Performance analysis of multi services on container Docker, LXC, and LXD

Adinda Riztia Putri, Rendy Munadi, Ridha Muldina Negara
Adaptive Network Laboratory, School of Electrical Engineering, Telkom University, Indonesia

Article Info

Article history:

Received Nov 3, 2019
Revised Feb 3, 2020
Accepted Mar 15, 2020

Keywords:

Container
Docker
LXC
LXD
PaaS

ABSTRACT

The emergence of the container in various cloud platforms from Open Stack to Google Cloud Platform has marked the industry interest in opting for container as their cloud service solution. However, the cloud users should aware of performance overheads of different virtualization solutions in order to avoid quality of service degradation because different container platforms delivered different performances. This research evaluated how different container platforms (Docker, LXC, and LXD) impacted in running different TCP services and also measured system performance of each container compared to the native system without any container solution based on overall performance metrics. This research focuses on the three most used PaaS: FTP Server, Web Server, and Mail Server. Related to previous works, our evaluation results show that performance could vary between containers. In terms of system performance, LXD shows better performance while server performance result varies depending on what service is being evaluated.

Corresponding Author:

Ridha Muldina Negara,
School of Electrical Engineering,
Telkom University,
Jalan Telekomunikasi, Buahbatu, Bandung 40257, Indonesia.
Email: ridhanegara@telkomuniversity.ac.id

1. INTRODUCTION

Virtualization has played a great role in the growth of cloud computing and considered as one of foundation technology for clouds [1]. There are two kinds of virtualization solution, hypervisor and container. The latter is the newest technology in virtualizing cloud infrastructures. Adopting and deploying technologies is critical to build newer paradigm to keep up with the growth of exchanged data and the consequent need to increase capability of data centers by means of server virtualization [2-5]. Therefore, the main focus of virtualization has shifted from hypervisor-based virtualization to container-based virtualization. Containerization offered light-weight virtualization as contrast to hypervisor-based virtualization where virtualization overhead is considered as an issue [6-7].

Container achieves generally better performance when compared to traditional virtual machine as what hypervisor-based virtualization offers [2]. Cloud users should aware of which containerization technology to adopt according to their need. In this research we evaluated different container platforms widely used by enterprises, that is Docker, LXC, and LXD. Although these containers are derived from the same idea of the emergence of Linux container, performance of each container is different depending of what service is running on top of it. Then, raised a question of what container should a cloud user use in order to run a certain kind of cloud service [8]. In order to answer it, we evaluated the most common cloud PaaS service: web server, FTP server, and mail server when run on the top of different container platform. System performance evaluation also performed in order to take a deeper attention of how well container manage its filesystem, CPU performance intensive, and RAM speed by doing several batch benchmark to stress container system.

− Related work

In this part, we provide an overview of different technologies and how they're leading to the use of containerization, also previous works regarding performance evaluation of virtualization technologies, most of it is related to containers.

Hypervisor and container:

Both container and hypervisor are used to create an abstraction level to effectively use the available resource to support virtualization [9-14]. Virtualization itself evidently can increase hardware utilization rates from 10 or 15 percent to 70 or 80 percent [15]. Hypervisor works by virtualizing hardware-level and isolates necessary resources such as networking, disk, memory, and CPU to create a virtual machine (VM). VM creates its own kernel on top of it and can only run one OS at a time [16]. Container, on the other hand, also isolates these resources but not create another kernel on the top of the host machine. Consequently, the container only provides OS-level virtualization when there is an OS already running while hypervisor provides hardware-level virtualization where there is no running OS. While virtual machine in hypervisor has to boot up its own kernel, the container is using the working kernel of the host [16, 17]. This benefit container, in reduced startup time and lower virtualization overhead, but lack of isolation processes and security.

Figure 1 provides a better picture of how hypervisor compared to container in architecture. Container fundamental concept is to make virtual instances, share a single host OS and relevant libraries, drivers or binaries [18]. The possibility of making such a lightweight virtualization lies behind the existence of namespace, cgroups, and chroot. Namespaces wrap a global system resource on the host and make it appear to the processes within the same namespace (container) as though they have their own isolated instances of the global resource [19]. Cgroups used to allocate resource such as CPU, memory, network, and disk access bandwidth among user-defined groups of process (container) [19]. Chroot is a Linux command to change the root directory of the current process and its children to a new directory [8].

