<http://www.diva-portal.org>

Postprint

This is the accepted version of a paper presented at *IEEE Conference on Network Function Virtualization and Software defined Networks, Verona, Italy, 27-29 November 2018*.

Citation for the original published paper:

Gokan Khan, M., Taheri, J., Kassler, A., Darula, M. (2018)

Automated Analysis and Profiling of VirtualNetwork Functions: the NFV-Inspector Approach

In: *2018 IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)* IEEE

<https://doi.org/10.1109/NFV-SDN.2018.8725697>

N.B. When citing this work, cite the original published paper.

© 2018 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

Permanent link to this version:

<http://urn.kb.se/resolve?urn=urn:nbn:se:kau:diva-71388>

Automated Analysis and Profiling of Virtual Network Functions: the NFV-Inspector Approach

Michel Gokan*, Javid Taheri*, Andreas Kassler*, Marian Darula†

*Department of Computer Science

Karlstad University, Karlstad, Sweden

Email: {michel.gokan, javid.taheri, andreas.kassler}@kau.se

†R&D Technology and Industry

Ericsson, Stockholm, Sweden

Email: marian.darula@ericsson.com

Abstract—Discovering insights about Virtual Network Function (VNFs) resource demand characteristics will enable cloud vendors to optimize their underlying Network Function Virtualization (NFV) system orchestration and dramatically mitigate CapEx and OpEx spendings. However, analyzing large-scale NFV systems, especially in mobile network environments, is a challenging task and requires tailor-made approaches for each particular application. In this demo, we showcase NFV-Inspector, an open source and extensible VNF analysis platform that is capable of systematically benchmark and profile NFV deployments. Based on its pluggable framework, NFV-Inspector classifies VNFs resource demand characteristics and correlate their Key Performance Indicators (KPIs) with system-level Quality of Service (QoS) measurements.

Keywords—Network Function Virtualization, Profiling, Classification, Platform, Quality of Service.

I. INTRODUCTION

One of the main advantages of using Network Function Virtualization (NFV) architecture is the elastic and flexible deployment of Network Functions (NFs) across general purpose hardware through leveraging virtualization technology instead of using proprietary devices. Combining NFV with Software-Defined Networking (SDN) makes networks even agiler by enabling programmable changes in network topologies and configurations. However, alongside the dynamicity that NFV/SDN approaches bring to networks, they also impose many performance challenges that require considerable amounts of optimization efforts –both in industry and academia– to smoothen the transition between hardware-based NFs and software-based Virtual Network Functions (VNFs).

In order to optimize NFV systems, administrators/agents need to have a fairly complete view regarding required resources and operational behavior of such systems, including, (1) VNFs resource demand characteristics under different traffic loads, (2) impact of different NFV Infrastructure (NFVI) configuration on Quality of Service (QoS) measurements, and (3) correlation between VNFs Key Performance Indicators (KPIs) and QoS measurements.

In this demo, we showcase NFV-Inspector [1], an open-source and extensible platform for analyzing and profiling of VNFs. This platform has a microservice architecture that is developed using Node.js [2] and Python [3]. NFV-Inspector au-

tomates the process of data gathering, knowledge extraction and analysis of VNFs. Using NFV-Inspector, we will also demonstrate how it can be used for VNF classification as well as correlating QoS measurements with system KPIs.

II. NFV-INSPECTOR ARCHITECTURE

NFV-Inspector architecture is comprised of the following four main microservices:

NFV-MON to monitor system KPIs, gather benchmark reports, and form a training set for machine learning or statistical analysis. It consists of two main parts: (1) NFV-MON clients that are installed inside each Virtual Machine (VM) or container to send resource usages and system KPIs, and (2) an NFV-MON server that is installed on a dedicated server –either bare-metal or a VM– to receive all the collected data from NFV-MON clients.

On the client side, we used *collectd* [4], an open source system statistics collection daemon, to collect metrics and system KPIs. On the server side, we used *Logstash* [5] to receive collected metrics from NFV-MON client, and *Elasticsearch* [6], an open source and distributed search engine and time-series database, to store and aggregate all received measurements.

