International Journal of Electrical and Computer Engineering (IJECE)
Vol. 10, No. 2, April 2020, pp. 1507~1514
ISSN: 2088-8708, DOI: 10.11591/ijece.v10i2.pp1507-1514

Performance analysis of container-based networking Solutions for high-performance computing cloud

Sang Boem Lim[1], Joon Woo[2], Guohua Li[3]
[1]Department of Smart ICT Convergence, Konkuk University, Korea
[2,3]National Institute of Supercomputing and Networking, Korea Institute of Science and Technology Information, Korea

Article Info

Article history:

Received Jul 4, 2019
Revised Oct 11, 2019
Accepted Oct 20, 2019

Keywords:

Cloud computing
Container-based networking
HPC
Network
Performance analysis

ABSTRACT

Recently, cloud service providers have been gradually changing from virtual machine-based cloud infrastructures to container-based cloud-native infrastructures that consider performance and workload-management issues. Several data network performance issues for virtual instances have arisen, and various networking solutions have been newly developed or utilized. In this paper, we propose a solution suitable for a high-performance computing (HPC) cloud through a performance comparison analysis of container-based networking solutions. We constructed a supercomputer-based test-bed cluster to evaluate the serviceability by executing HPC jobs.

Copyright © 2020 Institute of Advanced Engineering and Science.
All rights reserved.

Corresponding Author:

Guohua Li,
National Institute of Supercomputing and Networking,
Korea Institute of Science and Technology Information,
245 Daehak-ro, Yuseong-gu, Daejeon, 34141, Korea.
Email: ghlee@kisti.re.kr

1. INTRODUCTION

Traditionally, high-performance computing (HPC) was used mainly by natural science areas like weather forecasting, molecular biology, and space exploitation. HPC becomes very demanding technology due to the importance of big data analysis, that cannot be processed with the traditional computing environment [1]. A wide range of computer architecture is, also, required to process big data. One solution to these requirements is to add cloud capability to the HPC environment. Cloud computing can easily adart rapid changes of hardware and sofrtware techologies. By using cloud computing, most of the users also can reduce analysis time and cost of hardware and software [2].

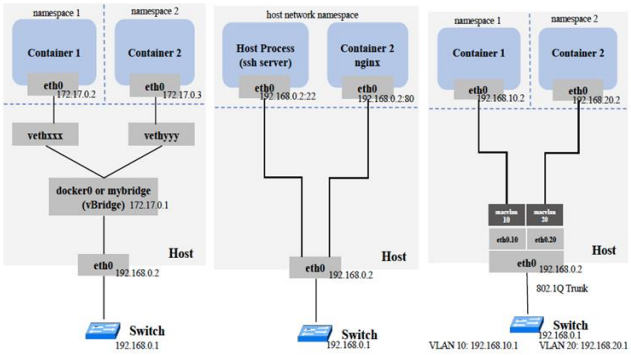
The design of networks in a cloud infrastructure configuration is usually divided into public, management, and guest networks [3]. A guest network is capable of data communication between virtual instances which are running on one host, multiple hosts, or across the different subnets [4-6]. The Docker [7] container platform is an open-source container management project launched by Docker Inc. This is a lightweight container technology that bundles and runs the service operating environment. When configuring a cloud environment based on the Docker container, an orchestration software such as Kubernetes or Docker Swarm is needed to effectively manage and efficiently allocate the resources required for containers [8-10].

The goal of this paper is to evaluate a suitable container-based network solution for HPC cloud by performing benchmark tests using Message Passing Interface (MPI) benchmark suite. We have tested several networking solutions using Kubernetes which has an excellent auto-recovery capability. This study includes a result of the performance tests on network bandwidth of various cluster network configurations and an evaluation of the HPC serviceability of the bare-metal and container. Additionally, a summary of the evaluation result and recommendation on container-based network solution is provided.

2. RELATED WORKS

2.1. Container networking on a single node

There are two types of container-based networking solutions that are related to a sing node and multiple nodes. Representative networking solutions for containers on a single node are divided into three catagories: bridge networking, host networking, and macvlan networking [11]. In the bridge networking [12], a *docker0* or user-defined bridge is created through the physical network interface to control the traffic between containers in two namespaces on a single host Figure 1(a). A host process [13], such as the *sshd* daemon, is created with a specific port while another main service daemon in a container is created with another port having the same virtual IP address in the same namespace for the host networking Figure 1(b). The macvlan networking [14] devides the physical network interface with *Macvlan tags* corresponding to containers in different namespaces Figure 1(c).