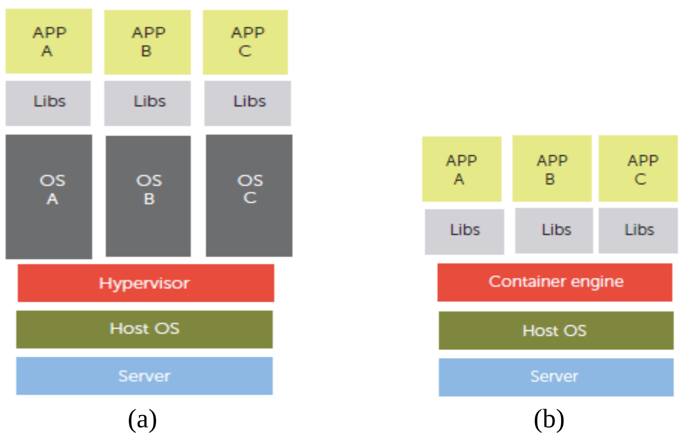


Figure 1. Comparison of (a) Hypervisor-based virtualization architecture vs (b) Container-based virtualization [20]

There are various works related to performance analysis of virtualization in different aspects. In 2014, Dua et al [8] discussed comparison of hypervisor and various containers technologies (Warden container, Openvz, Docker, and lmctfy) to support PaaS. It concluded that containers have inherent advantage over VM because of performance improvement and reduced startup time. Like Dua et al, Morabito et al [2] adopted the idea of comparing hypervisor and container in a more detailed parameter than the previous work. They focused of two containers, Docker and LXC. LXC is said to be a more low-level solution that has been available a long time while Docker is on the higher-level platform. On summary, Morabito et al also concluded that containers perform well and suggested to take LXD into account on the next work.

Still in comparison of hypervisor and container, Sampathkumar [20] compared XEN, KVM, and LXC. He concluded that LXC is preferred to virtualize infrastructure that us dynamic by design, and does not require a high degree of isolation. While, Docker is useful for the purpose of application development where developers need a variety of library rather than a single OS. In 2016, Gupta et al [21] evaluated LXD and Docker and examines the performances of a set of stress test based on a number of benchmark for various aspects such as computation power, memory bandwidth, memory latency, I/O bandwidth and memory snapshots and point out LXD performed a bit better than the virtual machine and Docker container in computation speed performance.

2. PROPOSED METHOD

This part explains how we evaluate both performance of the server itself and the corresponding container.

2.1. System model design

This evaluation scenario uses one server and one client connected by a router. Focus on this research is the server part that will be virtualized by container to run a certain application server (FTP server, web server, and mail server). Specifications of hardware used in this research are listed in Table 1.

Table 1. Hardware specification

Processor		RAM		HDD		NIC	
Server	Client	Server	Client	Server	Client	Server	Client
Intel Core i3 (3rd Gen) 3240T/2.9 GHz	Intel Celeron Dual CPU 1000M@1.8 GHz	4 GB DDR3-1600MHz	2 GB	1000GB SATA 3Gb/s 7200 rpm	500 GB	DELL Wireless 1703	1 Gigabit NIC

Note: Server=DELL Inspiron One 2020, Client=HP 1000 Notebook PC

On server side, we installed Ubuntu 16.04 LTS as the host system. Then we run the step by step evaluation scenario:

1. Install container, then perform system performance benchmark
2. Install application server, then server workload benchmark
3. Record the result, then redo step 2 with other application server focused in this research until all application server done the evaluation
4. Redo step no.1 with other container platform

Figure 2 shows architecture of this evaluation. In order to keep stability of the system, everything besides elements inside the virtual environment box is remained constant through evaluation. Virtual environment changes according to what is being evaluated. Application servers evaluated in this research are all TCP-based service. We did not evaluate UDP service because UDP has lower performance in container and has reported as an anomaly [22]. In this research, we performed two procedures of evaluation, system performance evaluation and application server evaluation.

