Automated Generation of VNF Deployment Rules Using Infrastructure Affinity Characterization

Conference Paper · June 2016

READS
9

5 authors, including:

Michael J Mcgrath
Intel
46 PUBLICATIONS **359** CITATIONS
SEE PROFILE

Michail Alexandros Kourtis
National Center for Scientific Research De…
13 PUBLICATIONS **10** CITATIONS
SEE PROFILE

George Xilouris
National Center for Scientific Research De…
47 PUBLICATIONS **131** CITATIONS
SEE PROFILE

Harilaos Koumaras
National Center for Scientific Research De…
61 PUBLICATIONS **326** CITATIONS
SEE PROFILE

Automated Generation of VNF Deployment Rules Using Infrastructure Affinity Characterization

Vincenzo Riccobene, Michael J. McGrath
Intel Labs Europe,
Leixlip, Co. Kildare, Ireland

Michail-Alexandros Kourtis, George Xilouris, Harilaos Koumaras
NCSR "Demokritos",
Institute of Informatics and Telecommunications,
Athens, Greece

***Abstract*— As the rollout of NFV in the Telecommunications domain gains momentum, methods that enable the automated and performant deployment of VNFs are required to support large scale deployments. This paper presents a framework that enables the automated characterization of VNFs and modelling of the relationships between resource types, quantities and achieved performance. A workload characterization approach is proposed to identify the most appropriate types of resources to be allocated to a VNF at deployment time. Machine learning modelling was used to identify the appropriate deployment configuration which meets required performance targets, such as network throughput. The proposed framework has been experimentally validated based on the optimization of a Traffic Classifier VNF deployment in an OpenStack cloud environment.**

Keywords—SR-IOV, Open vSwitch (OVS), Virtual Network Function Descriptor (VNFD), Network Function Virtualization (NFV), Virtualised Network Function (VNF), Enhanced Platform Awareness (EPA),

I. INTRODUCTION

The advent of the Network Function Virtualization (NFV) paradigm has attracted significant interest from the Telecommunications domain with the promise of network transformation. This evolution is based on moving network functions from a proprietary fixed appliance approach to virtualized software appliances running on standard high volume servers. However, the delivery of this transformation in a performant manner creates a number of significant challenges [1]. With a traditional fixed appliance approach, application software is tightly coupled to its supporting hardware infrastructure. This close coupling supports predictable performance and reliability for carrier grade network functions.

In an NFV approach, virtualized network functions (VNFs) are typically deployed on host Virtual Machines (VMs) running in cloud compute environments such as OpenStack. These environments abstract the underlying infrastructure and have limited capabilities to support intelligent placement decisions. An intelligent placement decision can be described as a placement decision that allocates the optimal quantity and type of resources to a VNF on the most appropriate physical nodes. The decision should also consider the dynamic behavior of the target platform in order to ensure both quantifiable performance and predictable behavior of the workload.

978-1-4673-9486-4/16/$31.00 ©2016 IEEE

While virtualization brings many benefits to Enterprise IT such as workload consolidation, improved utilization of resources etc., realizing these benefits for VNF type workloads creates new challenges [2]. Virtualization can incur performance penalties in comparison to bare metal deployments. Innovations such as Intel® Virtualization Technologies (e.g. VT-x, VT-c and VT-d) have been developed to close the gap with respect to native implementations [3]. Software and hardware packet acceleration technologies, such as Intel's Data Plane Development Kit (DPDK) and Single Route Input/Output Virtualization (SR-IOV) have also emerged in an effort to improve VNF performance [4, 5]. DPDK for example is a software library that supports accelerated packet processing directly in userspace and exploits hardware features available in some Network Interface Cards (NICs). SR-IOV is a standard specification released by the PCI-SIG organization that allows a Peripheral Component Interconnect Express (PCIe) device (including NICs) to appear to be multiple separated physical devices. The key advantage for VMs hosting network related workloads is the ability to use PCI pass-through to gain direct access to an SR-IOV network device with a dedicated physical channel assigned to their network traffic without the need to rely on driver capabilities from the host OS [6]. However, the appropriate composition, configuration and optimization of the virtualized resources and packet acceleration technologies are critical in realizing performant deployments.

In order for NFV to scale appropriately in real world applications, automated deployments are required. From this perspective considerations such as performance optimization, topology, resilience etc. in cloud environments need to be properly addressed at an orchestration level [7]. Deployments objectives such as VNF and network service performance are becoming increasingly important for orchestration in Telco environments [8]. The current approach is based on the use of pre-planned configurations for the node(s) hosting the VNF workload and then simply requesting instantiation of that configuration to bring the VNF into service [9]. This approach has limitations and does not necessarily match the characteristics of the specific workload (e.g. data plane, control plane, signal processing etc.) to the optimized allocation of resources (e.g. packet acceleration technologies). As cloud Data Centers (DCs) become more heterogeneous with various generations of compute resources consisting of different