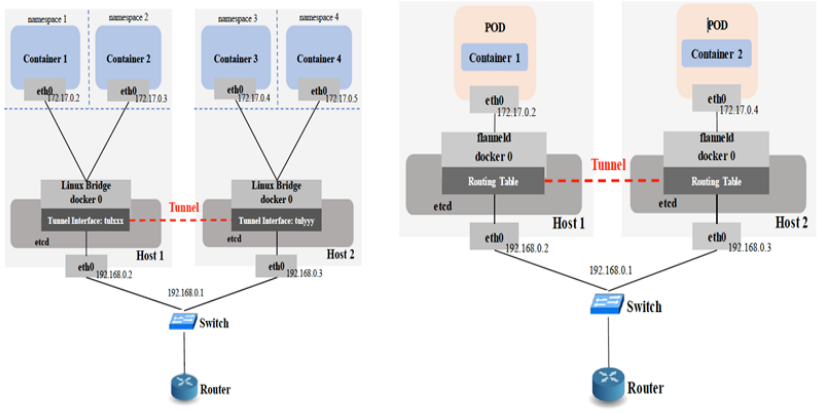


(a) Bridge Networking (b) Host Networking (c) Macvlan Networking

Figure 1. Container networking solutions on one host [15]

2.2. Container networking on multiple nodes

An overlay network [16] is the most commonly used container network on multiple nodes. An overlay network configuration is needed to allow containers to communicate with each other on multiple nodes [17-19]. Representative overlay networking solutions for containers are divided into two catagories: linux bridge overlay network and Flannel networking. Docker Overlay Network and Weave Net networking solutions use the Linux Bridge driver as a tunnel interface to form a tunnel for the network traffic between containers on different hosts, as shown in Figure 2(a). Weave Net creates its own Weave Bridge as a virtual router called *vRouter*. Flannel network is developed specifically for Kubernetes, and is easy to configure. Figure 2(b) shows that it corresponds to the POD structure of Kubernetes and references the Routing Table through the *docker0* interface via *flanneld*.



(a) Linux Bridge Overlay Netwo (b) Flannel networkin

Figure 2. Container networking dolutions on multiple nodes [15]

Calico networking [20] is a networking solution that is optimized for the native cloud and simple, scalable security technology. Unlike other multi-host overlay networking solutions explained above, it uses Calico's own driver instead of the Linux Bridge kernel driver. As shown in Figure 3, traffic through all the containers is routed according to the in-kernel rule by utilizing the firewall functions.

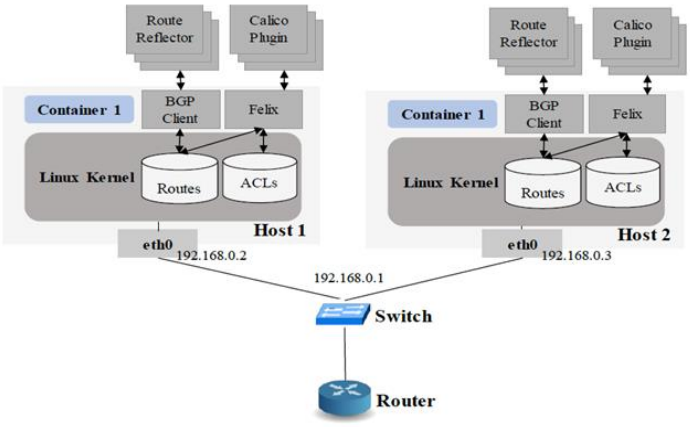


Figure 3. Calico networking [15]

3. PERFORMANCE ANALYSIS

3.1. Bandwidth test

For the benchmark environment, we are using a cluster specified in Table 1 with CentOS operating system. One controller node and four work nodes are assigned. Among the four work nodes, two nodes have 7Gbytes of memory and twelve cores and the other two nodes have 15Gbytes of memory and eight cores. As specified in Table 2, a network bandwidth test is performed while changing the network configuration and the cluster configuration at two bare-metal nodes constituting a 10G Ethernet network. The Docker network configuration is divided into Docker Linux Overlay, Weave Net, Flannel, Calico with IP-in-IP, and Calico without the IP-in-IP configuration. The cluster configuration is devided into etcd for Docker Linux Overlay, Kubernets for Weave Net, Flannel, Calico with IP-in-IP, and Calico without the IP-in-IP configuration, and 10G ethernet for bare-metal.

