Performance Comparison of Docker and LXD with ApacheBench

Y A Auliya, Y Nurdinsyah and D A R Wulandari
Network and Security Research Group, Faculty of Computer Science, University of Jember

E-Mail: yudha.alif@unej.ac.id

Abstract. Container is a popular approach in computer virtualization. If virtual machine separated operating system resources, container will receive same core resources with main operating system. Because of that, container size is smaller than virtual machine. Docker and LXD are part of a popular container vendor. Even though it has a different way to working, docker and LXD both have advantages. Docker only allows computer application to run in single thread mode, so it can protect hardware resources from process overload. LXD allows application to run in multi thread mode, so the application can run normally like installed in normally operating system. This paper tried to compare the performance of docker and LXD. Web server application installed on the docker and LXD container, then we do a stress test using ApacheBench. ApacheBench send a large number of request package to test the container performance. Better container condition can handle more number of request package.

1. Introduction

Container technology is developing rapidly, especially for the topic of computer virtualization. The Docker method is a concept that attracts the attention of researchers and technicians. Docker abstracts low-level "unified fire." At present, many projects integrate or use the concept of container. One of them is Google [1], stating that they have developed a tool to manage containers in large numbers. The operating system is optimized to run containers on a larger physical cluster, called Datacenter Operating Systems (DCOS). Many similar projects discuss networking, packaging, scheduling and various aspects of the container. Container development leads to the development of ecosystems of container technology.

One of the main technologies that permit this idea is virtualization. Virtualization facilitates to create contained environment to isolate the computing loads from the underlying resources. This offers a separation of concern between host machine owners and guest machine owners that run the workloads.

Containers are based on a set of kernel features of Unix and Linux systems. The two most critical among these features are logical separation and isolation of process execution and resource limitations for processes. In Linux kernels, the respective capabilities are realized by implementations called namespaces and control groups, which in turn find adoption by various higher-level implementations, for example Linux Container (LXC) [2] and Docker [3].

In previous studies, comparisons between virtual machines and Linux containers have been carried out which results in better linux containers in several aspects [4]. virtual machine separated operating system resources, container received same core resources with main operating system. Because of that, container size is smaller than virtual machine. Docker and LXD is part of a popular container vendor. Even though it has a different way to working, docker and LXD both have advantages. Docker only allows computer application to running in single thread mode, so it can be protected hardware resources from process overload. LXD allows application to running in multi thread mode, so the application can running normally like installed in normally operating system. This paper compare performance of docker and LXD. Web server application installed on the docker and LXD container, then we do a stress test using ApacheBench. ApacheBench send a large number of request package to test the container performance. The request package is sent with a trial scenario of 1000-50000 data packages.
The test method uses a large number of data packet delivery scenarios. Data packets are sent simultaneously to determine the reliability of the architecture used. The parameters used are the number of requests, failed requests, transfer rates and time per request. In this case study the three machines were run using different architectures, the first engine uses the LXD concept, the second engine uses the docker container concept and the third engine is used as a test engine.

2. Docker Container

Container based virtualization provides the ability to create containers which, acts like a lightweight OS. It works at OS level and creates isolated user instances by partitioning the physical machine [5]. Each container shares the kernel and hosted OS. With the recent advancement in virtualization technologies, few containers provide near native performance. LXC, Docker, Open VZ are examples of few mainstream container virtualization technologies. The of container based virtualization is presented in Figure 1.

![Figure 1. Container based virtualization](image)

Application developers prefer containers as they are highly portable, scalable, and reduces cost of resource requirements when compared with VMs. Containers reduce the time for deployment and have well-segregated management [7].

3. LXD

LXD is a daemon which facilitates a REST API to manage the container. LXD runs on top of the LXC. LXD provides a simple command line user experience for easily managing the container by reducing the complexity. The containers in LXD are designed securely by default and works through image based instead of distribution templates. Live migration of LXC is supported by LXD with proper CRIU dependencies [6]. LXD is not similar to Docker, which is an application container manager whereas LXD is system container manager. LXD provides better user experience by addressing the shortcomings of LXC like live migration and restrictions of dynamic resources. In the present study, LXD is used to perform efficient live migration of containers.

4. Test Methodology and Environment

The methodology used is to make 3 machines using different architectures. The first machine was run using the docker container concept to access the apache server, the second machine was run using the LXD concept to access the apache server. Machine 3 is used to test both machines. Testing is done using ApacheBench by doing a stress test. The first machine and the second machine are assigned to access the apache server page repeatedly. While the third machine is installed ApacheBench to measure reliability from the first machine and the second machine. Stress tests are carried out by sending data packets of 1000 - 50000 data packets. Each shipment is sent 100 data packages. The parameters tested are the number of requests, failed requests, transfer rates and time per request. This study wants to know the architecture that is better at sending data packets and requires the shortest
The scenario for using 3 machines that are run using the docker and LXD architecture can be seen in Figure 2.