features and add-on devices (i.e. PCIe) which offer varying hardware acceleration capabilities, this challenge increases significantly. Making automated allocation decisions at deployment time which can reconcile VNF performance requirements against a dynamic resource environment also significantly increases the complexity of VNF orchestration. However, adding more intelligence into placement decisions is critical to ensuring that VNFs are deployed in a performant manner and are not adversely effected by the types or quantities of resources allocated. A number of approaches to VNF/VM scheduling/orchestration policies have been reported in the literature [9], however they generally adopt a high level view to VM placement within an Network Function Virtualized Infrastructure (NFVI) to achieve specific goals such as minimizing intra-data center traffic [10]. These approaches however do not necessarily address the specific mapping of resources to the characteristics and needs of an individual VNF workload. This paper presents a possible approach to addressing this challenge based on the identification of the quantity and types of resources to be allocated to a VNF at deployment time. By generating a model which expresses the allocation of resource in relation to specific levels of performance, rules are generated which can used by an Orchestrator to add intelligence to a VNF deployment request in a cloud environment such as OpenStack.

The paper is organized as follows; Section II presents the problem of supporting the automated orchestration of VNF deployments in a Telco cloud compute environment. Key to solving this problem is matching the characteristics of VNF workloads to their hosting infrastructures. Section III presents the proposed solution and introduces the usage of a machine learning model in order to identify the deployment configuration which can satisfy required performance goals, such as network throughput. In Section IV the experimental configuration of the Traffic Classifier VNF evaluation process in an OpenStack cloud environment is presented. Detailed test case scenarios are described in section V, and the results of the experimental process are discussed in Section VI. Finally, Section VII concludes the paper.

II. Problem Statement

The first generation of VNFs are often based on software porting from the fixed appliance versions of network functions to virtualized forms. There is limited focus on the characteristics of the commodity hardware environments used to host the VNFs beyond the standard abstracted view of resource allocations in virtualized environments i.e. allocations of virtual CPU's, RAM and storage. This approach can result in reduced performance which service providers are willing to accept due to the gain in flexibility that NFV offers. However, as the proliferation of NFV grows, the performance of VNFs will become more and more important, with the expectation that they will approach the performance levels of fixed appliances particularly for network edge functions (e.g. customer premise firewalls, WAN accelerators etc.). Matching the characteristics and performance of VNF workloads to their hosting infrastructures will play an important role in improving their performance and Service Level Agreement (SLA) assurance. However coupling VNF's to specific resource types cannot be at the cost of losing the flexibility provided by virtualized deployments.

Enhanced Platform Awareness (EPA) has emerged in recent versions of OpenStack [11] as a mechanism to address issues relating to resource abstraction which can negatively impact the performance of certain VNF workload types. These workloads can benefit from the access to certain platform features to improve their performance or to increase the predictability and stability of their behavior. For example workloads with cryptography related functionality can benefit from being scheduled on CPUs with Advanced Encryption Standard New Instructions (AES-NI) instructions to improve the speed of encryption and decryption tasks. As EPA evolves in Telco cloud environments, it will bring significant benefits as the richness and granularity of the resource landscape improves. This will help to bridge the current gap between resource abstraction and platform specific requirements of VNFs.

In order to meet customer requirements, service providers need to provide an accurate and reliable indication of performance which is commonly encapsulated in SLAs. These SLAs include targets relating to various aspects of performance expectations. In order to avoid penalties associated with SLA breaches, providers will typically over-provision resources. This approach generally leads to a waste of valuable resources. Secondly, it does not take into account the performance characteristics of available technologies and does not match them to the performance target in an intelligent and optimal fashion.

Therefore, a clear problem exists in relation to matching the performance characteristics of platform resources that are discoverable through EPA to meet the performance targets of a VNF which may be defined in an SLA. Additionally, the specific quantity of any given resource should also match the desired performance (for example the number of virtual CPUs to be assigned to a workload or the number of SR-IOV network connections to be used, and so forth). Finally any solution must be implemented in an automated fashion to support intelligent orchestration of the VNF deployment process.

III. Proposed Solution

For any VNF type workload a detailed understanding of the performance characteristics and affinity for physical resources is important from a deployment perspective in order to maximize performance and to optimize the number of allocated resources. Optimization can also reduce the financial costs of deployments through optimal resource allocation (i.e. reduction in the over-allocation of resources, or not allocating more expensive resources which have no significant impact on the performance of the VNF workload being deployed). It is necessary to build a picture of the VNF's affinities for compute resource allocations and platform specific features. To appropriately map these affinities requires the deployment of various compute resource allocation cases and to quantify the performance of a VNF using embedded telemetry in a statistically meaningful manner. Even for relatively simple VNFs with constrained boundary conditions in terms of resource allocation ranges, tens or hundreds of distinct configurations are initially required, which can quickly grow into thousands of deployments to ensure data robustness. It is impractical and expensive to deploy thousands of potential VM flavor combinations manually; therefore for the purpose of this

work a framework was developed which automates the deployment process. The framework has two distinct functions: the first one is the characterization of VNF workload performance using different resource allocations in a quantitative manner; the second one is modelling the relationships between VNF performance and resource allocations.

