Open-Cloud Computing Platform Design based on Virtually Dedicated Network and Container Interface

Ki-Hyeon Kim, Dongkyun Kim, Yong-Hwan Kim

Abstract Background/Objectives: ***Companies that perform services based on the data center use server virtualization technology to reduce costs. Most companies use hypervisor-based server virtualization technologies, which require hardware virtualization to handle many I/O and resources at the kernel level.***

Methods/Statistical analysis: ***hypervisor-based technology is limited by low speed, which can be resolved by using container-based virtualization technology. However, container-based network also has issues. It is difficult to construct flexible container-based network, and data transmission performance declines if the existing container network interface is used.***

Findings: ***To solve these problems, this paper uses SDN Technology and Kubernetes, which is used as a container orchestration system. Furthermore, network configuration is linked with CNI of Kubernetes and virtually dedicated network (VDN) application based on the Open Network Operating System (ONOS) Controller for SDN. In other words, we have developed a new CNI by deleting the overlay network and a virtual bridge used in the existing Kubernetes. The newly developed CNIs are directly connected to the host network for high-performance network operations.***

Improvements/Applications: ***The Interconnection of VDN and Kubernetes provides high performance and flexible networking capability. We compare the network performance of the existing CNIs and the newly developed CNI in this paper, which describes the superiority of the new CNI in terms of the measured network throughput results.***

Keywords: KREONET-S, VDN, Kubernetes, Science DMZ, SD-WAN

I. INTRODUCTION

Many companies with data centers such as Amazon, Google, Naver and Daum-Kakao perform services based on data centers. These companies strive to reduce the cost of data center operations. Servers run at very low average utilization levels (less than 15%). Virtualization software increases utilization typically by fourfold or more, which means for a given workload that can be virtualized, a company can typically reduce the number of physical servers by fourfold [1]. In order to reduce the costs in terms of data centers, it is necessary to reduce the waste of server resources. Companies use server virtualization technology, which logically divides server resources and enables various services, to reduce such waste. Server virtualization technology can be divided into hypervisor and container technologies.

In hypervisor-based technology, OS can be freely installed, while the kernel layer should process a considerable amount of I/O as it uses hardware virtualization. Accordingly, it results in a system that is heavy and slow when performing a service [2]. Container-based virtualization technology appeared to resolve this issue. A container is a virtualization technology that enables light and swift operation by installing additional OS to separate the process for improved virtualization. A container-based open source virtualization platform is called a Docker, and Google developed a Docker orchestration tool called Kubernetes [3,4]. Kubernetes executes containers in multiple hosts and resolves problems in container networking among hosts and resource distribution of the host machine. However, it has an issue of performance degradation when organizing the Container Network Interface (CNI), and it does not guarantee network flexibility [5].

There are various research articles on CNIs used in Kubernetes. Hao Zeng et al [6], the performance of networks using existing Kubernetes CNIs was investigated. Network performance was analyzed by using three CNI types that can be used in Kubernetes. The CNIs used in the study were Calico [6], Flannel [7], and Swarm Overlay [8]. According to the tests on each CNI's delay of ping, TCP throughput, and UDP throughput, Calico showed the highest performance. The study revealed that existing CNIs have network issues and cannot perform as well as connected bandwidths. Therefore, this study aimed to configure Kubernetes to provide a high-performance network.

KREONET [10] is a national R & D network managed and operated by Korea Institute of Science and Technology Information (KISTI). KISTI is constructing and developing KREONET-S [11], a software-defined network (SDN) based wide area network infrastructure, as a next generation service model based on KREONET. KREONET-S includes domestic and international networks, and all network factors of KREONET-S

Revised Manuscript Received on May 22, 2019.

