TECHNIUM

Technium Vol. 8, pp.25-31 (2023)
ISSN: 2668-778X
www.techniumscience.com

A Comparative Evaluation of the Performance of SDN Controllers (ONOS) using DOCKER Container

Ghazwan Abdullah Mohammed , Omar Abdulmonem Ibrahim Aldabbagh

Computer Science Department
University of Mosul

Abstract. software-defined networks SDN represents a significant shift in the networking field through the ability to solve problems facing traditional networks, such as updating components for networks and integration between devices. Open-source components and protocols were used in SDN. Among its advantages are the separation of the control plane from the data plane, the use of the OpenFlow protocol, and the transition to Decentralization in network management. In SDN, there are several types of controllers. In this paper, work has been done on ONOS controllers in cluster mode from one to five controllers, and a comparison between the performance of different numbers of controllers to obtain the best performance through the use of Cbench. It was found that using one or two controllers achieves the best performance

Keywords. SDN, ONOS, here.

1. Introduction

Software Defined Networking (SDN) is a promising approach to networking because it separates the control plane from the data plane. It provides centralized intelligence to increase the adaptability and effectiveness of the network. Control Plane is considered the brain of the SDN network architecture as it contains the centralized controller, which manages and controls the network devices. the control plane provides the router and the settings that will Use by Data Plane. It performs the necessary actions to forward the packets. The packets have been checked using the header provided by the data plane within the packets. Can change the network-wide infrastructure with a centralized console Through SDN. It means converting the hardware-intensive traditional network to a fully virtualized and programmable network which fulfils the need for scalability, agility, and visibility of the networks. Attention to reliability and performance improvement is one of the basics of SDN work. This paper is concerned with evaluating the performance of one of the common controllers (ONOS) to obtain the optimal number of controllers for the best performance in the form of a cluster using the docker container. ONOS stands for Open Network Operating System. ONOS supplies the control plane for a software-defined network (SDN), managing all network components. Such as links and switches running software to supply connecting services to end hosts and contiguous networks. Running ONOS as a cluster on a docker container, many target machines can run together as an integrated, cohesive distributed system configured as a cluster. A cluster is an excellent way to eliminate single-controller failures where these controllers work together in one connected entity. A Docker container image is a light standalone executable pack of software that contains everything needed to run an application: code, runtime, system tools, system libraries, and other settings.

2. Background of SDN

SDN represents the best way to create networks based on open-source devices and technologies with traditional networking devices. One of the main problems in traditional networks, which is manual processing, has been solved by SDN. Also, one of the main advantages of SDN is separating the control plane from the data plane.[1][2] Wide-ranging user requirements include delay, Throughput, availability, and bandwidth. Were challenging to define or articulate. The SDN

design was introduced to address the drawback of this conventional networking system, where the control plane is conceptually centralized and separated from the data plane.[3] Software-defined networks have three levels: control plan, data plan, and application. The data plan is controlled through a specific interface through SDN. Plane data is the layer responsible for data flow, and the control plane is responsible for this process. In addition, the control plane layer transfers the data plane operations to the application.[4] fig 1.

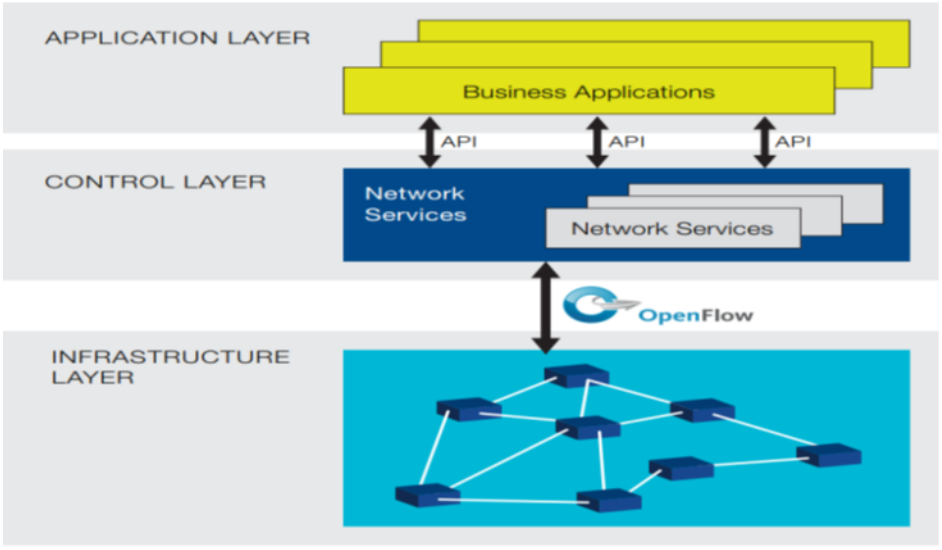


Fig. 1: SDN architecture[4]