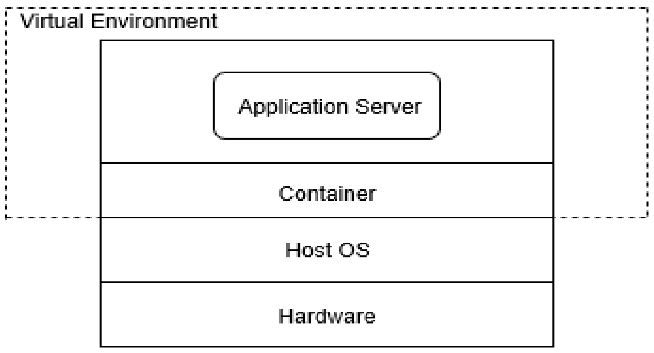


Figure 2. Virtual environment architecture of container evaluation [22]

2.2. System performance evaluation

System performance of this research is based on overall performance metrics, it is including CPU Performance, RAM Speed, and IOzone filesystem as parameters. All OS either in host system or inside container were using Ubuntu 16.04 LTS.

− CPU performance

Different containers platform are tested using Sysbench version 0.4.12 with an argument of CPU testing [23]. Sysbench uses CPU to test if a number is a prime number. In this test, we compared a different number of threads from single thread up to 8 threads. Single thread result of containers was being compared to native system while multiple threads (form 2 to 8) was compared among containers only to get a picture of how well containers perform multiple jobs at the same time.

− RAM speed

RAM Speed test average data rate value from container to host system [19]. This test was done using Phoronix test suite with option of ramspeed batch benchmark

− IOzone filesystem

This test purpose was to test filesystem and hard disk drive utilization performance in read/write activity. Once again we used Phoronix test suite to perform this testing under the option of IOzone batch benchmark. We used 4 GB file with record size 4Kb.

2.3. Application server performance evaluation

The benchmark used to measure each application server performance is different depends on its service.

− Web server service evaluation

The goal of this evaluation is to measure capability of Apache2 as web server on container in serving HTTP requests using Apache Benchmark (ab). Ab sent a number of HTTP request simultaneously to the web server. We used 1000, 2000, 3000, 4000, 5000, 6000, 7000, and 8000 as number of users. Result of this benchmark is total time of evaluation and value of request per time served by web server.

− FTP server service evaluation

This evaluation measured total time, latency, and transfer rate of FTP transaction between server and client. We used Apache JMeter to monitor the FTP transaction.

− Mail server evaluation

We used postal as a benchmark tool to perform this evaluation. Postal sent up to 500 messages with size of 25 MB per message, and built 10 concurrent threads to send these messages using SMTP protocol. Output of this evaluation is number of messages sent by mail server per minute.

3. RESULTS AND DISCUSSION

This part explains briefly of overhead calculation and discussed result of this research evaluation and analyzes it. Each measurement procedure was run multiple times to account system uncertainty [24], in this case we run it for 30 times. To calculate performance overhead, we use overhead ratio [25] as defined in (1).

$$Op = \frac{|Pm - Pb|}{Pb} \times 100\% \tag{1}$$

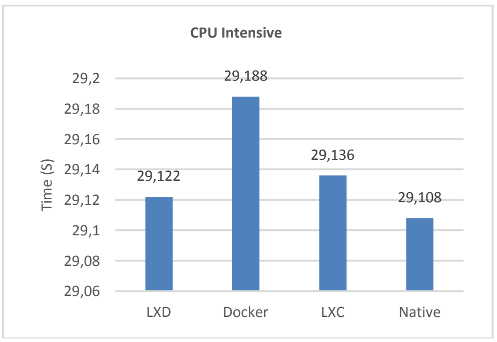
Op refers to performance overhead; Pm denotes the benchmarking result as a measurement of a service feature while Pb indicates the baseline performance of the service feature, in this case is native system. In overall system performance, this value denormalized to value between 0 and 1. 1 is the perfect score held by native system as we assumed native as the baseline of this research and the perfect condition where there was no virtualization overhead affected its functionality.