NFV-BMS to run/launch third-party external benchmarks and report QoS measurements (e.g., average response time and/or timeouts) to the NFV-MON. We used a plugin architecture to enable integration with external benchmarking tools; we also developed a CLI framework in Node.js, to control NFV-Inspector operations. We used *MariaDB* [7], a relational database emerged through a community fork from *MySQL* [8], to store experiments start/stop timestamps and benchmark configurations.

NFV-VMS to automate the entire process of orchestrating and running commands on VMs/containers/hosts; it acts as an abstraction layer between the NFV-BMS and NFV systems. We used *Ansible* [9] for infrastructure deployment/configuration automation. To date, we developed integration plug-ins to communicate with *OpenStack* [10] for VM-based NFV systems; integration plug-ins for container-based environments such as *Kubernetes* [11] is under development.

Fig. 1: NFV-Inspector system architecture

NFV-LAB, the core analytical component of NFV-Inspector, fetches training sets from the NFV-MON and performs various analytical calculations. It also has a pluggable architecture so that developers can write their own analytical extensions to perform machine learning and statistical analysis for different purposes. For example, in one of our recently submitted papers [1], we developed two plug-ins: one to classify VNF resource demand characteristics, and another to calculate correlation among system KPIs and QoS of applications. NFV-LAB is based on Anaconda [12], an open source distribution of Python and R [13] programming languages specifically designed for data science and machine learning applications.

III. NFV-INSPECTOR DEMONSTRATION

To demonstrate analytical capabilities of NFV-Inspector, we designed a demo aiming to showcase classification of VNFs in an Open5GCore [14] installation that was deployed on top of an OpenStack setup. Open5GCore is a practical implementation of 4G/LTE network that mirrors pre-standard advancements on the mobile core network. During our demonstration we will showcase the followings:

- NFV-Inspector will perform a set of UE (user equipment) attachment, detachment and handover operations using Open5GCore benchmarking tool under different traffic loads and deployment policies. We will then classify VNFs resource demand characteristics and build a classification model from the extracted training set.
- We will show the correlation between UE attachment response time and different resource KPIs such as CPU and memory utilization, network bandwidth usage and disk I/O operations.

Fig. 2: NFV-Inspector Workflow

IV. CONCLUSION AND FUTURE WORK

We have currently developed major components of NFV-Inspector and published a pre-alpha release [15] under GPL v3 license. Beside finalizing NFV-Inspector's first stable release, we plan to extend NFV-Inspector and add more analytical plug-ins such as fault detection and neural network based VNF classification. We also plan to improve NFV-Inspector capabilities by adding more dynamic modes for on-demand resource optimization and make it compatible with reinforcement learning as well as supervised learning optimization approaches. In parallel, we aim to extend NFV-Inspector to support Kubernetes environments as well.

V. ACKNOWLEDGEMENT

This work was partially supported by the Grant No. 20160182 funded by the Knowledge Foundation of Sweden (KK-stiftelsen).

REFERENCES

- [1] M. Gokan, S. Bastani, J. Taheri, A. Kassler, and S. Deng, "Nfv-inspector: A systematic approach to profile and analyze virtual network functions," 2018, manuscript submitted.
- [2] Node.js. [Online]. Available: <https://www.nodejs.org/>
- [3] Python. [Online]. Available: <https://www.python.org>
- [4] collectd. [Online]. Available: <http://www.collectd.org>
- [5] Logstash. [Online]. Available: <https://github.com/elastic/logstash>
- [6] Elasticsearch. [Online]. Available: <https://github.com/elastic/elasticsearch>
- [7] Mariadb. [Online]. Available: <https://www.mariadb.org/>
- [8] Mysql. [Online]. Available: <https://dev.mysql.com/>
- [9] Ansible. [Online]. Available: <https://www.ansible.com/>
- [10] Openstack. [Online]. Available: <https://www.openstack.org/>
- [11] Kubernetes. [Online]. Available: <https://www.kubernetes.io/>
- [12] Anaconda. [Online]. Available: <https://www.anaconda.org>
- [13] R programming language. [Online]. Available: www.r-project.org
- [14] Open5Gcore. [Online]. Available: <https://www.open5gcore.org/>
- [15] Nfv-inspector source code. [Online]. Available: <http://source.nfvinspector.com/>