Table 1. Test-bed cluster specification

Node	CPU	Core	Memory
Controller	Intel(R) Xeon(R) CPU E5530 @ 2.40GHz	8	16G
Work node 1	Intel(R) Xeon(R) CPU E5645 @ 2.40GHz	12	7G
Work node 2	Intel(R) Xeon(R) CPU E5645 @ 2.40GHz	12	7G
Work node 3	Intel(R) Xeon(R) CPU E5620 @ 2.40GHz	8	15G
Work node 4	Intel(R) Xeon(R) CPU E5620 @ 2.40GHz	8	15G

Table 2. Test-bed network configuration

	Docker Network Configuration	Cluster Configuration
1	Docker Linux Overlay	etcd
2	Weave Net	
3	Flannel	
4	Calico (with IP-in-IP)	Kubernetes
5	Calico (without IP-in-IP)	
6	Bare-metal	Host network with 10G Ethernet

For the bandwidth test, we are using *Iperf3* [14] bandwidth tool. Iperf3 is a bandwidth measurment tool to measure the maximum achievable bandwidth on IP network. As shown in Figure 4, Calico without the IP-in-IP configuration performs much better than other overlay networks. Upon testing with Iperf3, the bandwidth of the Calico overlay network without IP-in-IP is 8,500 Mbits/s, which is as high as the value for bare metal that performs 9,300 Mbits/s. We believe this result is because Calico uses the Calico driver which is loaded into the kernel, but other overlay solutions use the default Linux Bridge driver in the kernel.

However, with the IP-in-IP configuration, Calico exhibits a bandwidth of 4,600 Mbits/s, that is, half of 8,500 Mb/s. For this reason, in cross different subnet environments, the network performance is 50% of that of bare metal in any one subnet environment.

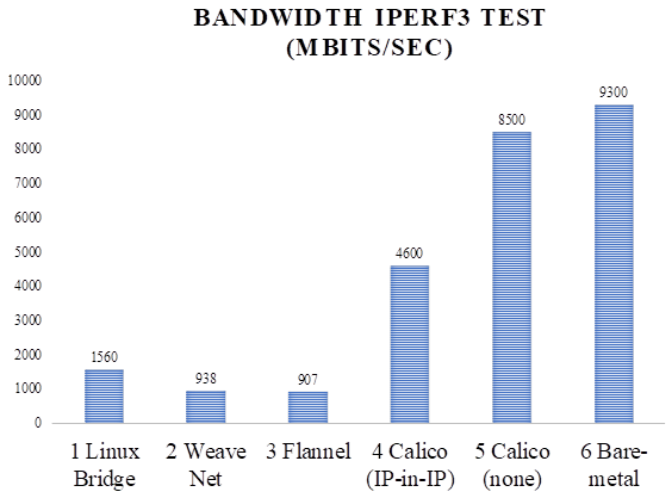


Figure 4. Bandwidth test for networking solution

3.2. Throughput test

Based on this performance comparison, we selected Calico to construct a cloud environment for HPC services. The performance verification of the environment in which HPC services can be provided was compared with the measured throughput of TCP/UDP [21] using a HPC performance benchmark tool – HPCBENCH [22]. This test compares bare metal, Calico, and Canal which incorporates the Calico security policy for Flannel. All the tests were done after setting MTU=9000 on all devices such as the switch, host, and container. We chose Flannel from several networking solutions that use the Linux Bridge kernel driver because it is easy to be integrated with Kubernetes and configure with Calico's security policy as Canal networking. Therefore, using only Calico and Canal, we can also acquire get the best results without testing other networking solutions.

As shown in Figure 5, the TCP throughput of Calico is equal to 80% of that of bare metal, but Canal is equal to only 20% of the bare metal value. For UDP, Calico's throughput is equal to 70% of that of bare metal, but for Canal, it is equal to only 30%. In addition, the loss-rate of Calico's UDP is almost 0%, which is more stable than that of bare-metal. Canal has a 25% loss-rate which is inferior in terms of stability.