Ki-Hyeon Kim, Korea Institute of Science and Technology Information, 245 DaehangnoYuseong Daejeon 34141 Korea, ASIKRKS015, Korea

Dongkyun Kim, Korea Institute of Science and Technology Information, 245 DaehangnoYuseong Daejeon 34141Korea, ASIKRKS015, Korea

Yong-Hwan Kim Korea Institute of Science and Technology Information,245 DaehangnoYuseong Daejeon 34141 Korea, ASIKRKS015, Korea

* E-mail : mirr@kisti.re.kr

infrastructure are operated by the Open Network Operating System (ONOS) [12] control platform for new SDN network operations, management and services. In particular, KREONET-S provides virtually dedicated network (VDN) dynamic establishment services for cutting-edge collaborative research, which requires a large volume of data transmission and control, by establishing a high-performance network requested by users in a short time period. The main VDN services include VDN slicing, VDN federation, virtual network access control (vNAC), and VDN DHCP (vDHCP). VDN Slicing (vSlicing) is a functionality that provides a network that can guarantee the bandwidth requested by users for an arbitrary host group by using the OpenFlow meter table. VDN Federation creates connection data among different SDN network domains, exchanges the data in JSON format, and enables communication among hosts that allow mutual connection. vNAC enhances the security of data communication with external networks for hosts connected with external IP gateways, and vDHCP is a functionality for setting automatic IP allocation to hosts participating in a VDN.

In this paper, we developed new CNIs by interconnecting VDN and Kubernetes to provide users with a high-speed transmission service for a large volume of scientific data and tested their performances. We proceeded with system virtualization through the ONOS controller by applying the developed CNIs to KREONET-S and aimed to demonstrate the outstanding performance of the new CNIs through performance tests.

II. DEVELOPMENT OF AN OPEN-CLOUD COMPUTING PLATFORM

In this paper, we developed an open-cloud computing platform by interconnecting VDN and Kubernetes. In this paper, we explain the differences between the network structures of the existing Kubernetes and that of the Kubernetes in the newly developed open cloud computing platform, as well as the gap between existing CNIs and developed CNIs.

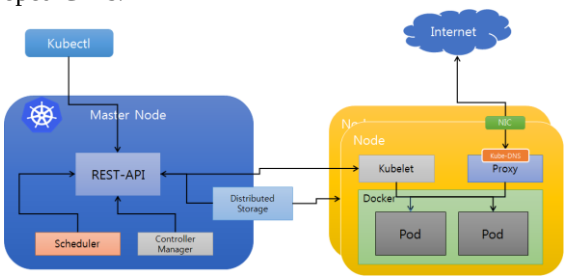

Figure 1. Structure of Kubernetes

The structure of the existing Kubernetes is shown in Figure 1. The network structure of the existing Kubernetes sets main network rules and performs load balancing by using a proxy. Pods can be first created in Kubernetes only when an IP address is allocated, which requires the DHCP server, and resource location information is necessary as it is not known in which node they will be created. Such a network pattern is called a service discovery pattern. Kubernetes uses the internal DNS server to improve the service discovery pattern. When a new resource is created, the IP address and DNS names are mapped with the resource to allow access to it.

The structure of the open cloud computing platform developed in this study is shown in Figure 2. In the platform, Kubernetes organizes a virtual network by connecting to VDN. In order to create resources in Kubernetes, an IP address should be allocated before the resource creation. The IP address can be allocated from vDHCP by calling VDN REST APIs. The created Pods can perform communication after organizing virtual networks based on VDN by using one of the two newly developed CNIs.

