ISSN: 2526-1371

Performance Analysis of SDN Virtualization

Davi Daniel Gemmer[1], **Augusto Foronda**[1]

[1]Department of Computer Science
Federal University of Technology - Paraná (UTFPR)
Post-Graduation Program in Computer Science

gemmer@alunos.utfpr.edu.br, foronda@utfpr.edu.br

Abstract. *Software Defined Network (SDN) is an emerging networking paradigm to overcame the limitations of a traditional network infrastructure. SDN can improve the monitoring, management, security and traffic engineering of a network. A solution for SDN architecture is through Virtual Machine (VM) and container. However, there is a lack of performance comparison between VM and container with SDN. This paper analyzes the virtualization of SDN using VM and Docker container in terms of performance. Simulations were done with OpenDaylight SDN controller and an evaluation was done to understand the efficiency and the scalability of the network. The results show that compared with virtual machines, containers can scale a larger number of flows. This paper can help the network designers to make a better SDN architecture in terms of which virtualization method should be used.*

Keywords – SDN, Virtual Machine, container, performance.

1. Introduction

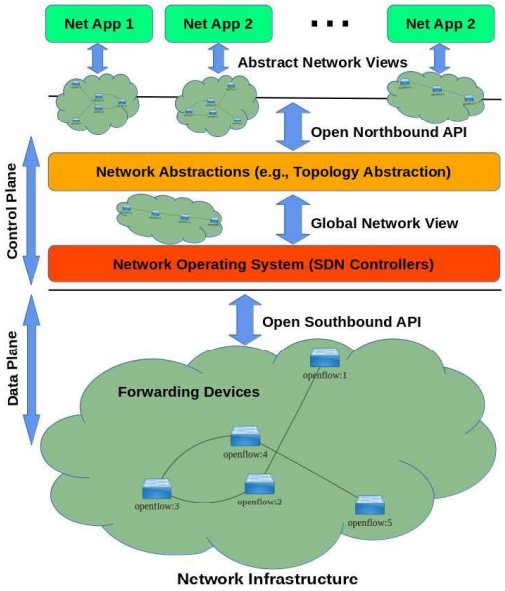
Traditional network technology has inherent problems of rigid structure and complex configuration and cannot meet the requirement of network innovation which demands dynamic and flexible management. Software Defined Network (SDN) is proposed to overcame the problems of a traditional network and presents a new idea with three layers: data plane, control plane and application plane. Data plane is composed by network devices and forwards packets according to a decision made by the control plane, which acts as a mediator for the data plane and the application plane and handles the traffic flow in the network. Application plane achieves customized application, such as network automation, network management and network security [Joy 2015]. SDN architecture involves the usage of virtualization technologies such as virtual machines (VMs) and containers. VMs are extensively used as they permit workloads to be isolated from each other and for the resources to be well controlled. However, containers have the great advantage of allowing applications to run separately from the host infrastructure, such as Docker container. Some papers have shown that Docker container can offer superior performance compared to VM [Felter et al. 2015]. This paper presents an SDN controller built on VMs and Docker containers. We perform experiments to evaluate the memory and number of flows.

This paper is organized as follows: Section II discusses the SDN architecture. In Section III, we present a comparison between VM and container. Section IV describes the related work. Section V present experiments and give the results. Finally, Section VI presents the conclusion of our work.

2. SDN Architecture

Figure 1 shows the SDN architecture. There are 3 layers: data layer, control layer and application layer. Data layer can communicate with control layer through Southbound interface and control layer can communicate with application layer through Northbound interface. Data layer is composed of networking equipment's which forms the network infrastructure to forward, modify or drop network traffic according to the instructions given by controller. Control Layer is composed by the SDN controller, which is a logical entity that receives instructions from application layer and relays them to the networking equipment's. SDN controller manages the network and it has logic control for switching, routing, firewall security rules, etc. The SDN controller also extracts information about the network and send it back to the application layer such as statistics and events about the traffic. Application layer is composed by programs that controls a set of resources of one or more SDN controllers via application programming interfaces (APIs) [Joy 2015]. Southbound interfaces are APIs that enables the communication between control layer and data layer. OpenFlow is the first and probably most well-known southbound interface. However, it is not the only one available. Northbound interfaces enable the communication between control layer and application layer and it is configured through REST APIs of SDN controllers, which are used to facilitate automation of the network to align with the needs of different applications with SDN network programmable [Felter et al. 2015].