The framework developed (see Figure 1) is based on a Python implementation which automatically generates different deployment configurations for a given VNF under test. The templates represent all possible configurations for selected parameters (e.g. vNIC configuration, number of virtual CPUs, amount of RAM, etc.) within a given range of values. The framework interacts with an OpenStack cloud environment and generates different configurations in the form of Heat templates, which are deployment scripts based on the Heat Orchestration Template format [12]. Templates are then deployed sequentially and data from each deployment is collected and analyzed in order to evaluate the effect of a given deployment configuration on the VNF's performance. The key steps are as follows:

1. Each template is deployed through OpenStack (namely the Heat service) which creates the VMs, installs the VNF software and a telemetry agent to measure internal metrics at a 1 Hz sampling interval. Details of the telemetry agent and supporting backend have been previously reported [13];
2. Once the deployment phase is concluded, the framework starts a packet generator sending network traffic to the VNF;
3. Traffic is processed by the VNF and is forwarded to a test point destination for performance measuring purposes;
4. After a predefined time interval, the packet generator is stopped and all data collected during the experiment are saved to a Hadoop database.
5. The deployed test case is deleted and the test environment is reinitialized ready for the next test case.
6. Steps 1-5 are repeated for each template.

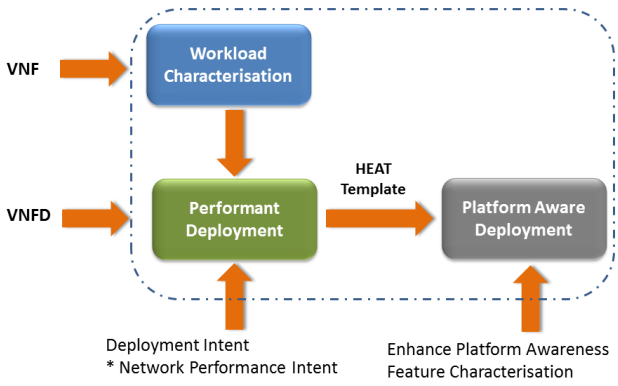

Fig. 1. High level framework for determining optimised VNF deployments

The data collected is then analyzed using a machine learning approach to identify relationships between the types and quantity of resource allocated and VNF performance. A decision tree is generated which relates specific performance characteristics (such as network throughput) to the deployment configurations. An example of a decision tree is described in Section VI. The decision tree would be generated during acceptance testing for a new VNF prior to operational deployment by a service provider. The decision tree can then be encoded for use by an Orchestrator to optimize the allocation of specific resources during automated deployments. Finally, EPA is used to identify the location of a host which has the necessary resources.

IV. Experimental Configuration

A. *Traffic Classifier VNF*

The Traffic Classifier (TC) VNF used comprises of two Virtual Network Function Components (VNFCs), namely a Traffic Inspection engine and a Classification and Forwarding function. The two VNFCs are implemented in respective VMs. The Traffic Classifier is based upon a Deep Packet Inspection (DPI) approach, which is used to analyze a small number of initial packets from a flow in order to identify the flow type. After the flow identification step no further packets are inspected. The Traffic Classifier follows the Packet Based per Flow State (PBFS) in order to track the respective flows [14]. A flow can be defined as a sequence of packets sent from one application to a receiving application. The PBFS method uses a table to track each session based on the 5-tuples (source address, destination address, source port, destination port, and the transport protocol) that is maintained for each flow. As an application flow is typically comprised of multiple packets, once the packet initially received has been marked as belonging to that application all subsequent packets in the flow are marked similarly. The TC uses the Type of Service (TOS) field in the packet header to identify the flow type.

Both VNFCs can run independently from one another, but in order for the VNF to have the expected behavior and outcome, the 2 VNFCs are required to operate in a parallel manner.

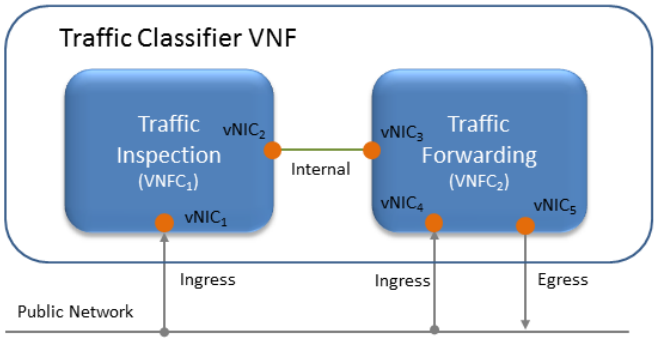

Fig. 2. Traffic Classifier Architecture

The Traffic Inspection VNFC is the most processing intensive component of the VNF. It implements the filtering and packet matching algorithms in order to support the traffic forwarding capability of the VNF. The component supports a flow table (exploiting hashing algorithms for fast indexing of

flows) and an inspection engine for traffic classification. The implementation used for these experiments exploits the nDPI library [15]. The packet capturing mechanism is implemented using libpcap. This component does not block unidentified traffic as traffic is mirrored to both VNFCs. When the DPI engine identifies a new flow, the flow register is updated with the appropriate information and communicated to the Traffic Forwarding VNFC which then applies any required policy updates.

