Bulletin of Electrical Engineering and Informatics
Vol. 10, No. 2, April 2021, pp. 837~847
ISSN: 2302-9285, DOI: 10.11591/eei.v10i2.2742

Live migration using checkpoint and restore in userspace (CRIU): Usage analysis of network, memory and CPU

Adityas Widjajarto, Deden Witarsyah Jacob, Muharman Lubis
School of System and Industrial Engineering, Telkom University, Indonesia

Article Info

Article history:

Received Oct 25, 2020
Revised Jan 17, 2021
Accepted Feb 15, 2021

Keywords:

Container
CRIU
Docker
Live Migration

ABSTRACT

Currently cloud service providers have used a variety of operational mechanism to support the company's business processes. Therefore, the services are stored on the company's server, which present in the form of infrastructure, platform, software and function. There are several vulnerabilities have been faced in the implementation such as system failure, natural disasters, human errors or attacks from unauthorized parties. In addition, the time of unavailability of services can be minimized by doing a LM, which many servers have been used the containers to behave like a service provider. Actually, its existence replaces the virtual machine that requires more resources although the process only can be done through Docker with checkpoint and restore in userspace (CRIU). In this research, LM processes are done to the Docker container using CRIU by analyzing the quality of service (QoS), memory and CPU usage. Thus, the simulation are carried out by establishing the LM using 2 (two) different platforms through scenarios with one and three containers respectively. The performance analysis results aim to examine several indicators in comparison with the achievable result to reduce problem occurred in the cloud service.

Corresponding Author:

Adityas Widjajarto
School of System and Industrial Engineering, Telkom University
Jalan Telekomunikasi, Dayehkolot, Bandung, Indonesia
Email: adtwjrt@telkomuniversity.ac.id

1. INTRODUCTION

Containerization or container-based virtualization has been known since the late 60s, which allows platforms to provide more efficient services compared to the virtual machines (VM). It has begun to shift the position of the traditional VM to perform system maintenance, fault management and load balancing while providing end users with anticipating costly service downtimes and perceptible interruption in the service [1]. In fact, live migration (LM) across the cloud service can improve the provision of resources aggregation from a single data center to multiple disparate data centers geographically. Meanwhile, it is also possible to execute the process through ad-hoc solutions utilizing network file systems or replication of private storage arrays or even proprietary block devices with software used in coordination with more popular memory migration methods. Therefore, this loose set of mechanisms makes the relay structures become more complex, inflexible and unreliable, which its performance is extremely poor compared to the traditional LAN (local area network) migration. In addition, the platform must be online at all times to provide robust service to every clients while shutdown and restart for maintenance purposes should be organized accordingly to prevent data corruption or loss. Individual or even company prefer service provider platforms through paid service for cloud compared to develop own servers due to the practicality and manageability [2]. Thus, the use of containers that are lightweight technology is proven to be helpful and supportive for application

management in the cloud although several copying memory on remote host task cause time frozen [3]. On the other hand, the container implementation can be very useful in the cloud to accommodate the application services be migrated by using Docker [4], which is very risky because it requires a lot of consideration such as time and cost.

There are roughly three methods of direct migration of VMs, which are pre-copy, post-copy, and hybrid method, where direct migration of VMs is mainly done by pre-copy or the variation method [5]. In short, LM has been defined as a solution to reduce platform downtime when maintenance is carried out by moving containers to the backup platform so both source and destination is still online with the monitoring process can be done simultaneously [6]. However, it is also have the ability to freeze the running applications, record the memory content according to the current state of the application, moving the record results to the backup platform, restoring the record data from the main platform on the backup platform and keeping the application to run according to the conditions at the checkpoint [7, 8]. In actual sense, it is very useful for maintaining service availability when the platform has to pause at certain moment due to several connection issues. On the other hand, Docker is an open-source project based on Linux containers that provides an open platform for developers and administrators to be able to automatically build, package and run applications as lightweight containers [9] while at the same time allows the user to separate applications from infrastructure to build software quickly [10]. It provides the ability to run applications in an isolated environment because the lightweight nature cause the running process has been done without adding the burden of a hypervisor, so more containers on certain hardware combinations can be done using a virtual machine [11]. In this case, the LM has been established in the real time by moving containers one by one between different physical machines without disconnecting the network connection with the client [12].