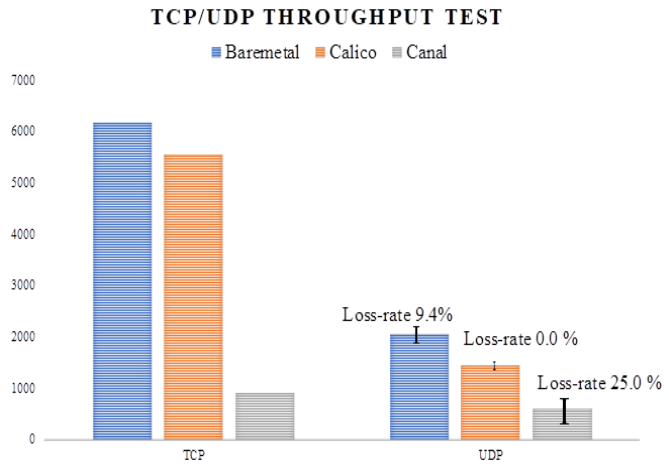


Figure 5. TCP/UDP throughput test (Mb/s)

4. EVALUATION

To evaluate the HPC serviceability of the selected Calico networking, we compared it with bare metal, Singularity, key-value store network, and swarm. Singilarity sharing the host networks and which is specifically designed for use with HPC services [23-25]. We constructed two nodes for running HPL [26] that is a portable implementation of the high-performance Linpack benchmark for distributed-memory computers. In order to run HPL accurately, we need to set the partitioning blocking factor (NB) and the memory usage factor (NS). In this evaluation in Figure 6, we tested various conditions using NS and NB factor. We are using NS factors of 80% and 90%, which means that HPL uses up to 80% and 90% of available memory. For portioning block size, we are using 64 bytes, 128 bytes, and 256 bytes. The test result shows bare metal, the Docker container with Calico and Kubernetes, and Singularity offer similar levels of performance.

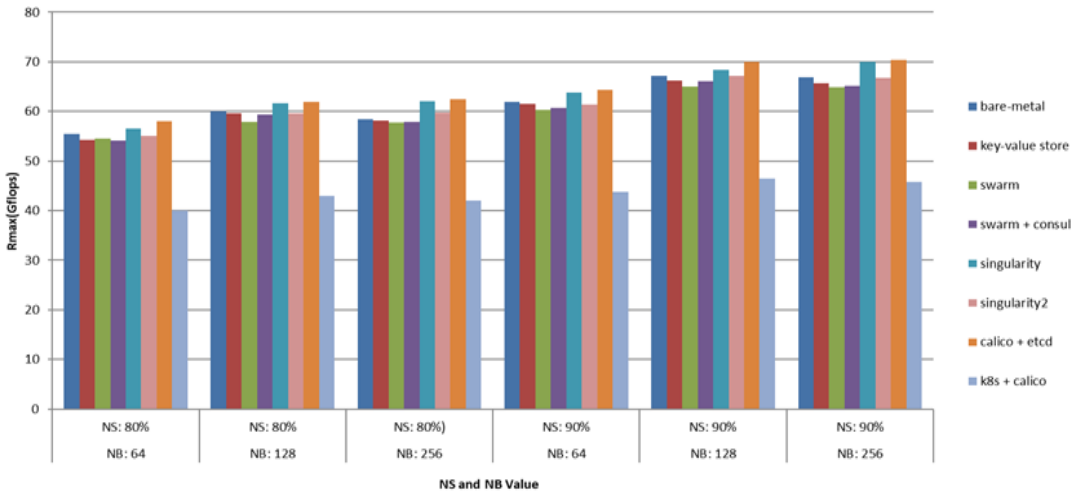


Figure 6. MPI bandwidth test results

We constructed two nodes for running parallel MPI tasks to check the bandwidth using the osu-micro-benchmarks MPI benchmark tool [27] Figure 7. The test results show that bare metal, the Docker container with Calico and Kubernetes, and Singularity all offer similar levels of performance in terms of bandwidth and latency. In addition, we constructed four nodes (with 72 cores) for running parallel MPI tasks to check the MPI all-to-all personalized exchange latency test Figure 8 and the MPI allgather personalized exchange latency test Figure 9 which incurs the highest communication load among all the functions provided by the benchmark tool. The test result shows that bare metal, the Docker container with Calico and Kubernetes, and Singularity all offer the same levels of latency performance even in a multiple-node structure.