Figure 1. SDN Architecture



Adapted from [Kreutz et al. 2014]

One of the most famous SDN controllers is OpenDayLight (ODL), which provides a centralized management system that allows to have a programmable network. ODL controller can be used as a platform for configuring different aspects of the network and solving different network challenges. ODL uses open source integration standards and APIs to make the network more programmable, intelligent and adaptable [Bhimani et al. 2017].

3. Container vs Virtual Machine (VM)

System virtualization separates the underlying physical device and the upper operating system. A single physical machine can be divided in multiple machines to maximize the resource utilization and flexibility [Peng et al. 2009].

There are some differences between these two technologies: 1) A container share the same OS (operating system) on the same machine with other containers and VMs do not share the OS. Therefore, container is more lightweight than a VM; 2) VM needs an hypervisor to translate an instruction that can be executed by the host because a VM runs in a non-privileged mode. On the contrary. A container does require an extra layer because it communicates with the OS through the system calls; 3) Each VM has its own image file, while different containers may share some of their images. These differences shows some advantages of using containers: the size of a container is smaller than the size of a VM and a container usually takes less hardware resources since it does not need to maintain an OS [Bedhief et al. 2016].

A container packs together all the necessary software that the application needs to run with, such as the libraries and the other dependencies. There are some particular features that enables different containers to run on the same physical machine, such as Cgroups and namespaces [Morabito 2016]. Cgroup allows system administrators to allocate resources such as CPU, memory, network to the running containers, which can be adjusted dynamically. However, it can not use more resources than specified in the cgroups. Namespace guarantees that processes running in different containers will not conflict with each other because it provides IDs, network interfaces and host names for each container [Bedhief et al. 2016].

4. RELATED WORK

Many research efforts have been made to explore the advantages of containers as well as to compare the containers with the VMs. Some papers have investigated the container performance. It was showed that a container running in a Raspberry Pi 2 has almost the same performance compared to native execution [Morabito 2016]. Miguel et al demonstrated that a container technology achieves a very low overhead compared to native setups. Other papers have compared VMs and containers. A comparison has been made between Linux containers and virtual machines in terms of the performance and extensibility [Joy 2015]. Performance analysis has been made of virtual machine (VM) application and deployments and compares them with the Linux containers [Felter et al. 2015]. Janki et al. [Bhimani et al. 2017] compares the performance of Spark jobs running in a container cluster and a fix-sized VM cluster with one physical machine. Wes et al. [Felter et al. 2015] showed a better container performance than VM in terms of disk and network I/O. And some papers have demonstrated solutions with SDN and container. IoT networks and applications are extremely complex and an SDN architecture with Docker container has been proposed to manage such networks. The experiments proved the feasibility of the proposed architecture and the communication established between smart devices through an SDN-based network [Bedhief et al. 2016]. Y. Xu et al proposed an SDN-based autodocker framework integrated within the switches for enabling auto-docking/-undocking of applications at the edge switches. In this manner, the Docker framework automatically and effectively manages the storage, computing, and

networking resources of the switch [Xu et al. 2016]. VMs and containers have been used in some architectures, such as radio access network (RAN). A. Gopalasingham et al compared their performance and analyzed the two different architectures. Measurements and comparisons confirmed that the Docker container based architecture could provide a superior performance compared to the VM based architecture [Xingtao et al. 2016].

5. PERFORMANCE EVALUATION

Our evaluation considers OpenDayLight SDN controller. The performance metrics considered are: memory use and average flows. The main goal is to investigate which virtualization method gives the best result with these metrics. The evaluation is carried out using WCBench [Farrell 2019], a performance measurement tool to benchmark SDN controllers.

