CLOUD INFRASTRUCTURE FOR RESEARCHERS BASING ON NFVMANAGEMENT AND ORCHESTRATION

V.Antonenko, R.Smeliansky, A.Ermilov, A.Romanov, N.Pinaeva, A.Plakunov

Moscow State University, 1Leninskiegory, Moscow, Russia

The paper presents C2 Platform that is suitable for establishment of the testbed environment for interdisciplinary research. The C2 architecture incorporates both the network service orchestration and service life-cycle management. C2 implementation follows ETSI NFV MANO standard and uses TOSCA to describe network functions and application functions. The paper provides experimental results of the platform's network traffic processing efficiency.

Keywords: SDN, NFV, Cloud

1. Introduction

Modern scientific research is impossible without the sophisticated computational and data-processing infrastructure. Different science domains present a variety of challenges to the cyber-infrastructure, which today may consist of desktop computers, small institutional clusters, cloud resources and supercomputers purpose-built to address specific problems.

In this paper, we explore a new approach to constructing such an infrastructure based on Software-Defined Infrastructure (SDI) technologies that combine performance with the required deep programmability, for example using Network Function Virtualization technology [1].

Network Function Virtualization (NFV) is a technology that supply required service functionality by software with commodity hardware. This approach separates the service functions from the hardware executing these functions by virtualization of computational, network and storage resources (ICT resources further).

In this paper, we will use abbreviations NFV as a name of the technology, VNF as virtual network function and VAF as virtual application function. The difference between a virtualized function and virtualized service will be explained later.

All Virtualized Network Functions could be separated into two main classes: network functions (VNF) and application functions (VAF). The VNFs are designed for network traffic processing: control and engineering.

The VAF is any data processing application in DC. VAF is not limited by the functionality of any legacy network equipment. For example, it can be a database management application, a web-server application or any other functions required by scientific fellows.

Different computational solutions may require differently optimized computational and data-processing architectures. For example, part of a given computational workflow may be executed using "map-reduce", followed by calculations in a tightly-coupled, massively parallel MPI environment. Recent advances in many-core systems present new infrastructure optimization points by allowing the use of Intel GPGPU [2] co-processors targeted at specific computational approaches.

A number of science domains are beginning to encounter a problem that is generically referred to as the "Big Data" problem, where the ability to generate the data by scientific instruments exceeds the ability of the computational elements to process them due to e.g. inadequate network bandwidth and/or insufficient computational power. Added to that are issues with long-term data storage and provenance.

Today's progress of science relies on collaborative efforts by multiple institutions and requires cyber resources belonging to multiple organizations. The multi-disciplinary nature of science requires the participation of experts from multiple domains. For example, bio-informatics brings together computer scientists and geneticists with the goal of designing efficient genotype processing algorithms.

In this paper, we will show there is no need to use several types of cloud platforms: one for VNF and one for VAF. The C2 Platform architecture will be presented and it will be demonstrated as this platform covers both VNF and VAF requirements. The C2 Platform architecture is based on the reference implementation ETSI NFV MANO model [1] and provides the full VNF life-cycle support: initialization, configuration, execution and deinitialization.

The life-cycle of a virtual function (VF, it doesn't matter VNF or VAF) is shown in Figure 1 and covers as monitoring as adjustment of the key indicators of the VF operation like performance and availability.

In this paper, for VF specification, we use TOSCA (Topology and Orchestration Specification for Cloud Applications) [3]. VF description in TOSCA is a list of specifications based on YAML language [4]. This description, called TOSCA-template, is all that needed to set a VF. TOSCA-template includes the structure of a cloud application, application management policies, OS image and scripts to start, to stop and to configure the application that implements the VF. The C2 platform assumes that a cloud administrator should provide the TOSCA-template as a zip or tar archive with a predetermined structure. There are more details in Section IV of the paper.

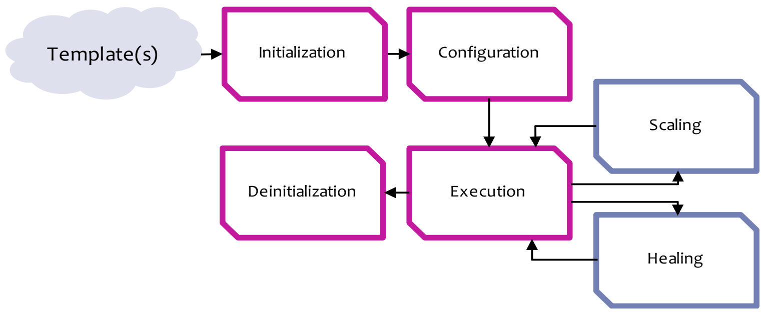

Figure 1. The VF life-cycle stages

The structure of the rest of the paper is as follows: Section II provides the survey of the existing NFV platforms and their suitability for scientific testbed platform use-case. Section III provides the basic C2 platform modules description and relations between them; the function registration and service management process. Section IV discusses the results of an experimental study of the C2 platform comprised of the analysis of dependency between the packet delay and the number of VF instances in network service as well as the analysis of network throughput for multiple customers sending data at the same time.