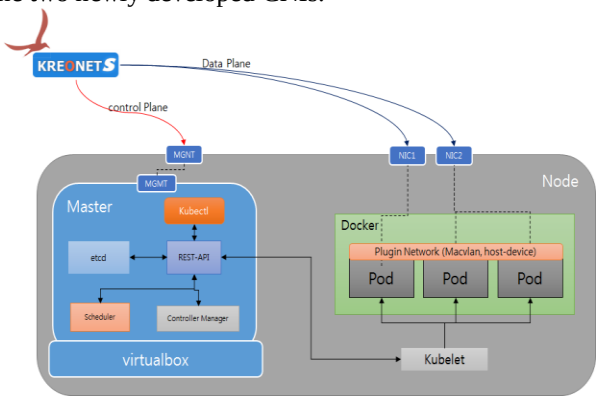

Figure 2. Structure of the Open-Cloud Computing Platform

The CNIs developed in this study are basically organized in two ways. Macvlan can connect multiple hosts and was developed for high performance network, while Host Device was developed for application in a system that requires high performance as it organizes a network with the allocation of one NIC to one Pod. Host Device may need many NICs; however, it is a necessary function for users who need high bandwidth. The existing CNIs in Kubernetes use the overlay network method that uses a bridge network, which causes performance degradation. However, we developed the Macvaln network and host device, which have simpler and smoother network structures by deleting a bridge between the container and host. The Macvlan method can have multiple Mac addresses at the same time with the virtualization of physical devices to the host machine. Host Device can bring and use the actual NIC to a newly created Pod. Figure 3 shows the structure of the developed CNIs (Macvlan and host device).

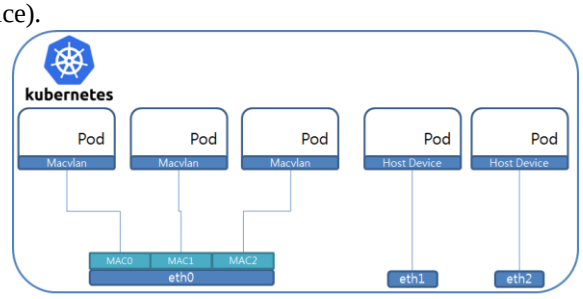

Figure 3. Structure of developed CNIs

III. KUBERNETES PERFORMANCE TEST ENVIRONMENT

This section describes the network performance tests that were performed after

organizing networks using Kubernetes with existing CNIs and the developed CNIs.

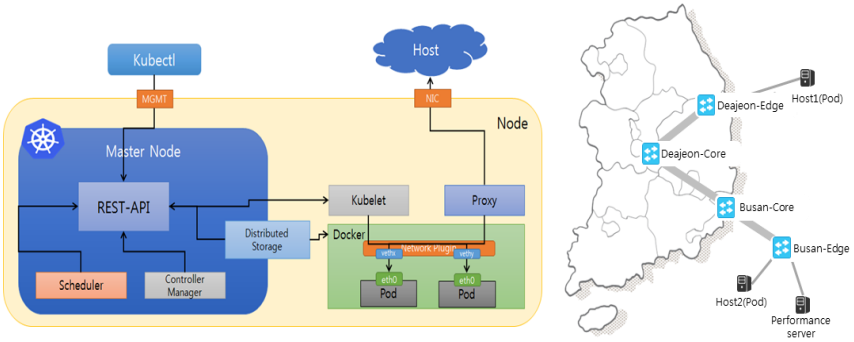

Figure 4. The environment for testing the Performance of CNIs

The environment for testing the performance of the CNIs in Kubernetes is shown in Figure 4. The master and node were installed in one server to organize Kubernetes. The master was organized on a virtual machine (VM) using Virtualbox, while the node was organized in the actual server host. For CNI performance verification in the WAN, rather than LAN section, the Kubernetes environment in Figure 4 was configured in Daejeon, and the performance measurement server was installed in Busan. The performance test was executed by establishing a virtual network with a 1 Gbps bandwidth between the Pod created by Kubernetes in Daejeon based on SDN and the performance measurement server in Busan. In this test, Calico, Flannel, and Weave Net were used among the existing CNIs, and the performance of the CNIs newly developed in this study was also tested in the same environment. In addition, for a performance test between different Kubernetes environments in which the developed CNIs are applied, two independent Kubernetes test environments were set up in Daejeon and Busan with a 10 Gbps virtual network created in between the two to measure the performance between Pods in Daejeon and Busan.

Furthermore, we used Iperf3, a server network performance measurement program to collect test data, and measured the performance of each CNI for 1,000 seconds.

IV. PERFORMANCE RESULT

This section shows the performance results of existing CNIs, i.e., Calico, Flannel, and Weave Net, used in Kubernetes and the CNIs developed in this study.