The Traffic Forwarding VNFC is responsible for routing and packet forwarding. It accepts incoming network traffic, consults the flow table for classification information for each incoming flow and then applies pre-defined policies (e.g. Type of Service/Differentiated Services Code Point (TOS/DSCP) marking to be applied to multimedia traffic for QoS enablement [16]) on the forwarded traffic. It is assumed that the traffic is forwarded using the default policy until it is identified and new policies are enforced. The expected response delay is considered to be negligible, as only a small number of packets are required to achieve identification. In a scenario where the VNFCs are not deployed on the same compute node, traffic mirroring may introduce additional overhead.

B. *Networking Traffic Generation Setup*

The anticipated performance of the TC VNF is dependent on the following factors with respect to the traffic profile, namely (i) the number of flows; (ii) flow duration; (iii) stateful protocol matching versus static port. In this context a traffic profile which consists of high volume and short-lived flows consumes a higher level of compute resources (CPU and memory) in order to support classification of the new flows. Additionally, other factors that affect the performance of the VNF are dependent solely on its configuration and enabled features. These factors are: (i) the number of protocols being matched against, (ii) the number of regular expressions used and (iii) matching algorithm complexity. Throughout the experimental protocol the same mixture of traffic and VNF setup was utilized in order to minimize any influence from the factors as outlined above which can affect the VNF's performance behavior.

The traffic profile used in these experiments was based on real traffic traces captured in the NCSR "Demokrito's" network. In total the captured traffic contained 36726 unique flows utilizing 28 different application protocols. The packet capture (PCAP) files were replayed during the experiments using a packet generator capable of reaching 10 Gbps line rates. The packet destination addresses in the PCAP files were re-written and replaced with a multicast destination address. This modification is necessary as the network traffic needs to be mirrored to both VNFCs. This method was preferred over performing traffic mirroring at the Open vSwitch level (at the Compute Node), which could introduce additional latencies and increase resource utilization and it is difficult to automate.

C. *OpenStack Testbed*

The architecture of the experimental configuration shown in Figure 3 comprises of an OpenStack Juno based cloud environment, which includes: a controller (Intel® Core™ i7, @ 3.40GHz 32GB RAM), two compute nodes (dual socket Intel® Xeon® E5 2680 v2@ 2.80GHz with 10 cores, 64GB RAM) and a traffic generator (Intel® Core™ i7@ 3.40GHz, 32 GB RAM) connected on the same network domain through a 10 Gbps switch. All compute nodes used Intel® Dual Ethernet Converged Network Adapters (X540-T2). The compute node configuration enabled the hosted VMs to use both virtual NICs (vNICs) connected to an Open virtual Switch in the form of a software-assisted network solution and physical NICs with PCIe pass-through, exploiting SR-IOV functionality in the form of a hardware-assisted solution. A packet generator was used to stress test the performance of the VNF under test. The deployment process in the testbed was controlled by a workload characterization and modeling framework which is described in the next section.

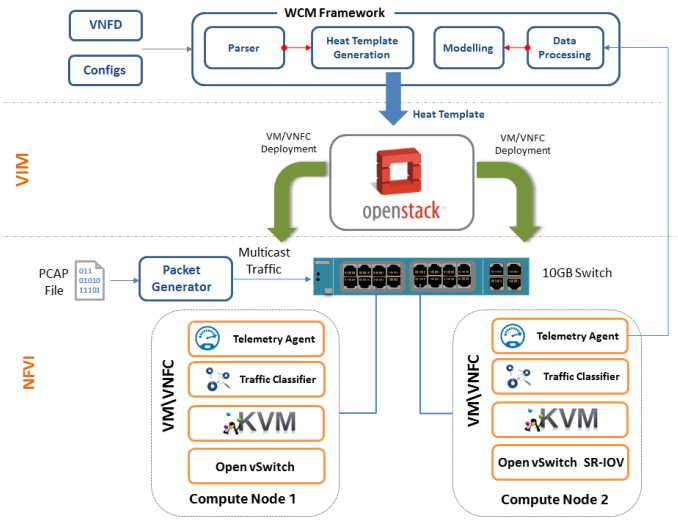

Fig. 3. High level architecture of experimental testbed.

D. *Workload Characterization and Modeling Framework*

The Workload Characterization and Modeling (WCM) framework characterizes VNF workloads under test. The key function of the framework is to measure the VNF's performance with different configurations and to automatically generate a model to support automated selection of the most efficient configuration at deployment time. It takes as its input an ETSI compliant Virtual Network Function Descriptor (VNFD) and a set of configuration parameter ranges which need to be applied in order to investigate different VNFCs' configurations.