3.1. Overall system performance result

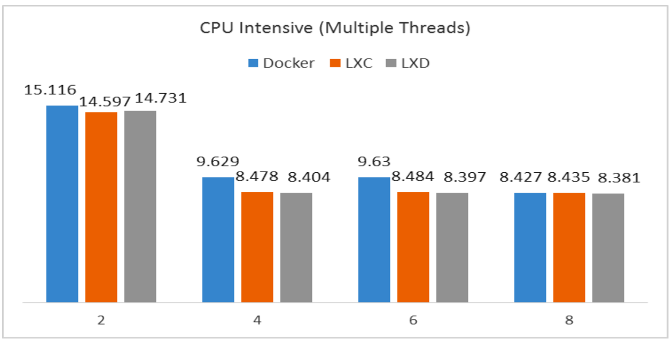
Overall system performance consists of three separated benchmark (CPU performance, RAM speed, IOzone read/write). We summarized in Table 2 from each measurement to determine which system has the best system in terms of overall performance. This test is performed when the server is idle.

3.1.1. Benchmark

This testing scheme is done by giving a load to the CPU which consists of calculations that are determined with maximum prime numbers. The results of this test indicate how long it will take for the CPU to do the calculation. Figure 3(a) shows each result of CPU Performance measurement of the system. LXD has the least overhead with Op 0.047%, LXC Op is 0.094%, and Docker Op is 0.27%. Result shows LXD get job done faster in term of CPU. In order to evaluate CPU in performing multiple threads, we doubled number of thread into 2, 4, 6, and 8. Based on Figure 3(b), CPU performance of each container experienced saturation on thread 6 to 8. This explained how container would be no longer effective when used to perform a multiple task, for example like microservice.



(a)



(b)

Figure 3. Benchmark Result of single and multiple thread CPU performance, (a) Single, (b) Multiple

3.1.2. IOzone read/write

Compared to native, container system performance is degraded to 20% on average based on Figure 4, both IOzone read and IOzone write. LXD shows the least overhead with 14.6% (97.93 Mbps) for IOzone Read and 16.5% (93.61 Mbps) for IOzone write. In the IOzone test, the performance of the container system in the process of reading/writing the hard disk better when the greater the value obtained.

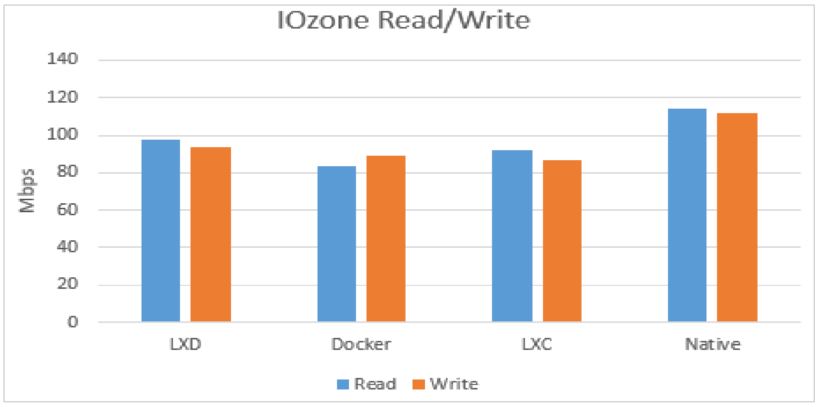


Figure 4. IOzone read/write performance result

3.1.3. RAM speed

We have measured average values of add, copy, scale, and triad operations. Container is well known for its performance in RAM Speed that as good as native. Figure 5 shows performance overhead of each container is less than 3%. Again, LXD point out with only 0.97% overhead (7106.17 Mbps). As seen in Table 2, LXD has the best result in terms of overall performance, proving that optimization from LXC is effective.