2. Related work

The ETSI NFV MANO document [1] is the reference model describing network function life-cycle, VNF orchestration and management. At the same time, there are no formal definitions of basic terms like network function and network service in this document. That gives us the opportunity to extend the approach of service life-cycle support described in [1] to VAFs too.

The architecture model has 3-layer structure: virtual infrastructure manager (VIM), VNF manager (VNFM) and NFV Orchestrator (NFVO). Today not all NFV platforms implement all these layers. For example, the OPNFV [5] project focuses only on the VIM functionality, the Cloudify [6] project - on NFVO, VNFM and uses external VIM as a basis like OpenStack [7], ESXi [8] and Azure [9].

The service life-cycle orchestration is one of the most important notions of the NFV MANO reference model. But today, NFV platforms do not support any automation of life-cycle stages. For example, the Cloudify project declares that it has the monitoring system with the capability for automatic function healing, but it's available only in the commercial version. Another example is the OpenStack Tacker [10]. Unlike the Cloudify project the Tacker project tries to implement MANO as an OpenStack internal module, but there is also no automatic management and orchestration of healing and scaling processes.

Today's NFV platforms have different visions on the VF description process. Some of them, like the Cloudify project, uses the TOSCA specification language for this purpose, others (Tacker) useits own specification language based on the YAML language with the intention to support TOSCA in the future.

In any implementation of an NFV platform, that intends to produce automated life-cycle support, there should be the monitoring tools. Usually, the developers use the third-party systems like Zabbix [11] or NAGIOS [12]. The system data monitoring lets an NFV platform identify VF instance state, e.g. "overloaded" or "unavailable", and apply actions to recover VF instance operation. The compatibility with the third-party monitoring systems API was previously announced by OPNFV, ManageIQ [13], OpenBaton [14].

It is important to mention, that no NFV platforms mentioned above consider the differences between VNFs and VAFs. We believe the reason for this is the lack of production-grade installations of such platforms. The only NFV platform project that clearly differentiates between VNFs and VAFs is the Central Office Re-architected as a Datacenter (CORD) [15]. CORD presents specially developed Telco Central Office architecture, and publish-subscribe model to produce three types of services: control plane services, data plane services and global cloud services. In our opinion, the main drawback of the CORD project is the lack of unification of these three types of services under one platform.

This brief survey shows that NFV platform development is the actual challenge and none of the mentioned above projects meet the needs of both VNFs and VAFs. There are no publications which demonstrate based on practical experience the capabilities of the platforms mentioned above.

3. C2 Platform architecture

The VF instance is a set of VMs with proper applications connected by isolated virtual networks. Under VF instance scalability we mean the horizontal one. In other words, we need the performance monitoring to launch additional instances when the performance goes down. The VF instance availability requires the healing monitoring that is to identify the moments the incorrect operation.

The C2 Platform is an extension of another project from our group called Self-Organized Cloud (SOC) [16] and research projects dedicated to the consistent orchestration problem [17] [18]. The C2 Platform is designed as a classic 3-layer MANO system (NFVO, VNFM, VIM). The C2 Platform architecture presented in Figure 2.

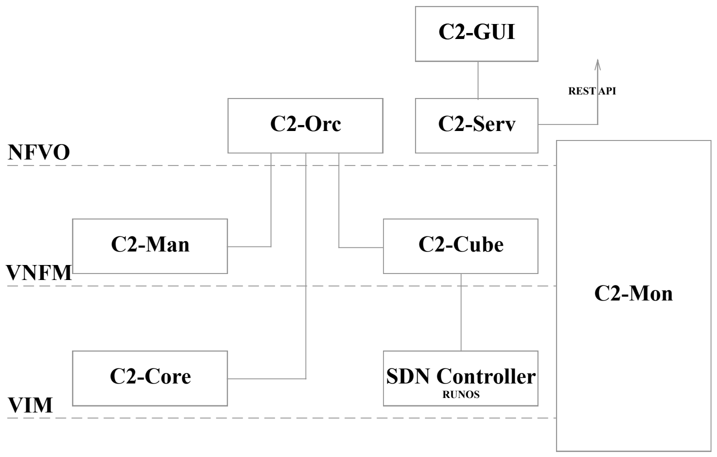

Figure 2. The C2 Platform Architecture

The keystone module is the NFV orchestrator (C2-Orc). It aims to support life-cycle of VF instances. The C2-Orc knows by what VF chain each service is represented. The C2-Orc knows nothing about VF instance description and details; it monitors and controls only VF instance state: normal, unavailable, overloaded etc.