3. Related work

Many papers discuss the comparison of the performance of different types of controllers. However, the papers discuss the comparison of the performance of several controllers of the same type is very few. In addition, the papers that study the comparison using the docker container are not very common. In [5], the researcher made a comparison between the performance of five controllers (Ryu, OpenDaylight, ONOS, POX, and libfluid) using a network simulator (mininet) with different numbers of switches. On the network to see how far the controllers stop responding. In [6], the researcher made a benchmark comparison between two well-known controllers (OpenDaylight.Floodlight). Performance was compared between the used controllers through Throughput and latency using Cbench.In [7], the performance of several of the most famous controllers (Open, OpenDaylight Network Operating System (ONOS), Ryu, and POX) was compared using multiple performance tests. The first is to connect each controller with a linear topology and a different and increasing number of switches. The second is to connect a different number of controllers to a different number of switches. Fixed switches and performance are measured from throughput results and RTT and jitter between end-hosts

4. Materials and Method

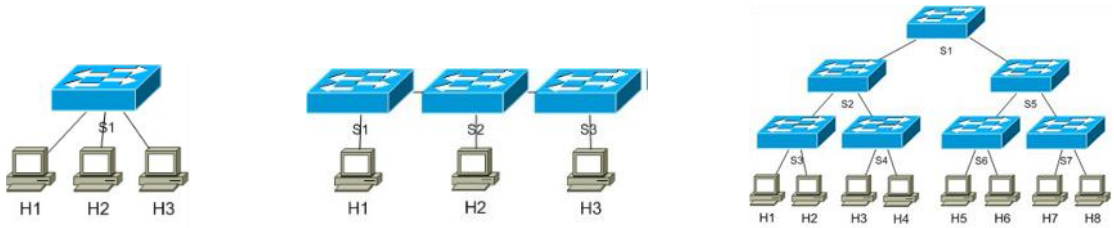
4.1 Methodology

many controllers can be used, but the onos controller was chosen because it has many features to improve performance, and the basis for doing this research is to get the best performance. The onos controller is run in cluster mode, and the number of controllers used in this cluster is from one to five. Controllers are gradually increasing, and the difference in performance is noticed. Implementation is done using the docker container. A container is created for each controller, and three other containers called Atomix are created to be used as databases for the network. If one

fails, the other completes the work to increase the network's reliability. As for the network, a mininet emulator was used, and the network was used tools to compare performance. Cbench to get the Throughput and latency of the controller . Host running on windows 10 and guest on Linux Ubuntu 16.04.03 LTS running on an Intel(R) Core(TM) i5-6440HQ CPU @ 2.60GHz 2.59 GHz with 16 GB of DDR4 RAM

4.2 Mininet

mininet is a network simulator used by researchers and developers working on virtual networks. Developed by the mininet team, this emulator creates an integrated virtual network consisting of a console, switches, and host hardware on a single machine.[5] Hosts on Mininet can run basic Linux commands and file systems. Mininet has three different topologies: linear, tree, and single topology. The linear topology consists of several switches on one level. Tree topology consists of a group of switches divided into several levels. A single topology consists of a single switch and a group of hosts. As shown in Fig 2.



(a) Single topology (b) Linear topology (c) Tree topology

Fig. 2: Three common topologies in Mininet.