5.1. Test Environment

Both WCBench and ODL controller were implemented on the same machine IBM X3850 M2 with (4x Intel® Xeon™ Processor X7350 CPU @ 2.93GHZ (4cores)), 64 GB of memory was available. The system was running CentOS 7 `x86_64` (1810) and the virtual machine was virtualized through the ESXi 6.0.0 Update 3. Docker was running with centos 7 container image, for tests the OpenDaylight Oxygen 0.8.4 controller was used.

5.2. Methodology

WCBench was used to emulate 128 switches and the tests were configured to run a loop of 2000 cycles with a duration of 4 minutes per cycle. For the execution of the tests, a total of 16 virtual processors and 60 GB of RAM were allocated for both virtualization technologies. The table 1 brings the settings used in WCBench.

Table 1. Parameters for WCBench test and ODL config

NUM SWITCHES	128
NUM MACS	100000
TESTS PER SWITCH	10
MS PER TEST	10000
CBENCH WARMUP	1
KARAF SHELL PORT	8101
CONTROLLER	OpenDaylight
CONTROLLER IP	localhost
CONTROLLER PORT	6633
SSH HOSTNAME	cbench

Adapted from [Farrell 2019]

5.3. Average RAM usage

When comparing the results between the two virtualization technologies, it was noticed that the Docker container had a slightly higher RAM utilization when compared to the virtual machine, with a difference of 131 MB more for the same experiment as can be seen in the figure 2 , where the virtual machine used an average of 2224 MB of RAM and the Docker container had an average of 2355 MB of RAM. WCBench analyzes the RAM used throughout the operating system, which has caused a problem due to the need to share host resources for OpenvSwicth (OVS) operation within the Docker container.

Consequently the analysis considered the use of host memory during testing while the hypervisor isolated the VM.

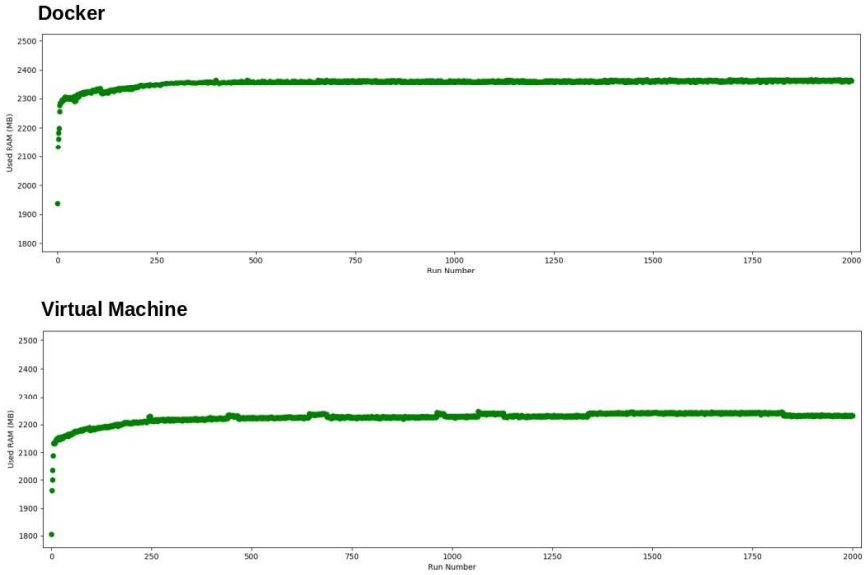


Figure 2. Used RAM

5.4. Average Flows per Second

When analyzing the average of transmitted flows, the container Docker obtained an average of 5,287 flows per second where the virtual machine obtained an average of 4,703 flows per second as can be seen in the figure 3. In the same test the container Docker obtained a difference of 584 more flows than the virtual machine.

