNFs on server-nodes and allocation of accelerators to VNFs is decoupled. This strategy can lead to sub-optimal placement solutions.
Besides [6], most of the research work concerning hardware acceleration in NFV is focused on improving the packet processing performance of VNFs in terms of latency and throughput. Hence, VNF placement models need to be revised in order to consider hardware accelerator resources while making decisions such efficient utilization of resources is achieved.

V. Evaluation

The ILP model discussed in section II was implemented in CPLEX 12.9 framework using doCPLEX Python API and the algorithm proposed in Alg. 1 was written in Python [7]. Evaluations of ILP and Alg. 1 were carried on a Ubuntu 16.04 server running on a Intel Xeon CPU with 16GB of memory. The values (range) of parameters used for the evaluation is given in Table II.

Fig. 2 shows the comparison between the placement efficiency of ILP and our algorithm in terms of the total number of

TABLE II
DESCRIPTION OF PARAMETERS AND DECISION VARIABLES

Parameter	Value or range	Parameter	Value or range		
$\mid C \mid$	10-290	$c_o(f^c)$	1-4		
$R_{cpu}(n)$	16-20	$c_i(f^c)$	(0.40 - 0.60)$c_o(f^c)$		
$R_{acc}(n)$	12-16	$frac_{acc}$	0.30		
$R_{bus}(n)$	80-120	accel. type	a_1	a_2	a_3
chain length	3-6	accel. size	7	5	6

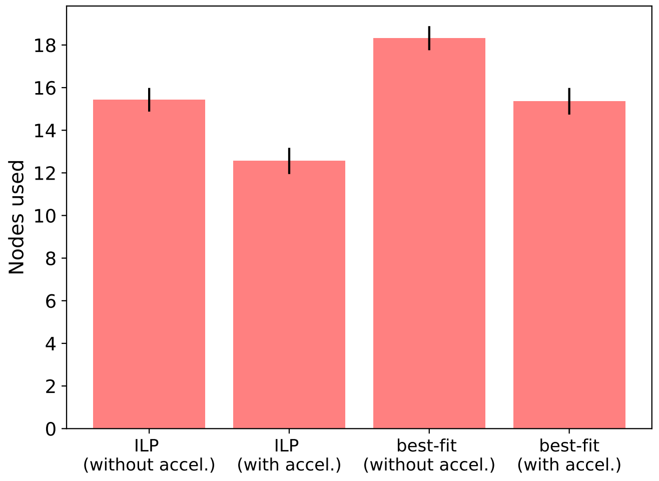


Fig. 2. Nodes required for placing 25 requests with ILP and best-fit algorithm.

nodes required for placing 25 requests. As expected, it can be seen that ILP results in better consolidation of VNFs than our algorithm. The exact-algorithm based on ILP produces global-optimal solution in contrast to the near-optimal solution which is generated as a result making locally optimal decisions at every step of our algorithm. However, VNF placement based on ILP is about three orders of magnitude slower as compared to our algorithm resulting in the issue of non-scalability of this approach in total terms of number of service-chains and server-nodes.
The variation of the total number of required nodes with the changing total number of chain-requests for both cases, i.e., (i) nodes without access to hardware accelerators, and (ii) nodes with access to hardware accelerators is plotted in Fig. 3. As the number of requests is increased, the number of nodes to host VNFs also increases both for case (i) and case (ii). We also observe that for a given number of requests, the total number of nodes required in case (ii) is always less than in case (i). This is a consequence of the additional VNF consolidation, on top of best-fit, due to the release of CPU resources by VNFs which are being offloaded to hardware accelerators.

VI. CONCLUSION

Packet-processing performance of VNFs running on general-purpose compute platforms can be improved by offloading functionalities of specific VNFs to the externally attached hardware accelerators. In this paper, we first presented an ILP model for VNF-AAP problem. Then we proposed a scalable heuristic method for making decisions about VNF placement as well as accelerator allocations to

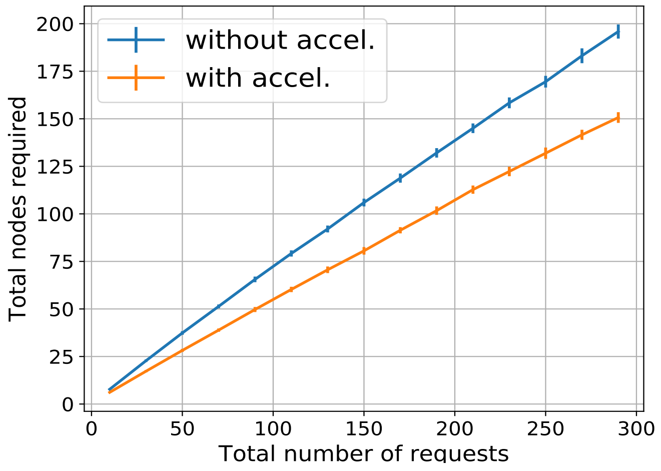


Fig. 3. Performance comparison of VNF placement and accelerator allocation algorithm without and with access to accelerator resources.

VNFs. The evaluation of the algorithm shows that efficient utilization of the physical resources can be achieved in a heterogeneous NFV infrastructure with about 20-25% reduction of the required number of server-nodes.
VNF chaining algorithm with the consideration of delay improvements as a result of hardware accelerators will be investigated in the future. Secondly, we will analyze VNF placement algorithms with an objective to optimize the total cost, i.e., both compute and accelerator costs.

VII. ACKNOWLEDGMENT

This work was funded through NGPaaS, under the grant number 761557, in the scope of the European Commission Horizon 2020 and 5G-PPP programs.

REFERENCES

[1] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. De Turck, and R. Boutaba, “Network function virtualization: State-of-the-art and research challenges,” *IEEE Communications Surveys Tutorials*, vol. 18, no. 1, pp. 236–262, Firstquarter 2016.

[2] X. Li and C. Qian, “A survey of network function placement,” in *2016 13th IEEE Annual Consumer Communications & Networking Conference (CCNC)*. IEEE, 2016, pp. 948–953.

[3] S. Doyle and G. Tkachuk, “Intel IPsec Acceleration,” Intel, Tech. Rep., 2018.

[4] G. P. Sharma, W. Tavernier, D. Colle, and M. Pickavet, “Dynamic hardware-acceleration of vnfs in nfv environments,” in *2019 Sixth International Conference on Software Defined Systems (SDS)*, June 2019, pp. 254–259.

[5] X. Ge, Y. Liu, D. H. Du, L. Zhang, H. Guan, J. Chen, Y. Zhao, and X. Hu, “Openanfv: Accelerating network function virtualization with a consolidated framework in openstack,” in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 353–354.

[6] H. Fan, Y. Hu, S. Zhang, and Q. Ren, “Hardware acceleration resource allocation mechanism for vnf,” *Procedia computer science*, vol. 131, pp. 746–755, 2018.

[7] IBM, “IBM ILOG CPLEX optimization studio.” [Online]. Available: https://www.ibm.com/analytics/cplex-optimizer