Configurations are given to the framework as a list of parameters and values (e.g. type of virtual NICs, number of vCPUs, etc.). The configuration file can also include the desired network performance: expressed in bits per seconds and represents the target bit rate that the VNF service provider wants to offer to their users. For example, given 1 Gbps, 5 Gbps and 10 Gbps network throughput values, the system creates a model to select the most efficient configuration for each one of these performance goals. The model takes the form of a decision tree, whose rules depend on the variables given as an input to the framework.

V. Test Case Scenarios

The following section describes two use case scenarios where the framework was applied.

A. OVS vs SR-IOV Characterization

The first use case scenario is focused on a performance comparison between OVS and SR-IOV network connections. The framework was used to investigate the effect of allocating OVS and SR-IOV network connections to the VNFCs comprising the VNF. As shown in Figure 2, the VNF has 5 network connections providing ingress, egress and inter VNFC connectivity. For the purpose of this comparison the framework generates two Heat templates reflecting these two configurations with the two instances generated being deployed on two different physical hosts at the same time. As shown in Figure 4 each instance had dedicated destination test endpoints (TP_2 and TP_3) implemented as virtual machines, which were used to measure the amount of traffic the VNF was able to forward. A third endpoint destination (TP_1) acted as a network throughput reference point by intercepting all traffic sent by the traffic generator. A traffic generator [17] was used to generate multicast traffic flows that were simultaneously sent to both VNFs and TP_1. The traffic rate was accelerated at runtime in order to send a linearly increasing traffic profile, ranging from 1Mbps to 9.6 Gbps. The results obtained are discussed in Section VI.

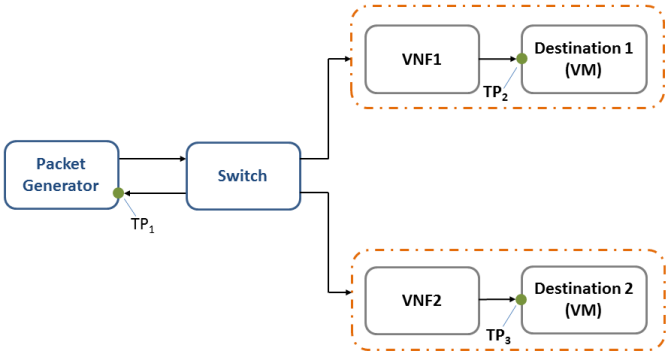

Fig. 4. Testpoint configuration for VNF testing

B. Performance Optimized Configuration

The second test case investigated an approach which supports VNF deployments based on network performance optimization. The main purpose of the test case was to implement an automated system capable of selecting the minimal configuration for a VNF that can provide the desired level of performance specified in an SLA. This relates to a scenario where a Service Provider wishes to provide the same service to many users with different SLAs by using different VNF flavors. The concept of flavor in this context is distinctly different to OpenStack's VM flavor. Here it is intended to define the complete deployment configuration of all the components of a VNF. It includes the configuration of the network interfaces, the scheduler hints and other elements necessary to instantiate the VMs in additional to the required allocations of vCPUs, RAM and storage. In order to select the VNF flavors, a set of experiments were carried out using the framework to collect and analyze the data based on various deployment configurations (SR-IOV allocation and VM sizes). Three bandwidth SLA targets were selected (400 Mbps, 800 Mbps and 3 Gbps). The data collected was analyzed using the C4.5 machine learning algorithm [18]. Specifically, the framework invokes the J48 module of the WEKA machine learning platform which provides an implementation of the C4.5 algorithm and returns a decision tree as its output. The approach is designed to support the automated building of VNF flavors, according to the network performance requirement of the customer, selecting the most efficient configuration (in terms of usage of resources) which can fulfil the required SLA.

In order to relate performance optimization to a specific configuration, an efficiency function was defined and calculated for each configuration to compare them and to select the most efficient configuration that can support the SLA. The function is defined as follows:

$$E = \frac{P}{\sum_{i=1}^{N}(w_i \cdot R_i)}$$

For each sample:

- P is the performance of the VNF (for the purpose of the use case, throughput was specifically addressed);
- N is the number of variables taken into account by the analysis (in this case is 2, vCPUs and RAM);
- w_i is a weight assigned to each resource by a service provider (the sum of all w_i is equal to 1);
- R_i is the number of units of resource i allocated in the configuration (this is subject of a min/max normalization with respect the resource with higher value, which in this case is RAM).

R_i depends on the number of available resource of type i that are allocated. This number is normalized within the value range related to a resource type that has the maximum number of resources available. For example, a deployment configuration may include resources such as processor, memory etc. For example, a maximum of 10 CPUs and a maximum of 20 GB of memory may be available to the deployment configuration. In this example, the minimum memory size may be one gigabyte. Thus, 20 GB of memory corresponds to 20 memory resources and 10 CPUs corresponds to 10 processor resources. Thus, 20 is the maximum number of resources of any resource type in this example. If the number of allocated CPUs is 5 and the amount of memory allocated is 5GB, the amount of allocated CPUs correspond to the 50% of the available ones (5/10 = 0.5), whereas the amount of allocated memory unis correspond to the 25% (5/20 = 0.25); then R_i for resource type processor is 0.5 x 20 =10, whereas R_i for resource type memory is 0.25 x 20 = 5

VI. Results and Discussion

This section presents the results relating to the test case scenarios described in the previous section.