![Diagram of container architecture scenario]

**Figure 2.** Container architecture scenario

5. Result and Analysis

Tests are carried out using 3 different architectures and tested with a request of 50000 times. Sending packet requests is done periodically from 1000 packet requests to 50000 packet requests with an internal 5000 packet request. Packet requests are sent in stages to determine the maximum capability of the machine using the docker container and LXD concepts. The test will compare the number of failed requests, transfer rates and time per second that are run on the docker container and LCD architecture. The first parameter is failed request. The failed request parameter describes the number of packet requests that did not arrive at the apache web server. The comparison details of failed request parameters can be seen in Figure 3.

![Comparison details of failed request parameter graph]

**Figure 3.** Comparison details of failed request parameter

Based on the results in figure 3 it can be concluded that the docker and LXD methods have advantages and disadvantages. When a small number of packet requests is carried out between 1000 and 10000 request packages, the LXD method and docker get a slight failed request. Both of them get failed requests around 1000 packets. When the number of requests is added between 15,000 and 25,000 the docker method obtains a number of failed requests that are much smaller than LXD. Packet requests continues with an interval of 5000. When the number of requests reaches 35000 to 45000 both methods experience a significant increase in the number of failed requests to 40000 packets. When the number of packet requests reaches 50000 both methods get a failed request that is slightly in the
range of 5000 packets. Based on the failed request test results it can be concluded that the docker method is better than LXD when there are moderate data requests between 35000 and 45000 packets. In the trial using the large data LXD method and docker get the same good results.

The second parameter is transfer rate. The transfer rate parameter describes the average speed obtained to complete package delivery. The comparison details of failed request parameters can be seen in Figure 4.

![Figure 4. Comparison details of transfer rate](image)

Based on the results in Figure 4 it can be concluded that the docker method and the LXD method have advantages and disadvantages. When a small number of packet requests is made for 1000 packet requests, the LXD method gets a smaller transfer rate than the docker. testing is done by sending a larger number of packet requests and the docker method results in a smaller transfer rate than LXD. Based on the results of the transfer rate parameter test, it can be concluded that the docker method is more reliable than LXD when the number of requests is small but when the number of requests is getting more, the LXD method gets a better transfer rate.

The third parameter is time per request. parameter time per request describes the average time needed to complete package delivery. The comparison details of failed request parameters can be seen in Figure 5.

![Figure 5. Comparison details of time per request](image)
Based on the results in Figure 5 it can be concluded that the docker method is more reliable than the LXD method. Testing is done by sending an increasingly large number of packet requests. The test results show that the docker method obtains a time per request greater than LXD.

Based on the test results using the docker method obtained an average failed request of 8765 packets. The test results using the LXD Method obtained an average failed request of 10913. While the parameter time per request docker method obtained an average of 73688 and the LXD method of 59833.

6. Conclusion
Based on the results of testing conducted by creating 3 virtual machines that are run using a different container architecture docker container, LXD and testing machine. Based on the results of testing carried out using 3 parameters it can be concluded that the docker and LXD methods have advantages and disadvantages. Docker container method has better reliability than LXD for large packet requests. When referring to the parameter transfer rate and time per request the LXD method is better than the docker container because it has a larger transfer rate and has less time per request than the docker container. In the next study, different container architectures can be developed so that they are more reliable in accepting large number of packet requests and having a shorter transfer rate and time per request.

Acknowledgments
The author would like to thank the University of Jember institutions and all the lecturers who helped carry out this research. Do not forget we thank the lecturers and employees in the NS research group.

References
[1] Marmol V, Jnagal R and Hockin T 2015 Networking in Containers and Containers Cluster (Canada: Proceedings of netdev 0.1)
[2] Qiu Y, Lung C, Ajila S and Srivastava P 2017 LXC Container Migration in Cloudlets Under Multipath TCP (Turin: IEEE Annual Computer Software and Application Conference) pp 31-36
[3] Joy A M 2015 Performance comparison between linux containers and virtual machines (Computer Engineering and Applications, International Conference on Advances) pp 342–346.
[4] Felter W, Ferreira A, Rajamony R and Rubio J 2015 An Updated Performance Comparison of Virtual Machines and Linux Containers (Philadelphia: IEEE International Symposium on Performance Analysis of System and Software ISPASS) pp 171-172
[5] Dua R, Raja A R and Kakadia D 2014 Virtualization vs Containerization to Support PaaS (Boston: IEEE International Conference on Cloud Engineering) pp 610-614
[6] Xavier M G, Oliviera I C D, Rossi F D, Passos R D D, Matteussi K J and Rose C A F D 2015 Performance Isolation Analysis of Disk-Intensive Workloads on Container-Based Cloud (Turku: Euromicro International Conference on Parallel, Distributed and Network-Based Processing) pp 253-260
[7] Adufu T, Choi J and Kim Y 2015 Is Container-based Technology a Winner for High Performance Scientific Applications? (Busan: Asia-Pacific Network Operations and Management Symposium) pp 507-510