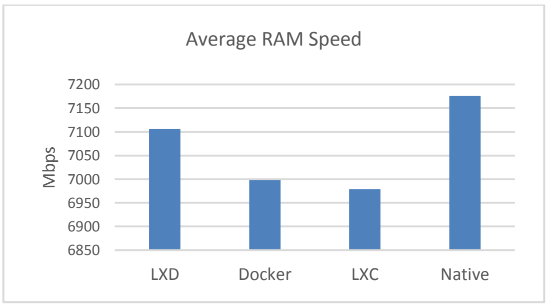


Figure 5. RAM Speed benchmark result

Table 2. Overall performance metric result

Parameter	System			
	Native	Docker	LXC	LXD
IOzone read	1	0.74	0.81	0.85
IOzone write	1	0.82	0.77	0.83
RAM speed	1	0.98	0.97	0.99
CPU intensive	1	0.73	0.91	0.95
Summary	4.00	3.27	3.46	3.62
Overall performance	100%	81.7%	86.5%	90.5%

3.2. Application server performance result

This section describes the performance of each server (FTP server, web server, and email server) that runs on different container platforms. This test is done to see the best platform to run the server and under what conditions.

3.2.1. Web server

The aim of this measurement is to evaluate server speed in serving HTTP requests with a defined number of requests. From figure 6, we concluded that LXD best to serve low concurrent level with user≤2000. For high user (user≥2000) LXC serve with the highest speed.

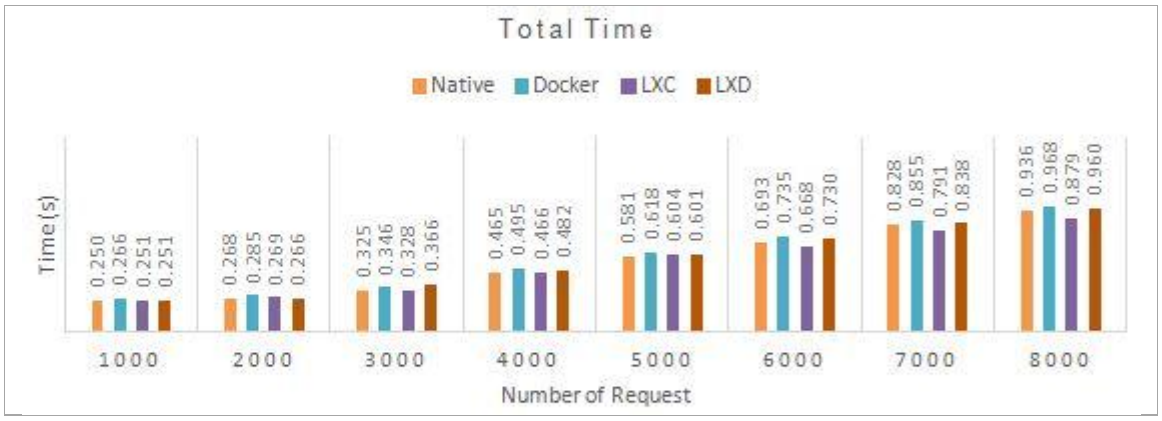


Figure 6. Web server total time result

Figure 7 shows the maximum request a server can serve when a defined number of users access web servers at the same time. Docker issued bottleneck [13] because we used basic network configuration to all containers to see their nature. Op value varied between container and concurrent level. Docker has Op range from 3.1% to 5.9%, LXC from 0.91% to 3.5%, and LXD from 0.005% to 10.8%.