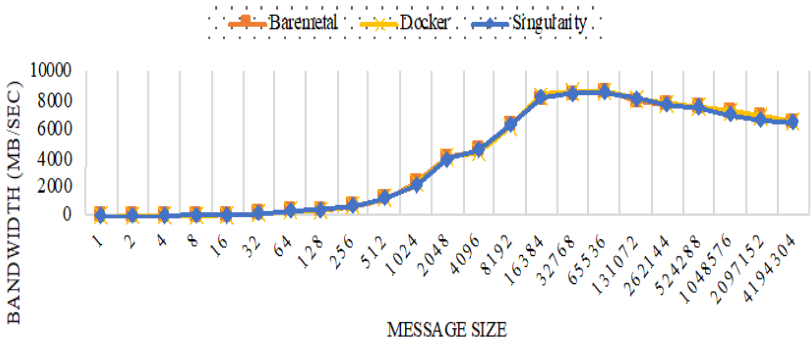


Figure 7. MPI bandwidth test results

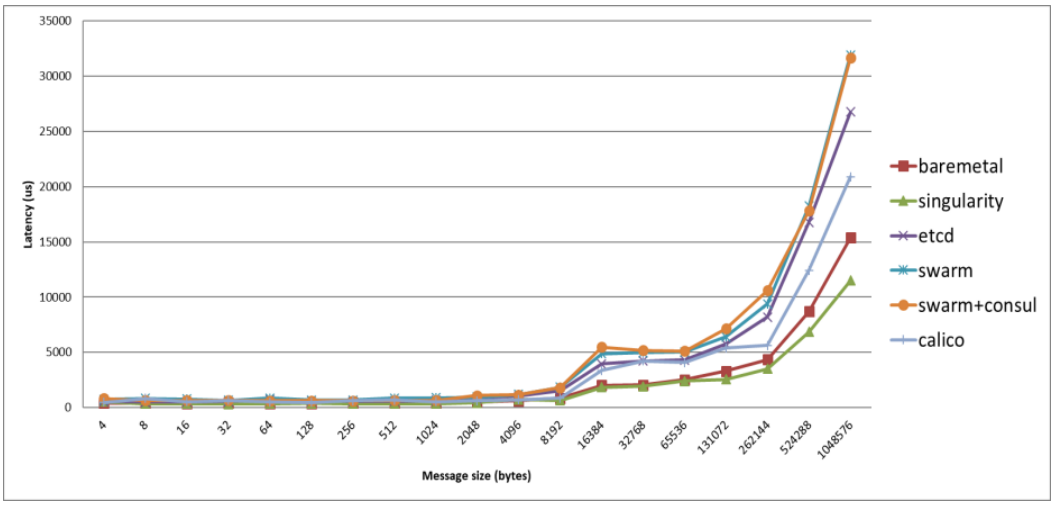


Figure 8. MPI all-to-all personalized exchange latency test results

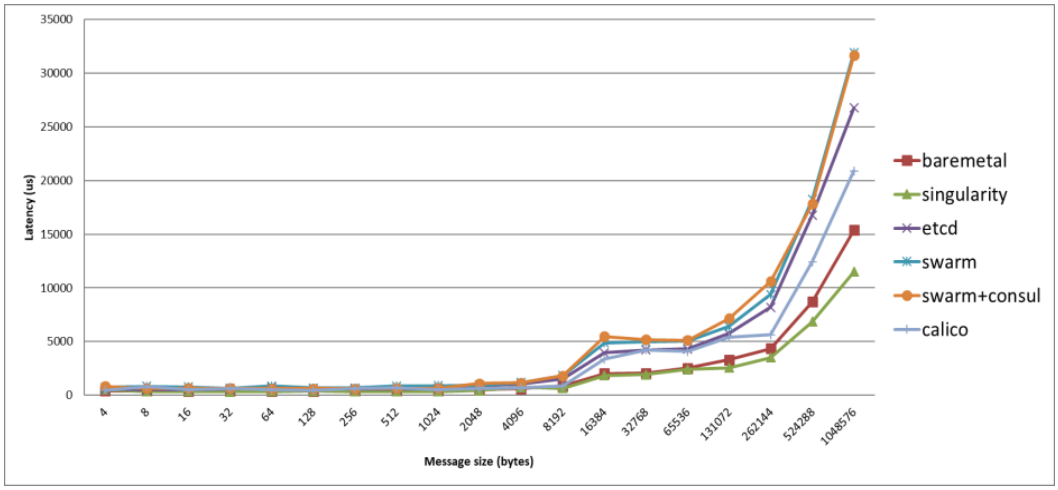


Figure 9. MPI allgather personalized exchange latency test results

5. CONCLUSION

In the present study, we addressed container networking solutions on multiple hosts for HPC services. Based on the results of these tests, we believe that Calico offers the best performance while a comparative analysis revealed that it is the easiest means of configuring an HPC environment. The main contribution of the present study was the testing of the performance of a real supercomputer-based HPC cluster. Based on our performance comparison analysis, we have proposed the best container-based networking solution for HPC services, attaining excellent results which are comparable with those for bare metal. In the future, we will determine the service value by executing network-intensive parallel jobs with which we can evaluate our findings.

ACKNOWLEDGEMENTS

This study was performed as a subproject of the KISTI project entitled "The National Supercomputing Infrastructure Construction and Service [K-19-L02-C01-S01]".

REFERENCES

[1] B. Gourav, *et al.*, "A Novel Approach for Clustering Big Data based on MapReduce," *International Journal of Electrical and Computer Engineering*, vol. 8, no. 3, pp. 1711-1719, Jun 2018.

[2] R. Rajak, " A Comparative Study:Taxonomy of High Performance Computing (HPC)," *International Journal of Electrical and Computer Engineering*, vol. 8, no. 5, pp. 3386-3391, Oct 2018.

[3] A. Celesti, *et al*., "Evaluating Alternative DaaS Solutions in Private and Public OpenStack Clouds." *Software - Practice and Experience*, vol. 47, no. 9, pp. 1185-1200, Sep 2017.

[4] K. Suo, Y. Zhao, W. Chen and J. Rao, "An Analysis and Empirical Study of Container Networks," *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications*, Honolulu, HI, 2018, pp. 189-197.