This study determined to use Ubuntu 16.04 due to the several reason such as this version is compatible with the logical requirement of Docker and CRIU and the operating system (OS) has been well known for community support in term of troubleshooting and operational purposes. When making a migration, the container will enter frozen time, that is, when the container pauses, the source node performs memory blocks, processes, file systems and network connections to record the state of the container at that time. Then, performance evaluation is carried out based on various criteria according to the research objectives, which in this case related to the usage analysis and the results are also studied to prove the goal. Most data migration projects flock to the main project without considering whether migration is possible, how long it will take, the technology it will need and the risks that lie ahead. A pre-migration impact assessment is recommended to verify the cost and likely outcome of migration. This study simulates the data migration to analysis several aspect of performance using CRIU based on the network bandwidth and throughput, memory and CPU usage.

2. MATERIAL AND METHOD

The container record data is copied to the destination node and will restore from source so it can operate and unfreeze normally [13]. Meanwhile, CRIU is a software tool on the Linux operating system that is able to freeze and make a checkpoint of an application that is running on the platform and also able to restore it to different physical machine with the same conditions as the frozen time [14]. It allows administrators to conduct LM of containers through communication protocol as a set of rules that must be obeyed to communicate between two or more computers. These protocols are needed to ensure that all data communications on each computer can be carried out according to the needs and objectives of the user. The basic functions of the communication protocol include connection, addressing, error control, data flow control and synchronization [15]. Meanwhile, TCP is a protocol used to transmit data with good reliability and has a system of congestion control or control of data transfer bottlenecks [16]. It is a byte-stream protocol that is able to control data flow based on byte numbers, the smallest unit of data transmitted on the Internet is a data segment or packet, each identified by the data octet number. When a destination receives a data segment, the host recognizes that segment acceptance by issuing a packet acknowledgment (ACK) to notify the status of the packet sent by the sending host [17]. On the other hand, UDP is a protocol for transferring data between computer network applications, which is called a connectionless protocol that does not require a connection between hosts before sending or receiving data so that a host can service many other hosts by sending the same data [18]. Therefore, Netlink is a protocol used on Linux operating systems for data communication between kernels and userspace, which has functions as a protocol between Forwarding engine components and control plane components that function to determine IP services [19].

Quality of service (QoS) is the ability of a network to provide good services by providing bandwidth, and overcoming the delay [20]. Delay is the time needed by a packet sent from the sender to the recipient and one of the parameters in QoS [21]. In a network, delay can be a reference to assess network quality with the smaller the delay value produced, the better the network performance. On the other hand,

packet loss is the loss of a number of data packets in the delivery to the destination address [22]. Actually, packet loss on computer networks has a huge effect, where if there is a certain amount of packet loss it will cause TCP interconnection to slow down [23]. Meanwhile, throughput is a value when a service can process a request [24], which is measured when it is finished transmitting data to a host or client. The most important part of throughput is that there is enough bandwidth. Next, random access memory (RAM) is a volatile hardware on a computer where data will be lost when the power on the computer is turned off [25]. In RAM, data can be accessed randomly, which its function is to store temporary data and read data. The central processing unit (CPU) is one of the computer components that composes a computer to function, which has aim to be the brain of a computer for processing inputted data and producing the output [26]. In addition to processing data, the CPU also controls all components of the computer so that it can work well, which it consists of three components; control unit, arithmetic logical unit (ALU) and register [27], which is explained visually in Figure 1. Lastly, process definition is a program that is being executed or running, which its process is the smallest work unit in the operating system. The process passes through a series of states and a variety of events can occur to cause state changes [28].