A. OVS vs SR-IOV Characterization

The results of the characterization experiments are shown in Figure 5. The network traffic load was increased linearly from 1 Mbps to 9.6 Gbps over the course of the experimental run which was approximately 200 seconds in duration. The VNF deployment utilizing OVS exhibited network saturation effects at approximately 360 Mbps. The VNF deployment utilizing SR-IOV exhibited a 10 fold improvement over OVS with saturation occurring at approximately 3.6 Gbps. However the results also

indicate a significant gap between the throughput achieved by the forwarding VNFC and the total traffic load.

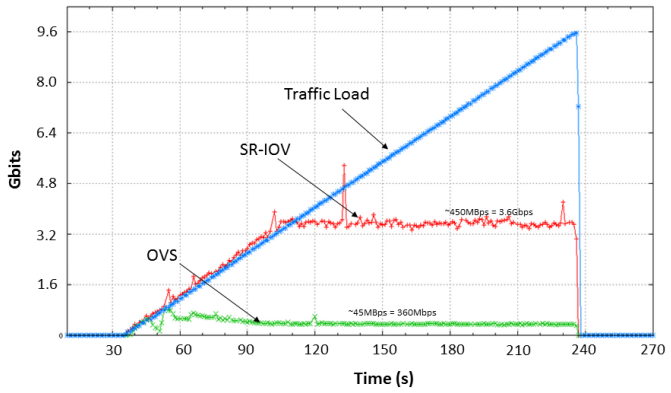

Fig. 5. SR-IOV vs OVS throughput.

The VNF also generates metrics to monitor its internal performance which are typically used by a VNF Manager (VNFM) to manage the VNF when deployed. For the purposes of this test case these metrics were captured and integrated into the telemetry platform for analysis. In Figure 6, the number of flows detected per second by the VNF is shown. The ability of the VNF to detect unique flows is significantly affected by the allocation of SR-IOV or OVS and corresponds with the pattern of behavior shown in Figure 5.

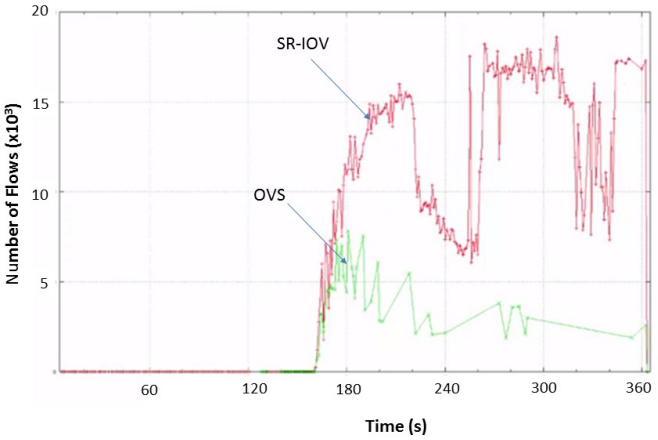

Fig. 6. SR-IOV vs OVS number of detected flows.

B. *Automatic Performance Configuration*

In order to explore different network performance targets, 32 different configurations which varied network connection type between SR-IOV and OVS to the VNFCs of the VNF were investigated. The test scenarios were automatically provisioned, tested and analyzed using the framework, generating all possible combinations of the variables (5 vNICs with 2 allowed values). Each configuration was tested 3 times using 60 second test durations. Over 6000 data points were generated and analyzed using the J48 algorithm [19] to generate a decision tree as shown in Figure 7.

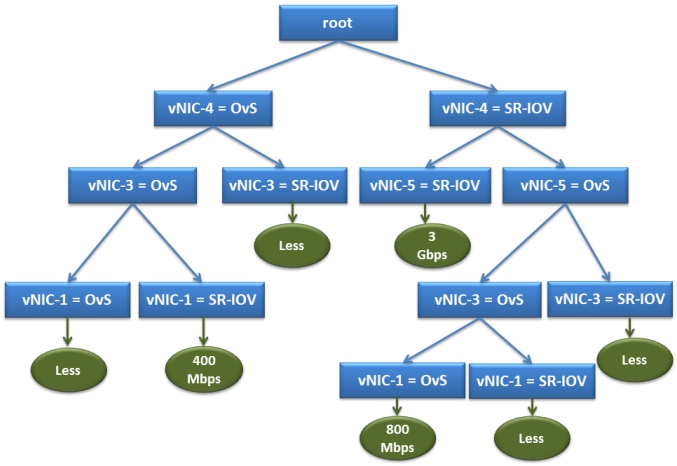

Fig. 7. Decision Tree returned by the J48 algorithm for the networkconnection type.