[5] Youngki Park, Hyunsik Yang, Younghan Kim, "Performance Comparison of Container Networking Technologies", *Journal of Korea Information and Communications Society*, vol. 44, no.01, pp.0158-0170, Jan 2019

[6] Pasquale Salza, Filomena Ferrucci, "Speed Up Genetic Algorithms in the Cloud Using Software Containers", *Journal of Future Generation Computer Systems*, vol. 92, pp. 672-681, Sep 2019

[7] M. De Benedictis, *et al.*, "Integrity Verification of Docker Containers for a Lightweight Cloud Environment." *Future Generation Computer Systems*, vol. 97, pp. 236-246, Aug 2019.

[8] M. K. Hussein, *et al*., "A Placement Architecture for a Container as a Service (CaaS) in a Cloud Environment." *Journal of Cloud Computing*, vol. 8, no. 1, Dec 2019.

[9] Max Alauna, Eric Vial, Nuno Neves, Fernando M.V. Ramos, "Secure Multi-Cloud Network Virtualization", *Journal of Computer Networks*, vol. 161, pp. 45-60, Oct 2019

[10] Marco De Benedictis, Antonio Lioy, "Integrity Verification of Docker Containers for a Lightwight Cloud Environment", *Journal of Future Generation Computer Systems*, vol. 97, pp. 236-246, Aug 2019

[11] C. Ramon-Cortes, *et al.*, "Transparent Orchestration of Task-Based Parallel Applications in Containers Platforms." *Journal of Grid Computing*, vol. 16, no. 1, pp. 137-160, Feb 2018.

[12] . Buh, *et al.*, "Adaptive Network-Traffic Balancing on Multi-Core Software Networking Devices," *Computer Networks*, vol. 69, pp. 19-34, Aug 2014.

[13] Z. Zhang, *et al.*, "Lark: An Effective Approach for Software-Defined Networking in High Throughput Computing Clusters," *Future Generation Computer Systems*, vol. 72, pp. 105-117, Jul 2017.

[14] J. Struye, *et al.*, "Assessing the Value of Containers for NFVs: A Detailed Network Performance Study," in *13th International Conference on Network and Service Management, CNSM 2017*, 2017, pp. 1-7.