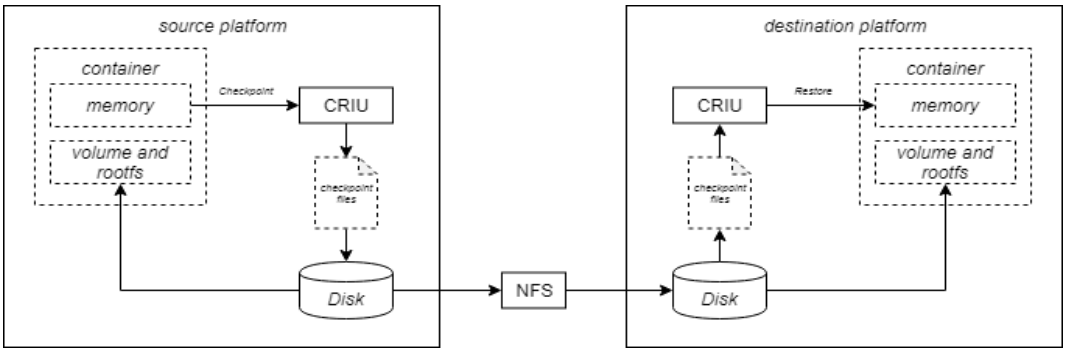


Figure 1. Live migration (LM) mechanism

The testing mechanism in this study is the structure and concept to describe on how the LM process takes place. The container on the platform consists of memory, volume and root stored on the disk. On the source platform, CRIU functions as software for performing memory checkpoints on containers. The results of the checkpoint by CRIU are stored on disk in the form of checkpoint files. NFS serves as a protocol for sending checkpoint files from the source platform to the destination platform. At the destination platform, CRIU functions as a software for restoring containers by using checkpoint files obtained from the source platform. Actually, LM is executed in the four test scenarios in this study.

- First scenario; one-way live migration from platform 1 as the source platform to platform 2 as the destination platform.
- Second scenario; one-way live migration from platform 2 as the source platform to platform 1 as the destination platform.
- Third scenario; two-way live migration with one service.
- Fourth scenario; two-way live migration with three services.

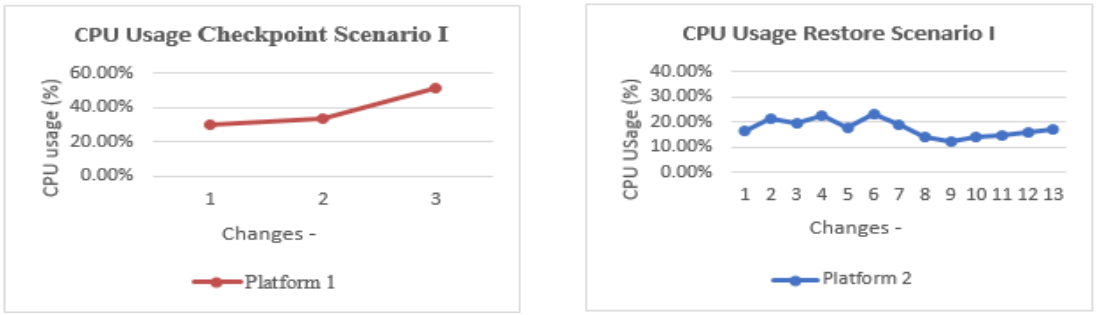
3. RESULTS AND ANALYSIS

System testing is executed by using Docker container and CRIU as the tool to do LM which is run on the Ubuntu Linux as the operating system on two platforms. It is based on a previously designed scenario, namely the scenario of testing LM, which in these tests, four LM scenarios were carried out differently.

3.1. First scenario; one-way live migration from platform 1 to platform 2

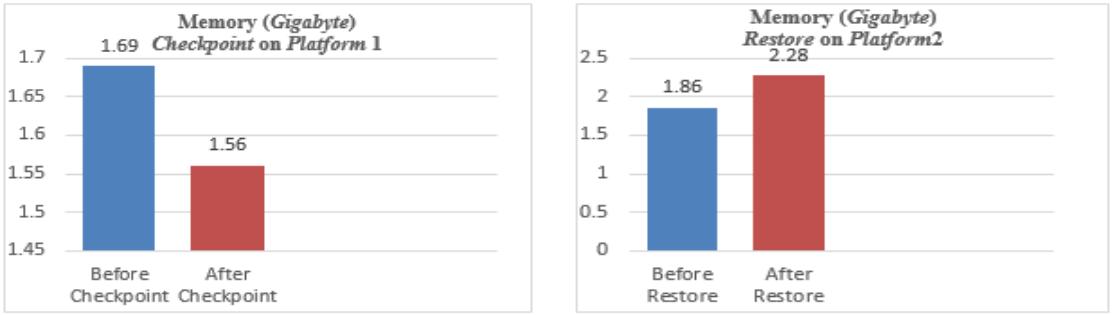
In scenario I the checkpoint process is carried out by platform 1, which lasts for 1,320 s and the platform 2 lasts for 17,963 s. The LM with VM is a very powerful tool for cluster administrators in many major scenarios, such as load balancing, which in actual VMs can be reorganized on physical devices into a group to relieve the load on the busy hosts. It is also can be done for online maintenance and proactive fault tolerance, which sometimes the physical machine may need an upgrade on its service for future system malfunctions. Thus, the administrator must move the running virtual machines to alternate devices to release or freeing the original machine for maintenance by improving serviceability and system availability [8].