4.3 ONOS controller

ONOS is an operating system developed to meet the needs of developers and operators who want to build solutions for companies and achieve high flexibility to create dynamic networks with simple software interfaces. One of the most important features of ONOS is that it supports real-time configuration and control of the network and that it does not need to run routing control protocols. Also, build SDN networks that are highly available, scalable, and high-performance. The source code of ONOS is written in Java, and a new version is released every three months. ONOS allows service providers to expand and add new components to the network without disturbing the rest of the system. The distributed architecture reduces the possibility of network failure, which gives high availability to the network[6]

4.4 DOCKER Container

Software developers are using Docker technology more frequently in a variety of applications.[7] DOCKER is a tool that facilitates the process of encapsulating software and deploying it on any environment at scale while simplifying workflows.[8] DOCKER is usually seen as a virtualization platform, but it is much more than that. DOCKER's work spans industries with different technologies such as fabric, Xen, OpenStack, and more. DOCKER represents a competitor to the most well-known technologies of the past decade. One of the most important features offered by DOCKER is a process of combining two things, the first is the ease of application publishing tools such as fabric and Capistrano, and the second is the ease of controlling virtual systems. Given the large and wide use of DOCKER in the industrial sectors, the information technology field has many innovative technologies used in automation and overcoming complexities. The main goal of virtualization is to improve the information technology infrastructure.[9] Docker images produce containers. Each component is necessary

for running the application in a constrained manner. The container is supposed to hold the application. The service needed can be used to construct the container images for the software or application. Let us say that a program that uses the Ubuntu operating system and the ONOS server Inserts the Docker file as an addition.[10] Using the "docker run" command, a container with an image of the Ubuntu OS made up of the ONOS server is built and activated. [11]

4.5 Cbench

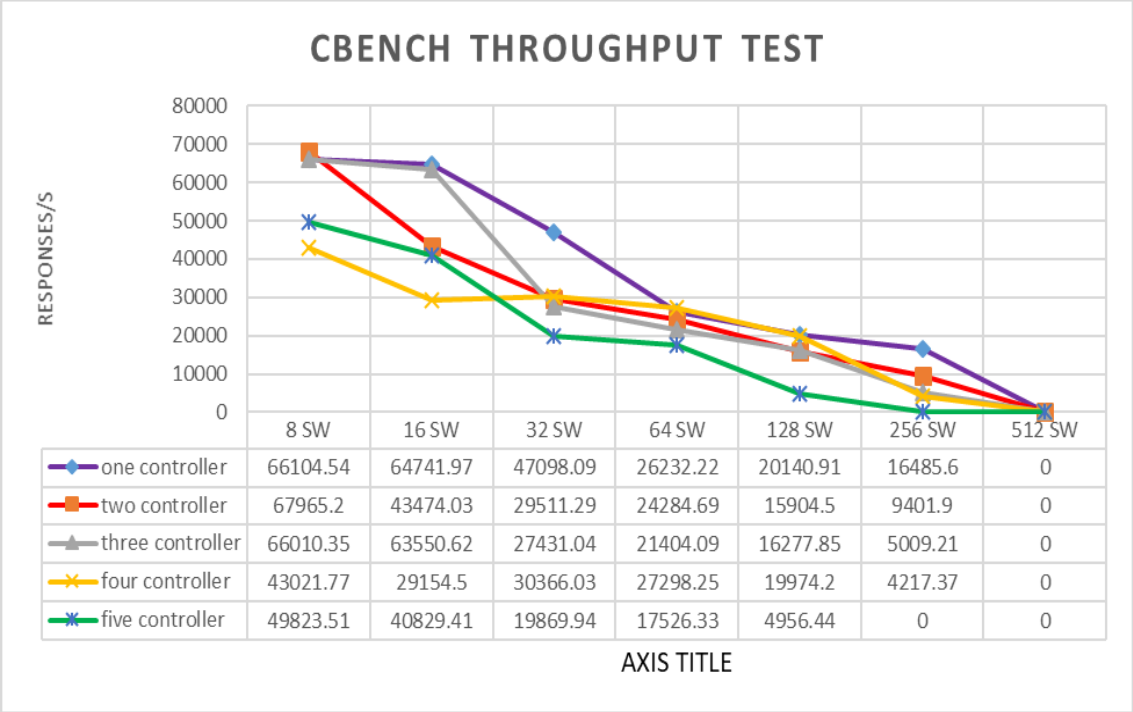
Cbench is a tool for evaluating OpenFlow SDN controller performance.[12] By sending a packet in a message and keeping an eye out for flow mod messages to establish a connection with a controller, Cbench simulates a customizable number of switches. Each switch is connected to a certain number of virtual hosts, and these hosts generate traffic to communicate with one another.[13] As a result, every new flow creates numerous requests for data paths. Cbench waits for the controller to react after receiving the requests before counting each installed response that happened. It operates in throughput and latency modes, respectively. Cbench measures how long it took the controller to respond to data path requests in the first mode, and in the second mode, it counts how many data route requests the controller satisfies each second.[14]