Using the rules generated by the decision tree it is possible to automate the selection of a VNF configuration to achieve a given SLA. The proposed approach uses the decision tree to generate the requirement configuration for a given deployment. The algorithm starts by navigating the leaves and selecting the leaf that corresponds to the desired network throughput. The selection of values for the configuration parameters is simply obtained by traversing the tree from the selected leaf to the root. For instance, the minimal configuration to achieve a forwarding throughput of 3 Gbps is achieved by allocating SR-IOV to vNICs 4 and 5 both and OVS to the other interfaces. In situations where more than one configuration is returned for a given SLA, the deployment configuration is selected by the system according to the efficiency function defined in Section V. In this way it is possible to achieve the target SLA while avoiding excessive allocation of resources. As the framework automates the completely process including the generation of the decision tree to define the deployment rules, only limited user intervention is required. Setup of the initial configuration file setup is the only key user task.

The same experiments were performed for various flavors of the Forwarding VNFC, specifically varying the quantity of vCPUs and of RAM assigned to the VM. For this experimental configuration SR-IOV channels were statically assigned to the vNICs 1, 4 and 5. The number of vCPUs assigned to the VM was varied between 1 and 10 and RAM allocations between 2 and 8 GBs in 1 GB incremental steps. The total number of possible configurations for this experiment was 70. Each configuration was tested 3 times over a 60 second duration using the framework. In total 14,000 data points were collected which were used for model generation in WEKA.

The results obtained did not indicate any particular dependency on the size of the VM i.e. the VNFC was not CPU or memory bound; therefore a minimum VNF flavor (1 vCPU, 2 GB RAM) was sufficient to achieve a 3 Gbps traffic throughput. Of course, this result is specific for the virtualized Traffic Classifier under test and for the physical infrastructure setup utilized. The J48 algorithm did not return any decision tree.

To check the validity of the previous results, a deployment based on a large resources allocation and a framework based allocation were generated and measured. A side to side performance comparison of the respective deployments is shown in Figure 8. The network throughput achieved is the same for both deployments. However, in the second deployment the result is achieved with a significantly smaller allocation of resources, which provides tangible support for the approach outlined in this paper.

The approach described in this paper is generalizable in terms of KPI targets. While throughput was considered for the vTC use case to demonstrate the application of the modelling approach developed. This approach can readily be extended to different KPI targets for other VNFs as necessary

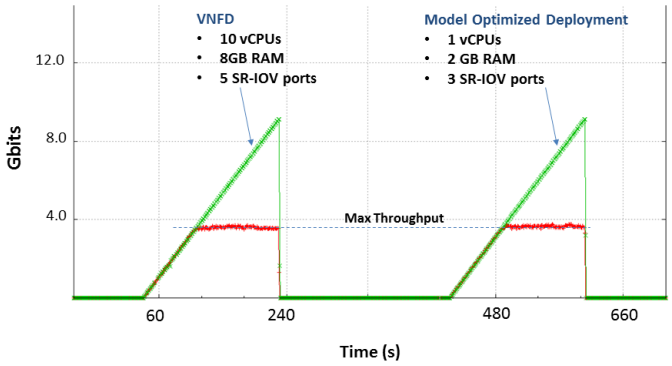

Fig. 8. Effect of vCPU's allocation (High and Low performance evenly distributed indicating no influence

VII. Conclusions and Future Work

Utilization of a VNFD which provides a pre-planned deployment configuration is the current industry approach to support the automated deployment of VNFs in cloud computing environments. However this has limitations from an orchestration perspective and does not take into account underutilization of compute resources or the allocation of specific resource types which can benefit workload performance. Secondly the approach will have scalability challenges as hundreds or thousands of templates will have to be maintained as adoption of NFV grows within a network service provider environment. To address this limitation, the potential of the automated generation of deployment rules based on infrastructure affinity awareness has been investigated. The effect of the deployment configuration, i.e. the *VNF flavor*, on the performance has been demonstrated using a comparison between OVS and SR-IOV network technologies. In addition, an approach for the automatic selection of the best VNF flavor has been proposed and a framework has been designed and developed to support dynamic configuration selection at deployment time.

It has been shown that this approach can be used to define appropriate deployment rules in an automated manner which can be used to orchestrate the deployment of a VNF combining optimized network related performance with a minimal configuration of allocated resources i.e. consumes the smallest amount of resources (e.g. number of SR-IOV channels, number of vCPUs, amount of RAM, etc.).

As previously outlined EPA has emerged to support the allocation of specific platform capabilities to the VMs hosting VNF workloads at deployment time, which can benefit VNF performance in a cloud environment [20]. EPA contributions are now starting to appear in OpenStack releases and include platform detection capabilities through the discovery, tracking, and reporting of enhanced features in the physical architecture and peripherals. Additionally, there are capabilities to schedule and deploy a VM onto a selected platform with the enabled features. This capability will enable the orchestrator to close the loop by allowing the requested resource types defined in the deployment rules to be scheduled appropriately during VNF deployment.