From the results of these tests in platform 1 can be obtained a value of 1.69 GB before doing a checkpoint and 1.56 GB after a checkpoint. Meanwhile, in the platform 2 obtained a value of 1.86 GB before doing a checkpoint and 2.28 GB after a checkpoint. The amount of time needed to do a checkpoint takes 1.32 s on platform 1 while restore takes 17.963 s on platform 2. Based on the result, more memory usage were needed within the context of before compare to after checkpoint due to upon completion, the VM return to running state while the old host's VM is deleted which the session will be disconnecting for several seconds before auto-reconnecting. On the other hand, in the process of restore, CRIU transform itself into a target task through orderly steps which are open images to read in VM, fork and pre-mmaps, open file mappings, open shared mappings, dive into restorer context and restore mapping in their places. When doing LM, a lot of time and resources is needed based on procedure to copy task's memory to the destination host as shown in Figures 2-4.



(a) (b)

Figure 2. CPU usage scenario I, (a) Checkpoint, (b) Restore



(a) (b)

Figure 3.Memory usage scenario I, (a) Checkpoint, (b) Restore

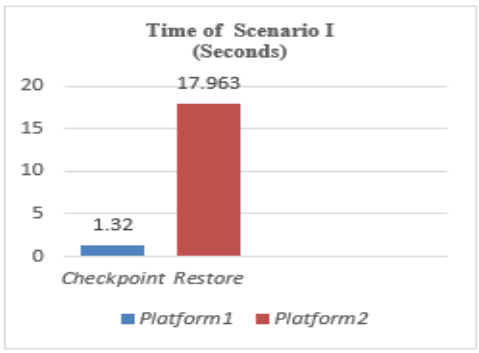
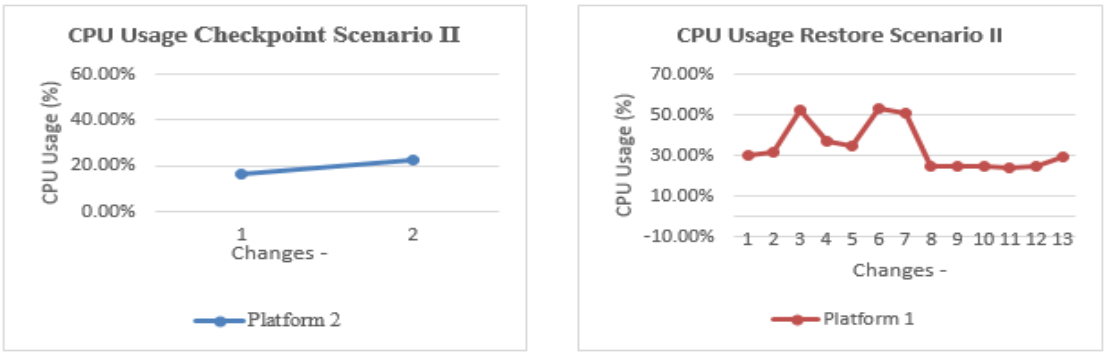