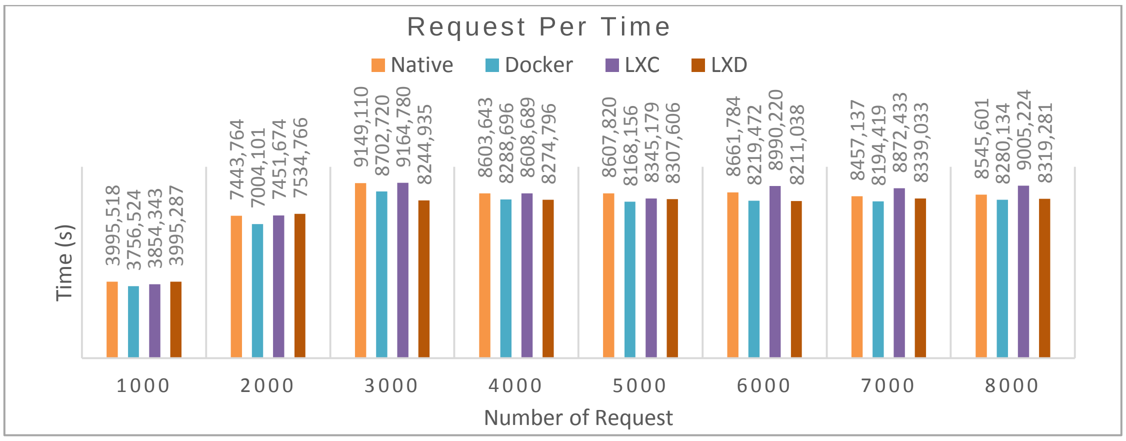


Figure 7. Request per time evaluation result

3.2.2. FTP server

From Figures 8, 9, and 10 we concluded Docker has the biggest overhead compared to LXC and LXD. The Container can speed up the FTP transactions because the container has simpler access and only access the resource it needed rather than took the whole process and be inefficient. Docker recorded 29.9% faster than native, LXC 44.2%, and LXD 47.4%. However, LXC has the lowest overhead in terms of transfer rate but the lowest latency than other containers. In FTP data transactions, each filesystem container will access where the data is placed and prepare it for sending.

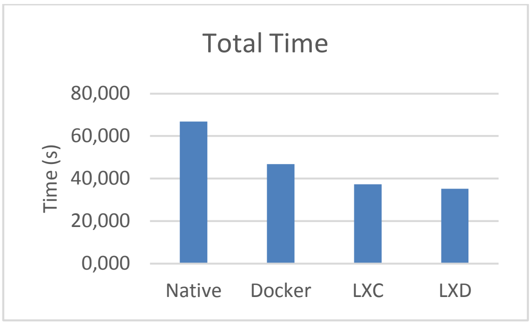


Figure 8. Total time FTP server evaluation result

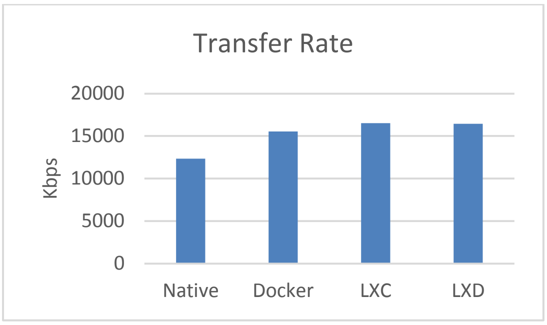


Figure 9. Transfer rate FTP server evaluation result

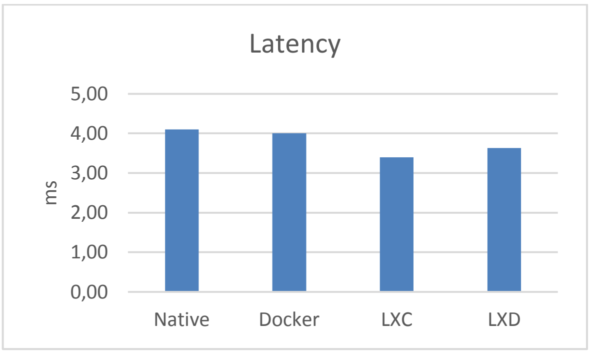


Figure 10. FTP server latency evaluation result

3.2.3. Mail server

Native and container have a very low-performance difference in terms of SMTP protocol evaluation, as shown Figure 11. Overhead of all containers recorded below 3% with LXD as the lowest (Op=1.9%). LXC come up with Op 1.85% and Docker with 2.09%. The evaluation of application server performance is summarized in Table 3.