5. Experimental Result

5.1 Cbench Results

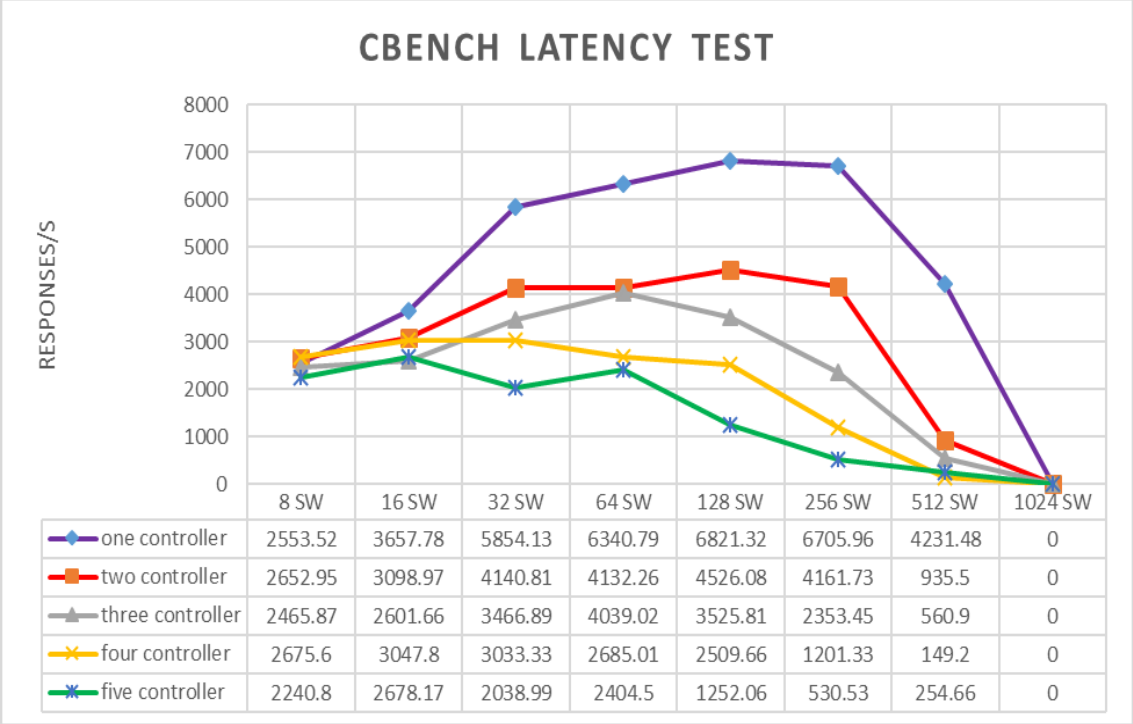
Cbench is a measure of the efficiency of the controller. This measure performs two functions, the first is Throughput and the second is latency. Test results showed the best results obtained when eight switches were used. Results decreased as the number of switches increased because many switches required more processing activities, increasing the network load. In general, the single controller achieved the best results, as shown in Fig 2. When executing the cbench in latency mode, we notice that the results gradually increase with the increase in the number of switches, as eight switches were used in the first test. After the number doubled, the results were as high as 128 switches. The results appeared this way because the latency depends on the number of responses per second when using eight switches, whose responses are less than when using 128 switches. However, after the 128th, the results begin to decline due to pregnancy. In general, the one controller and the two controller achieved the best results, as shown in Fig 3.