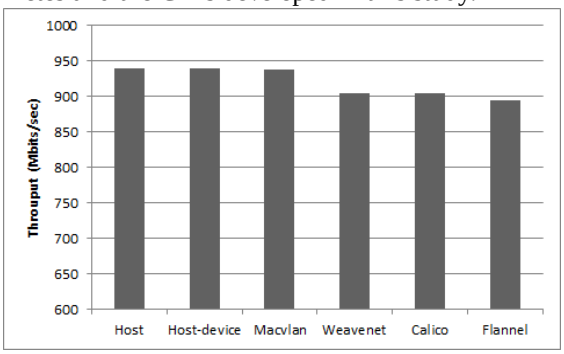

Figure 5. CNI Performance Measurement in the Kubernetes 1Gbps Testbed

Figure 5 shows the performance measurement results of the host and the performance measurement server through the 1Gbps test bed, the performance measurement results of the three CNIs of the existing Kubernetes, and the CNI performance measurement results. Figure 5 shows that Calico has an average performance of 904 Mbps, Flannel has an average of 895 Mbps, and Weave Net has an average of 905 Mbps. On the other hand, the performance of the developed CNI Macvlan is 938 Mbps, and the average performance of the host device is 940 Mbps. That is, when performance measurement is performed at a bandwidth of 1 Gbps, the performance of Macvlan and the host device is similar to that of the actual host, and the performance is higher than that of the conventional CNI of 30 Mbps. Figure 5 shows graphs of performance measurement results of the host and performance measurement server in a 1 Gbps testbed and those of three CNIs used in Kubernetes and the developed CNIs. In Figure 5, the average performances of Calico, Flannel, and Weave Net are 904 Mbps, 895 Mbps, and 905 Mbps. Meanwhile, in the case of the CNIs developed in this article, Macvlan shows an average performance of 938 Mbps while Host Device shows that of 940 Mbps. In other words, when performance is measured in the 1 Gbps bandwidth, the performances of Macvlan and Host Device are similar to that of the actual host, which is more than 30 Mbps higher than those of the existing three CNIs.

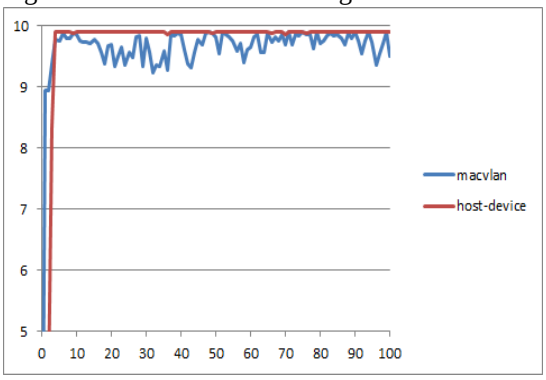

Figure6. Developed CNI performance measurement result viaKubernetes 10Gbps Testbed

When executing a test by applying the developed CNIs in the 10 Gbps bandwidth, the average performance of Macvlan was 9.7 Gbps while that of Host device was 9.88 Gbps. Figure 6 indicates that Macvlan shows a higher performance but an unstable network performance. However, it was confirmed that Host Device displays a stable network performance. This is because the NCI of the actual host is allocated to Host Device, which assures a high and stable

performance. Moreover, the developed CNIs can use flexible networks by connecting with a VDN. They can provide users with networks meeting their requirements without the complicated network settings that were previously required by using the network virtualization function of a VDN with more secure networking.

Figure 7 confirms the possibility of system virtualization of Pods created in Kubernetes via the ONOS controller when the open cloud computing platform is applied to KREONET-S. The figure shows that Pods were created by Macvlan and Host Device CNIs in Kubernetes systems installed in Daejeon and Busan, and the currently created Pods can be checked in the ONOS UI.