Additional work is planned to improve the capabilities of the framework and to extend the results further. From an SLA perspective, only bandwidth has been considered to date. Further developments will include the definition and automatic evaluation of more complex SLAs to take into account for example jitter, latency packet loss etc. From a technology characterization perspective, additional platform features will be investigated to determine both their individual and cumulative effect on VNF performance; examples include core pinning, NUMA awareness and huge pages.

Acknowledgment

This work was undertaken under the Information Communication Technologies, EU FP7 T-NOVA project, which is partially funded by the European Commission under the grant 619520.

References

[1] K. LeBlanc, (2015). Performance – Still Fueling the NFV Discussion. Available: https://www.sdxcentral.com/articles/contributed/vnf-performance-fueling-nfv-discussion-kelly-leblanc/2015/05/?utm_source=sdnc_slider&utm_medium=link&utm_campaign=links

[2] L. R. Battula, "Network Security Function Virtualization (NSFV) towards Cloud computing with NFV Over Openflow infrastructure: Challenges and novel approaches," in Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on, 2014, pp. 1622-1628.

[3] A. Semnanian, J. Pham, B. Englert, and W. Xiaolong, "Virtualization Technology and its Impact on Computer Hardware Architecture," in Information Technology: New Generations (ITNG), 2011 Eighth International Conference on, 2011, pp. 719-724.

[4] DPDK.org. (2015). DPDK: Data Plane Development Kit. Available: http://dpdk.org/

[5] J. DiGiglio and D. Ricci. (2013), High Performace, Open Standard Virtualisation with NFV and SDN. Available: www.intel.eu/content/dam/www/public/us/en/documents/white-papers/communications-virtualization-snd-nfv-paper.pdf

[6] L. Jiuxing, "Evaluating standard-based self-virtualizing devices: A performance study on 10 GbE NICs with SR-IOV support," in Parallel & Distributed Processing (IPDPS), 2010 IEEE International Symposium on, 2010, pp. 1-12.

[7] S. Oechsner and A. Ripke, "Flexible support of VNF placement functions in OpenStack," in Network Softwarization (NetSoft), 2015 1st IEEE Conference on, 2015, pp. 1-6.

[8] J. Keeney, S. van der Meer, and L. Fallon, "Towards real-time management of virtualized telecommunication networks," in Network and Service Management (CNSM), 2014 10th International Conference on, 2014, pp. 388-393.

[9] K. Giotis, Y. Kryftis, and V. Maglaris, "Policy-based orchestration of NFV services in Software-Defined Networks," in Network Softwarization (NetSoft), 2015 1st IEEE Conference on, 2015, pp. 1-5.

[10] M. Yoshida, W. Shen, T. Kawabata, K. Minato, and W. Imajuku, "MORSA: A multi-objective resource scheduling algorithm for NFV infrastructure," in Network Operations and Management Symposium (APNOMS), 2014 16th Asia-Pacific, 2014, pp. 1-6.

[11] OpenStack. (2015). Enhanced-platform-awareness-pcie. Available: https://wiki.openstack.org/wiki/Enhanced-platform-awareness-pcie

[12] OpenStack. (2015). Heat Orchestration Template (HOT) Guide. Available: http://docs.openstack.org/developer/heat/template_guide/hot_guide.html

[13] P. Veitch, M. J. McGrath, and V. Bayon, "An instrumentation and analytics framework for optimal and robust NFV deployment," Communications Magazine, IEEE, vol. 53, pp. 126-133, 2015.

[14] N. Cascarano, L. Ciminiera, and F. Risso, "Optimizing Deep Packet Inspection for High-Speed Traffic Analysis," Journal of Network and Systems Management, vol. 19, pp. 7-31, 2011.

[15] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "nDPI: Open-source high-speed deep packet inspection," in Wireless Communications and Mobile Computing Conference (IWCMC), 2014 International, 2014, pp. 617-622.

[16] Cisco. (2008). Implementing Quality of Service Policies with DSCP. Available: http://www.cisco.com/c/en/us/support/docs/quality-of-service-qos/qos-packet-marking/10103-dscpvalues.html

[17] Wind River Systems, (2015). Pktgen-DPDK. Available: https://github.com/Pktgen/Pktgen-DPDK/

[18] J. R. Quinlan, C4.5 Programs for Machine Learning. San Mateo, CA: Morgan Kaufmann, 1993.

[19] R. Arora and S. Suman, "Comparative Analysis of Classification Algorithms on Different Datasets using WEKA," *International Journal of Computer Applications,* vol. 54, pp. 21-25, 2012.

[20] A. Hoban, P. Czesnowicz, S. Mooney, J. Chapman, I. Shaula, R. Kinsella, and C. Buerger. (2015), A Path to Line-Rate-Capable NFV Deployments with Intel® Architecture and the OpenStack® Juno Release. Available: https://networkbuilders.intel.com/docs/PathToLine-Rate_WP_V1.pdf