[15] J. Langemak, "Docker Networking Cookbook, " Packt Publishing, 2016.

[16] J. Zhang, *et al.*, "On Achieving the Maximum Streaming Rate in Hybrid Wired/Wireless Overlay Networks," *IEEE Wireless Communications Letters*, vol. 8, no. 2, pp. 472-475, Apl 2019.

[17] Sandrine Vaton, Olivier Brun, Maxime Mouchet, Pablo Belzarena, Isabel Amigo, Balakrishna J. Prahbhu, Thierry Chonavel, "Joint Minimization of Monitoring Cost and Delay in Overlay Networks: Optimal Policies with a Markovian Approach", *Journal of Network and Systems Management*, vol. 27, issue 1, pp. 188-232, Jan 2019

[18] Min-Ho Ha, Zali Yang, Jasmine Siu Lee Lam, "Port Performance in Container Transport Logistics: A multi-stakeholder Perspective", *Journal of Transfport Policy*, vol. 73, pp.25-40, Jan 2019

[19] Tukasz Makowski, Paola Grosso, "Evaluation of Virtualization and Traffic Filtering Methods for Contianer Networks", *Journal of Future Generation Computer Systems*, vol. 93, pp. 345-357, Apr 2019

[20] A. V. Baranov, *et al.*, "Methods of Jobs Containerization for Supercomputer Workload Managers," *Lobachevskii Journal of Mathematics*, vol. 40, no. 5, pp. 525-534, May 2019.

[21] S. F. Szilaagyi, *et al.*, "Throughput Performance Comparison of MPT-GRE and MPTCP in the Fast Ethernet IPv4/IPv6 Environment," *Journal of Telecommunications and Information Technology*, vol. 2018, no. 2, pp. 53-59, 2018.

[22] B. Huang, *et al.*, "Hpcbench - A Linux-Based Network Benchmark for High Performance Networks," in *19th International Symposium on High Performance Computing Systems and Applications, HPCS 2005*, pp. 65-71, 2005.

[23] C. Yong, *et al.*, "Proposal of Container-Based HPC Structures and Performance Analysis," *Journal of Information Processing Systems*, vol. 14, no. 6, pp. 1398-1404, 2018.

[24] P. China Venkanna Varma, Venkata Kalyan Chakravarthy K., V. Valli Kumari, S. Viswanadha Raju, "Analysis of a Network IO Bootleneck in Big Data Environments Based on Docker Containers", *Journal of Big Data Research*, vol. 3, pp. 24-28, Apr 2016

[25] G. Calarco, M. Casoni, "On the Effectiveness of Linux Containers for Network Virtualization", *Journal of Simulation Modelling Pratice and Theory*, vol. 31, pp. 169-185, Feb 2013

[26] T. Sterling, *et al.*, "A Survey: Runtime Software Systems for High Performance Computing." *Supercomputing Frontiers and Innovations*, vol. 4, no. 1, 2017, pp. 48-68, 2017.

[27] S. Hunold and A. Carpen-Amarie, "Reproducible MPI Benchmarking is Still Not as Easy as You Think," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 12, pp. 3617-3630, 1 Dec. 2016.

BIOGRAPHIES OF AUTHORS

Prof. Sang Boem Lim received his PhD in Computer Science from Florida State University in 2003. Previously, he leaded Korea e-Science Project as technical Team Leader at Korea Institute of Science Technology Information (KISTI) Supercomputing Center. He currently is an assistant professor for Konkuk University. His research interests include high-performance computing and cloud computing.

Dr. Guohua Li received her PhD in Interdisciplinary IT from Konkuk University in 2018. She completed her M.S. degree from Konkuk University, KOREA, in 2013. She participated in cloud computing application for HPC service part-time training at Korea Institute of Science Technology Information (KISTI) as a student researcher from 2013 to 2015. She also participated in the development of container-based HPC cloud service platform as a co-researcher from 2016 to 2017. She currently is a post-doctoral researcher for KISTI. Her research interests include high-performance computing and cloud computing.

Dr. Joon Woo received his Ph.D. at Chungnam National University in 2018. He was involved in building infrastructure for the HPC service at KISTI Supercomputing Infrastructure Center. He currently is a senior researcher for KISTI. His research interest includes HPC, containerized HPC, and HPC Cloud.