When the C2-Orc needs to initialize new VF instance the one sends request to the VNF manager (C2-Man). It is expected that C2-Man in advance has got and parsed TOSCA-template of VF. The TOSCA specification assumes creating the template to represent the configuration of a complex application. The TOSCA template is based on integral, unified model covers various cloud applications. The templates describe the topology of application components at different layers including their structure, required resources and orchestration policies.

The TOSCA specification describes VNF in terms of "nodes" (subnet, a network, a server, or a software component) and "relationships" (nodes interconnections) [19].

The combination of TOSCA specification and VM OS image is called a C2-Template. The TOSCA specification includes TOSCA template and references to configuration scripts, binaries, configuration files.

The C2-Man gives to the C2-Orc info about VF details that is further redirected to the virtual infrastructure manager (C2-Core). The C2-Core is a module that operates with OpenStack API. The OpenStack, for now, is the only VIM that supported by the C2 Platform.

Usually, the functionality of OpenStack modules is not enough for life-cycle service orchestration. Especial concern is network management, for example, traffic mirroring, L2/L3 balancing. For that purposes, there is C2-Cube module, which with the help of SDN Controller RUNOS [20] extends the functionally of OpenStack network management (Neutron).

For monitoring of VF instance orchestration policy, the C2-Orc receives the data in real-time from the C2-Mon. The C2-Mon uses the third-party monitoring system Zabbix.

The C2 Platform has a C2-GUI to configure, deploy and monitor the VF instances, and to subscribe to services. The server-side of C2-GUI is the C2-Serv module. The C2-Serv provides two API types: the web socket API directly for the C2-GUI and the REST API for the C2 Platform.

4. Experimental Research

A. Chain Length Experiment

The experiment shows the time overheads caused by the length of VF chain. The experiment intended to find the dependency between VF chain length and a customer traffic delay. In the experiment, VF is a dummy virtual machine that just sends network traffic back into the network.

Figure 3 presents the results of this experiment – the dependence between a packet delay and a length of the VF chain. The length of chains shown on the X-axis, average delay length is on the Y-axis. The results demonstrate a linear dependence between the length of a VF chain and traffic delay. In certain case, each VF instance in the chain increases the delay less than ***1 ms***. Also, this overhead could be decreased by optimizing the VF's OS core in the field of network protocol stack for example by Intel DPDK [21].

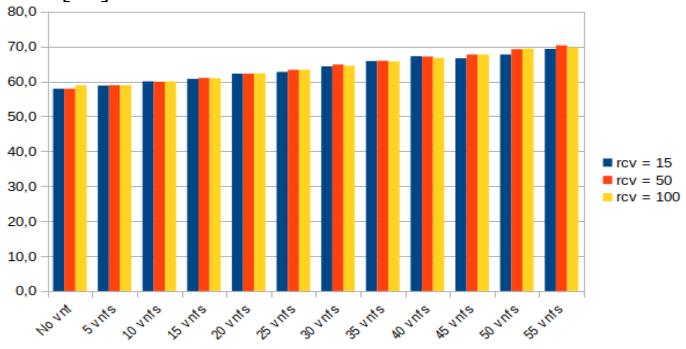

Figure 3. RTT dependence on the number of intermediate VFs

The results let us suppose that the C2 Platform can manipulate long VF chain with little influence on the customers' traffic.

B. Multi Customer Experiment

This experiment aims to test the C2 Platform works in multi customer mode, and check the network resources utilization, if all customers' traffic has an equal priority. We define the VF chain as a VAF which generates network traffic, NAT for the internet access and a firewall for access control for each customer that is placed on the testbed with 8 physical servers[1]. Let us call this VF chain as customer chain.

The purpose of the experiment is to measure network throughput of customer chains with as much chains as possible working simultaneously. This simulates multiple customers sending experiment data into outside network. The measurement is done by "iperf" program.

We create 111 customers' chains in the experiment. The number of chains depended on available physical resources (about 300 KVM VMs). And install a special agent that observes how chains will utilize the 1-Gbit link between cloud infrastructure and external network. The observed results are presented in Figure 4.

As we can see the average customer chain bandwidth is ***8.99 Mbps***, it means that all customers shared external link bandwidth evenly. The total bandwidth is ***989.92 Mbps***, which means that network utilization is close to ***1 Gbps*** maximum.

[1]Intel Xeon CPU E5-2640 v2 @ 2.00GHz, 64 GB RAM, 6.4 TB HDD.