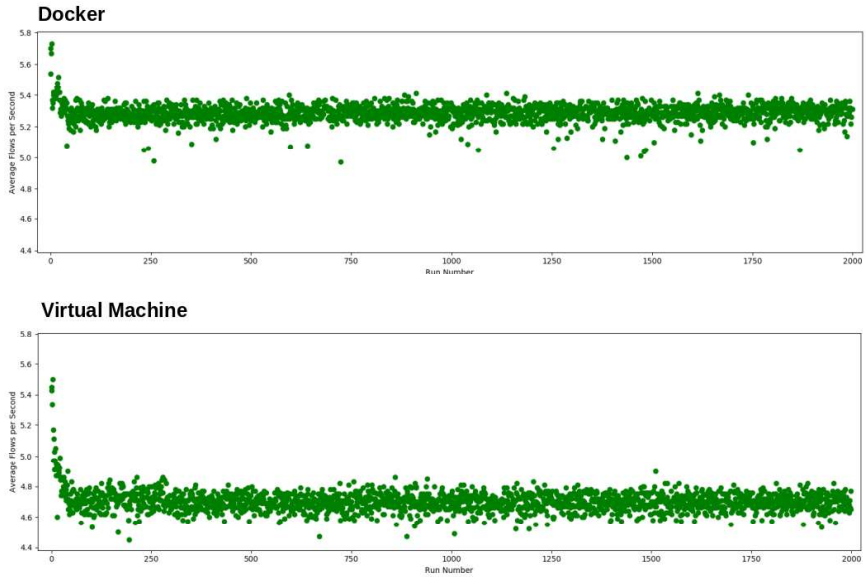


Figure 3. Average Flows

6. CONCLUSION

This paper presents a comparison performance evaluation of ODL SDN controller using VM and Docker container. From the performance evaluation, Docker container exhibited the best throughput results showing that it is able to respond to requests more promptly under traffic loads. This work provide users with guidelines towards which virtualization method provides a better performance. The use of RAM was stable and did not obtain much difference in the two cases but the small advantage of VM led us to believe it was due to the way WCBench performs RAM analysis.

References

Bedhief, I., Kassar, M., and Aguili, T. (2016). Sdn-based architecture challenging the iot heterogeneity. In *2016 3rd Smart Cloud Networks & Systems (SCNS)*, pages 1–3. IEEE.

Bhimani, J., Yang, Z., Leeser, M., and Mi, N. (2017). Accelerating big data applications using lightweight virtualization framework on enterprise cloud. In *2017 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–7. IEEE.

Farrell, D. (2019). Wcbench. *Available in:* `https://github.com/dfarrell07/wcbench`. Access in June 2019.

Felter, W., Ferreira, A., Rajamony, R., and Rubio, J. (2015). An updated performance comparison of virtual machines and linux containers. In *2015 IEEE international symposium on performance analysis of systems and software (ISPASS)*, pages 171–172. IEEE.

Joy, A. M. (2015). Performance comparison between linux containers and virtual machines. In *2015 International Conference on Advances in Computer Engineering and Applications*, pages 342–346. IEEE.

Kreutz, D., Ramos, F., Verissimo, P., Rothenberg, C. E., Azodolmolky, S., and Uhlig, S. (2014). Software-defined networking: A comprehensive survey. *arXiv preprint arXiv:1406.0440*.

Morabito, R. (2016). A performance evaluation of container technologies on internet of things devices. In *2016 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 999–1000. IEEE.

Peng, J., Zhang, X., Lei, Z., Zhang, B., Zhang, W., and Li, Q. (2009). Comparison of several cloud computing platforms. In *2009 Second international symposium on information science and engineering*, pages 23–27. IEEE.

Xingtao, L., Yantao, G., Wei, W., Sanyou, Z., and Jiliang, L. (2016). Network virtualization by using software-defined networking controller based docker. In *2016 IEEE Information Technology, Networking, Electronic and Automation Control Conference*, pages 1112–1115. IEEE.

Xu, Y., Mahendran, V., and Radhakrishnan, S. (2016). Sdn docker: Enabling application auto-docking/undocking in edge switch. In *2016 IEEE conference on computer communications workshops (INFOCOM WKSHPS)*, pages 864–869. IEEE.