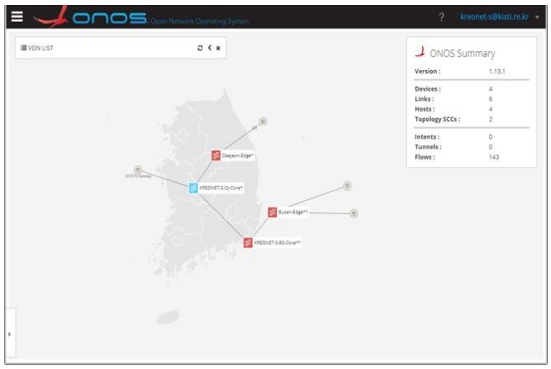

Figure 7. Pod Connected viaONOS GUI

V. CONCLUSION

Data centers generally use container-based server virtualization technology to provide light-weighted and highly scalable services. However, container-based server virtualization has network performance degradation and lack of flexibility issues. In order to address these issues, we have developed an open cloud computing platform by orchestrating Kubernetes and VDNs. In this study, we tested and measured the performance of new CNIs (Macvlan and Host Device) developed on the open cloud computing platform and existing CNIs (Calico, Flannel, and Weave Net) of Kubernetes in the system organized over the software-define wide area network (SDN-WAN) between Daejeon and Busan. When tested at the bandwidth of 1 Gbps, the performances of Calico, Flannel, and Weave Net were 904 Mbps, 895 Mbps, and 905 Mbps, respectively. For the developed CNIs, Macvlan showed a performance of 938 Mbps while Host Device showed 940 Mbps. As a result, with the existing CNIs, it was difficult to expect high performance results as they establish networks with a bridge and overlay methods; however, the developed CNIs demonstrated similar network performance results as the actual host does. Moreover, the performance result showed over 9.7 Gbps throughput with a 10 Gbps NIC provisioned for performance tests. As such, the study indicates excellent network performance (throughput) in terms of the developed CNIs.

FUTURE WORK

It is possible to establish high-performance network based on the virtualization environment by using the open cloud computing platform developed in this study. We are planning to establish data transfer nodes (DTNs) [13] which are specialized computing nodes for high-performance science big data transmission, by using the cloud platform developed in this study. Furthermore, more researches will be performed for virtualization of Science DMZ [14] architecture that should accommodate dynamic, on-demand, and virtually dedicated networking over KREONET-S.

ACKNOWLEDGMENT

"This research was supported by Korea Institute of Science and Technology Information (KISTI)"

REFERENCES

1. Gartner, "Identifies 10 Key Actions to Reduce IT Infrastructure and Operations Costs by as Much as 25 Percent", Sep., 28, 2011
2. Roger S. Pressman "Software Engineering A Practltiners' Approach" 3rd Ed. McGraw Hill
3. Felter, Wes, Alexandre Ferreira, Ram Rajamony, and Juan Rubio. "An updated performance comparison of virtual machines and linux containers." In Performance Analysis of Systems and Software (ISPASS), 2015, pp. 171-172.
4. Chang, Chia-Chen, Shun-Ren Yang, En-HauYeh, Phone Lin, and Jeu-YihJeng. "A Kubernetes-Based Monitoring Platform for Dynamic Cloud Resource Provisioning." In GLOBECOM 2017-2017 IEEE Global Communications Conference, 2017, pp. 1-6.
5. Dusia, Ayush, Yang Yang, and Michela Taufer. "Network quality of service in docker containers." In Cluster Computing (CLUSTER), 2015, pp. 527-528.
6. Zeng, Hao, Baosheng Wang, Wenping Deng, and Weiqi Zhang. "Measurement and Evaluation for Docker Container Networking." In Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), 2017 International Conference on, 2017, pp. 105-108.
7. CALICO web site, Retrieved Sep., 08, 2018, from https://www.projectcalico.org/
8. FLANNEL web site, Retrieved Sep., 09, 2018, from https://coreos.com/flannel/docs/latest/
9. SWARM web site, Retrieved Sep., 09, 2018, from https://docs.docker.com/network/network-tutorial-overlay/#walkthrough/
10. KREONE web site, Retrieved Sep., 06, 2018, from http://www.kreonet.net/
11. KREONET-S web site, Retrieved Sep., 08, 2018, from http://www.kreonet-s.net/
12. ONOS web site, Retrieved Sep., 07, 2018, from http://www.onosproject.org/
13. ESnet web site, Retrieved May., 22, 2018, from https://es.net/science-dmz/DTN/
14. Dart, Eli, Lauren Rotman, Brian Tierney, Mary Hester, and Jason Zurawski. "The science dmz: A network design pattern for data-intensive science." Scientific Programming 22, no. 2 (2014): 173-185.