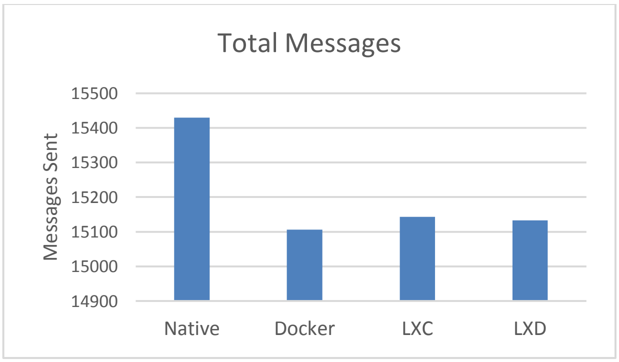


Figure 11. Performance result of mail server SMTP evaluation

Table 3. Summary of application server performance evaluation result

Platform	Web server		Mail server	FTP server		
	Total time	Request per time	Message sent	Total time	Latency	Transfer rate
Docker	Slowest in total time, stable at speed increment	Stagnant when user≥ 4000	Ideal for low to medium load mail server	Slowest among other containers	Low latency	Least in transfer rate
LXC	Fast for user ≥3000	Efficient in serving maximum request when user≥3000	Lowest overhead	Very likely to LXD	Low latency, lowest overhead	The fastest among other containers
LXD	Effcient for user≤2000	Efficient in serving maximum request when user ≤2000	Very likely to LXC	The fastest among other containers	Low latency	Very likely to LXC

4. CONCLUSION

There is a difference in system performance between containers and native due to the overhead of virtualization. However, performance overhead in container is considered small and in several cases almost not exists. Overall performance results in this study indicate that LXD has the most superior value from Docker and LXD with a percentage of 90.5%. These results indicate that LXD as an increase of LXC can improve overall Linux container system performance. LXD has patched up its system and has the best result in terms of system performance, proven that LXD can increase overall performance effectivity.

When compared with native system performance, all parameters tested in the overall performance metric show that native is still superior. In CPU intensive parameters, the performance difference between native and container is only 0.137% adrift. In other parameters, such as IOzone read, IOzone write, and RAM Speed test, found differences in performance ranging from 2.067% (at RAM Speed) and 20% (on the IOzone test). This performance difference is due to the isolation of cgroups so that the processes that occur in the container do not mix with the host system.

REFERENCES

[1] B. Wang, Y. Song, X. Cui, and J. Cao, "Performance comparison between hypervisor- and container-based virtualizations for cloud users," *2017 4th Int. Conf. Syst. Informatics, ICSAI 2017*, pp. 684-689, 2017.

[2] R. Morabito, J. Kjällman, and M. Komu, "Hypervisors vs. lightweight virtualization: A performance comparison," *Proc. - 2015 IEEE Int. Conf. Cloud Eng., IC2E 2015*, pp. 386-393, 2015.

[3] A. C. Baktir, Y. C. Kulahoglu, O. Erbay, and B. Metin, "Server virtualization in information and communication technology infrastructure in Turkey," *2013 21st Telecommunications Forum Telfor (TELFOR),* pp. 13-16, 2013.

[4] Y. Tachibana, J. Kon, and S. Yamaguchi, "A study on the performance of web applications based on RoR in a highly consolidated server with container-based virtualization," *2017 Fifth International Symposium on Computing and Networking (CANDAR),* Aomori, pp. 580-583, 2017.

[5] J. Nie, "A study on the application cost of server virtualisation," *2013 Ninth International Conference on Computational Intelligence and Security*, Leshan, pp. 807-811, 2013.

[6] A. Abuabdo and Z. A. Al-Sharif, "Virtualization vs. containerization: Towards a multithreaded performance evaluation approach," *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA),* pp. 1-6, 2019.

[7] R. Madhumathi, "The relevance of container monitoring towards container intelligence," *2018 9th International Conference on Computing, Communication and Networking Technologies (ICCCNT),* Bangalore, pp. 1-5, 2018.

[8] R. Dua, A. R. Raja, and D. Kakadia, "Virtualization vs containerization to support PaaS," *Proc. - 2014 IEEE Int. Conf. Cloud Eng. IC2E 2014*, pp. 610-614, 2014.

[9] M. I. Djomi, R. Munadi, and R. M. Negara, "Analysis of FTP service performance and video streaming based on network function virtualization using Docker containers," in Bahasa "Analisis performansi layanan FTP dan video streaming berbasis network function virtualization menggunakan Docker containers," *ELKOMIKA*, vol. 6, no. 2, pp. 180-193, 2018.

[10] R. F. Aswariza, D. Perdana, and R. M. Negara, "Analysis of VyOS virtualized network function throughput and scalability on VMWare ESXi, XEN, and KVM hypervisors," in Bahasa "Analisis throughput dan skalabilitas virtualized network function VyOS pada hypervisor VMWare ESXi, XEN, dan KVM," *JURNAL INFOTEL,* vol. 9, no. 1, SI, pp. 70-74, 2017.