Technium Vol. 8, pp.25-31 (2023)
ISSN: 2668-778X
www.techniumscience.com



	8 SW	16 SW	32 SW	64 SW	128 SW	256 SW	512 SW
one controller	66104.54	64741.97	47098.09	26232.22	20140.91	16485.6	0
two controller	67965.2	43474.03	29511.29	24284.69	15904.5	9401.9	0
three controller	66010.35	63550.62	27431.04	21404.09	16277.85	5009.21	0
four controller	43021.77	29154.5	30366.03	27298.25	19974.2	4217.37	0
five controller	49823.51	40829.41	19869.94	17526.33	4956.44	0	0

Fig .2



	8 SW	16 SW	32 SW	64 SW	128 SW	256 SW	512 SW	1024 SW
one controller	2553.52	3657.78	5854.13	6340.79	6821.32	6705.96	4231.48	0
two controller	2652.95	3098.97	4140.81	4132.26	4526.08	4161.73	935.5	0
three controller	2465.87	2601.66	3466.89	4039.02	3525.81	2353.45	560.9	0
four controller	2675.6	3047.8	3033.33	2685.01	2509.66	1201.33	149.2	0
five controller	2240.8	2678.17	2038.99	2404.5	1252.06	530.53	254.66	0

Fig .3

Conclusion

In this paper, the ONOS controller was chosen in cluster mode because the main function of this controller is to improve performance. The DOCKER container was used to run the controllers through the mininet network simulator. In general, the tests showed a decline in the results when increasing the number of switches, especially when using more than 16 switches. The difference in the results is significant. Several tests were conducted, such as Cbench with its throughput and latency modes Using one controller and two controllers achieved better results than the rest and using five controllers achieved the worst results. Moreover, using the DOCKER container in operating the controllers saved the consumption of physical resources. However, these tests are not sufficient to obtain the optimal number of controllers because some network properties require a larger number of controllers, such as reliability, but this will negatively affect performance

REFERENCES

[1] A. Nayyar, B. Singla, And, and P. Nagrath, *Software Defined Networks*. Scrivener Publishing, 2022.

[2] A. Nguyen-Ngoc, S. Raffeck, S. Lange, S. Geissler, T. Zinner, and P. Tran-Gia, "Benchmarking the ONOS controller with OFCProbe," in *2018 IEEE Seventh International Conference on Communications and Electronics (ICCE)*, 2018, pp. 367–372.

[3] R. F. Babiceanu and R. Seker, "Manufacturing cyber-physical systems enabled by complex event processing and big data environments: a framework for development," in *Service Orientation in Holonic and Multi-agent Manufacturing*, Springer, 2015, pp. 165–173.

[4] K. Nam and K. Kim, "A Study on SDN security enhancement using open source IDS/IPS Suricata," *9th Int. Conf. Inf. Commun. Technol. Converg. ICT Converg. Powered by Smart Intell. ICTC 2018*, pp. 1124–1126, 2018, doi: 10.1109/ICTC.2018.8539455.

[5] N. Gupta, M. S. Maashi, S. Tanwar, S. Badotra, M. Aljebreen, and S. Bharany, "A Comparative Study of Software Defined Networking Controllers Using Mininet," 2022.

[6] "https://wiki.onosproject.org/".

[7] M. U. Haque, L. H. Iwaya, and M. A. Babar, "Challenges in docker development: A large-scale study using stack overflow," in *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2020, pp. 1–11.

[8] A. M. Potdar, D. G. Narayan, S. Kengond, and M. M. Mulla, "Performance evaluation of docker container and virtual machine," *Procedia Comput. Sci.*, vol. 171, pp. 1419–1428, 2020.

[9] P. Raj, J. S. Chelladhurai, and V. Singh, *Learning Docker*. Packt Publishing Ltd, 2015.

[10] D. Gedia and L. Perigo, "Performance Evaluation of SDN-VNF in Virtual Machine and Container," *2018 IEEE Conf. Netw. Funct. Virtualization Softw. Defin. Networks, NFV-SDN 2018*, pp. 1–7, 2018, doi: 10.1109/NFV-SDN.2018.8725805.

[11] S. Kwon and J.-H. Lee, "Divds: Docker image vulnerability diagnostic system,"

IEEE Access, vol. 8, pp. 42666–42673, 2020.

[12] A. T. Albu-Salih, "Performance Evaluation of Ryu Controller in Software Defined Networks," *J. Al-Qadisiyah Comput. Sci. Math.*, vol. 14, no. 1, p. Page-1, 2022.

[13] A. Haggag, S. Awad, and A. S. Gaballah, "Controllers Performance Analysis in Software-Defined Networking," *Int. J. Sci. Res. Comput. Sci. Eng. Vol*, vol. 10, no. 3, 2022.

[14] O. Adeniyi, D. Chauhan, U. Ezidonye, M. Ayeni, and V. Patel, "Software-defined networking security," 2021.