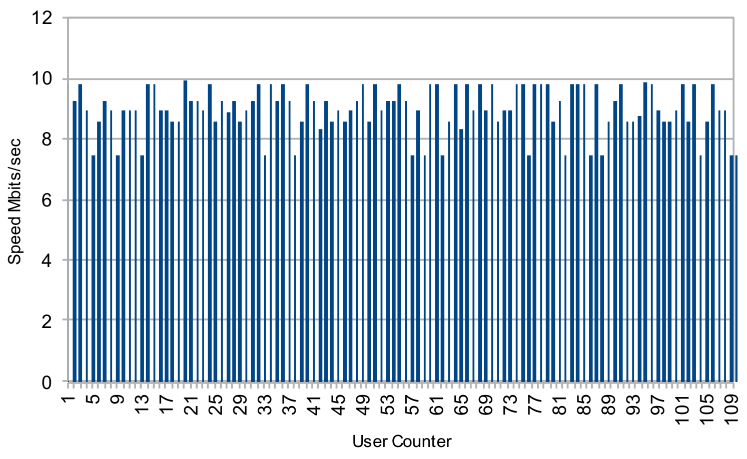

Figure 4. External link bandwidth destitution among customers

5. Conclusion

The paper demonstrates the possibility to bring the benefits of the NFV platform into a scientific testbed cloud. It explains what platform functionality is critical for this.

The survey of similar projects shows that none of the existing cloud platforms satisfies the requirements of a scientific testbed cloud. On top of this C2 platform provides VAF functionality described by the same TOSCA template language.

The experiments prove that platform with such a versatility does not have significant overhead, can work with the multiple customers, and operate with VF chains of complicated structure.

References

[1] *ETSI Network Functions Virtualisation — Introductory White Paper" 22 October 2012. Retrieved 20 February 2017.*
[2] *H. Kim, R. Vuduc, S. Baghsorkhi "Performance Analysis and Tuning for General Purpose Graphics Processing Units (GPGPU)." — Morgan & Claypool Publishers, 2012.*
[3] *OASIS. TOSCA Simple Profile in YAML Version 1.0*
[4] *YAML. Retrieved 7 October, 2017: http://www.yaml.org/spec/1.2/spec.html*
[5] *Linux Foundation, "OPNFV - An open platform to accelerate NFV". October 2014.*
[6] *Cloudify. Retrieved 7 October, 2017: http://getcloudify.org*
[7] *OpenStack, Manual, November 2011.*
[8] *C. Waldspurger: Memory Resource Management in VMware ESX Server. Operating Systems Design and Implementation (OSDI), (2002)*
[9] *Microsoft. Azure. Retrieved 7 October, 2017: https://azure.microsoft.com*
[10] *OpenStack Tacker. Retrieved 7 October, 2017: https://wiki.openstack.org/wiki/Tacker*
[11] *Zabbix Manual. Retrieved 7 October, 2017: http://www.zabbix.com/downloads/ZABBIX%20Manual%20v1.6.pdf*
[12] *Nagios. Retrieved 7 October, 2017: https://www.nagios.org/about/overview*
[13] *ManageIQ. Retrieved 7 October, 2017: http://manageiq.org/docs*
[14] *Open Baton. Retrieved 7 October, 2017: https://openbaton.github.io*
[15] *CORD Whitepaper. Retrieved 7 October, 2017: http://opencord.org/wp-content/uploads/2016/03/CORD-Whitepaper.pdf*
[16] *Kostenko, V., Plakunov, A., Nikolaev, A., Tabolin, V., Smeliansky, R., and Shakhova, M. Selforganizing cloud platform. In Proceedings of the 2014 international science and technology conference "Modern Networking Technologies (MoNeTec)" (2014), SDN@NFV modern networking technologies, Moscow, Russia*

[17] *V. A. Kostenko and A. V. Plakunov, "Ant algorithms for scheduling computations in data processing centers," Moscow University Computational Mathematics and Cybernetics, vol. 41, no. 1, pp. 44–50, 2017.*

[18] *P. M. Vdovin, I. A. Zotov, V. A. Kostenko, A. V. Plakunov, and R. L. Smelyansky, "Comparing various approaches to resource allocating in data centers," Journal of Computer and Systems Sciences International, vol. 53, no. 5, pp. 689–701, 2014.*

[19] *OASIS. Topology and Orchestration Specification for Cloud Applications (TOSCA) TC Retrieved 7 October, 2017: http://docs.oasis-open.org/tosca/TOSCA/v1.0/TOSCA-v1.0.html*

[20] A. Shalimov, S. Nizovtsev, D. Morkovnik and R. Smeliansky, "The RunosOpenFlow Controller", *2015 Fourth European Workshop on Software Defined Networks*, 2015.

[21] *Intel Corporation, Intel Data Plane Development Kit (Intel DPDK) Programmer's Guide, August 2013.*