[11] A. Babu, M. J. Hareesh, J. P. Martin, S. Cherian, and Y. Sastri, "System performance evaluation of para virtualization, container virtualization, and full virtualization using Xen, OpenVZ, and XenServer," *2014 Fourth International Conference on Advances in Computing and Communications*, Cochin, pp. 247-250, 2014.

[12] L. Ma, Y. Chen, Y. Sun, and Q. Wu, "Virtualization maturity reference model for green software," *2012 International Conference on Control Engineering and Communication Technology*, Liaoning, pp. 573-576, 2012.

[13] N. Jain and S. Choudhary, "Overview of virtualization in cloud computing," *2016 Symposium on Colossal Data Analysis and Networking (CDAN),* Indore, pp. 1-4, 2016.

[14] B. Lakshmikanth and M. R. Mundada, "Automation framework development for continuous integration and deployment in CT machines using LXC and Docker container lightweight virtualization techniques," *2018 International Conference on Current Trends towards Converging Technologies (ICCTCT)*, pp. 1-4, 2018.
[15] B. Golden, "Virtualization for dummies," John Wiley & Son Publishing Inc., 2011.
[16] A. A. Mohallel, J. M. Bass, and A. Dehghantaha, "Experimenting with docker: Linux container and baseos attack surfaces," *2016 Int. Conf. Inf. Soc., (i-Society)*, pp. 17-21, 2016.
[17] A. Blenk, A. Basta, M. Reisslein, and W. Kellerer, "Survey on network virtualization hypervisors for software defined networking," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 1, pp. 655-685, 2015.
[18] R. K. Barik, R. K. Lenka, K. R. Rao, and D. Ghose, "Performance analysis of virtual machines and containers in cloud computing," *2016 Int. Conf. Comput. Commun. Autom., (ICCCA)*, pp. 1204-1210, 2017.
[19] A. Sampathkumar, "Virtualizing Intelligent River ® : A comparative study of alternative virtualization technologies," Thesis, Clemson University, 2013.
[20] D. Bernstein, "Containers and cloud: From LXC to Docker to kubernetes," *IEEE Cloud Comput.*, vol. 1, no. 3, pp. 81-84, 2014.
[21] S. Gupta and D. Gera, "A comaparison of LXD, Docker and virtual machine," *Int. J. Sci. Eng. Res.*, vol. 7, no. 9, pp. 1414-1417, 2016.
[22] J. Claassen, R. Koning, and P. Grosso, "Linux containers networking: Performance and scalability of kernel modules," *Proc. NOMS 2016 - 2016 IEEE/IFIP Netw. Oper. Manag. Symp.*, no. Noms, pp. 713-717, 2016.
[23] Á. Kovács, "Comparison of different Linux containers," *2017 40th Int. Conf. Telecommun. Signal Process., (TSP)*, pp. 47-51, 2017.
[24] E. Casalicchio and V. Perciballi, "Measuring Docker performance : What a mess !!!," *Int. Works. on Autono. Control for Perfor. and Reabil. Trade-offs in Internet of Services, (ACM ICPE 2017 Companion)*, pp. 11-16, 2017.
[25] Z. Li, M. Kihl, Q. Lu, and J. A. Andersson, "Performance overhead comparison between hypervisor and container based virtualization," *2017 IEEE 31st Int. Conf. Adv. Inf. Netw. Appl., (AINA)*, pp. 955-962, 2017.

BIOGRAPHIES OF AUTHORS

Adinda Riztia Putri received her BSc degrees in Telecommunication Engineering, from the Telkom University, Indonesia in 2018, respectively. Her interests include Software Defined Networks, and Virtualization Network Engineering.

Rendy Munadi received his Doctor in Telecommunication Engineering from Indonesia University. He is current research in the area of Next Generation Network, Wireless Sensor Network, Routing Management and Virtualization Network.

Ridha Muldina Negara received her BSc and MSc degrees in Telecommunication Engineering, from the Institute of Technology Telkom, Indonesia in 2009 and 2013, respectively. Her interests include Software Defined Networks, Cyber Security, Telecommunication Systems and Computer Engineering.