Figure 4.Time usage scenario I

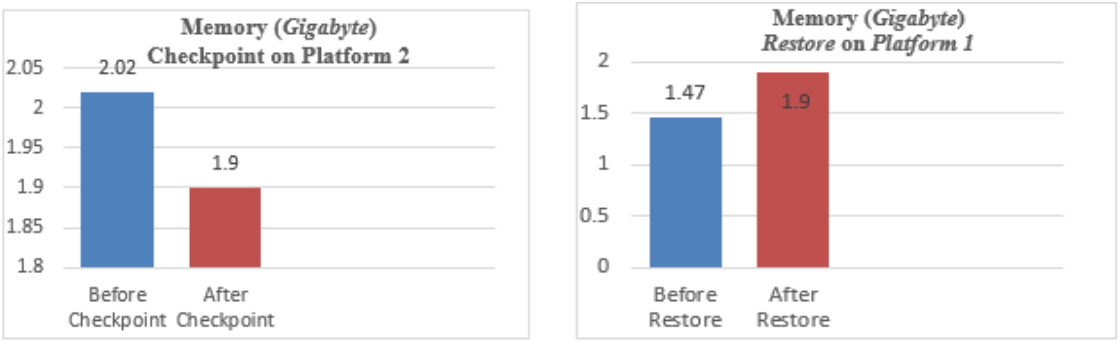
3.2. Second scenario; one-way live migration from platform 2 to platform 1

The checkpoint in scenario II for platform 2 lasts for 1,189 s while platform 1 lasts for 18,091 s. In power management scenario, the load and performance of servers are generally uneven but statistically uniform at different periods. When some of the virtual machines on the distributed hosts run light functions, which can be integrated across fewer hosts, the downloaded hosts can be stopped after the migration is complete. This strategy helps companies reduce IT operating expenses and benefit the natural environment. The performance of source influence significantly the total amount of time needed for the LM to be completed, which in this case in scenario I and II have been indicated with the fluctuative percentage usage in changes cycle. Actually, the number of pages moved is decided on how effective the VM handles and manipulating the memory pages. It will take the longer time for every page to transfer to the destination server due to the number of pages that have been changed or arranged. After the process completed, the destination server create the working set for VM testing, which in exact time with when the migration process has started. From the results of these tests in platform 2, it can be obtained a value of 2.02 GB before doing a checkpoint and 1.90 GB after a checkpoint while platform 1 has a value of 1.47 GB before restoring and 1.90 GB after restore. The amount of time needed to do a checkpoint takes 1,189 s on platform 1 while restore takes 18,091 s on platform 2. In each solution, there are three basic types of cases that must be migrated, which are physical memory for the virtual machine, network connections, and the state of the virtual machine, such as SCSI storage as shown in Figures 5-7. The most difficult problem is the paging of physical memory, as it is the main factor affecting paging downtime, that is, when the services in the virtual machine are not fully available.



(a) (b)

Figure 5. CPU usage scenario II, (a) Checkpoint, (b) Restore



(a) (b)

Figure 6. Memory usage scenario II, (a) Checkpoint, (b) Restore

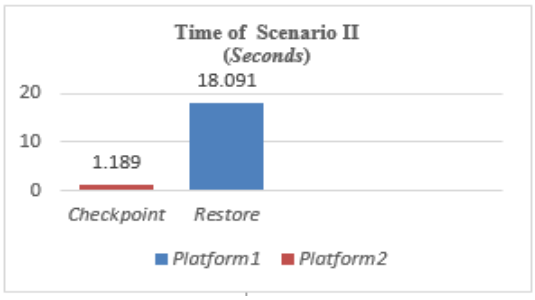


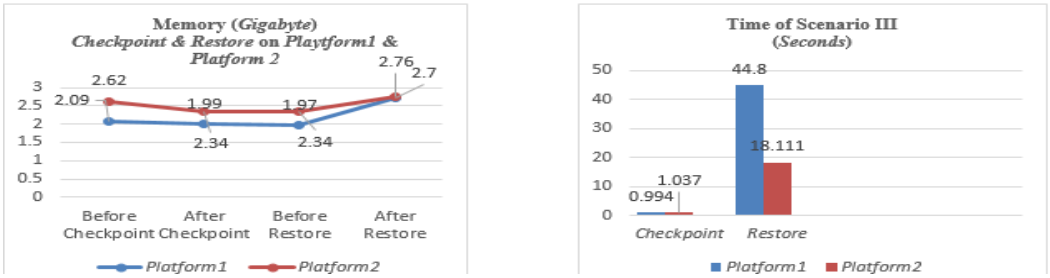
Figure 7. Time usage scenario II

3.3. Third scenario; two-way live migration with one service

In Scenario III the checkpoint process is done by platform 1 lasts for 0.994s and platform 2 for 1.037 s, which the restore process is done by both platforms simultaneously. Restore in scenario III on platform 1 lasts for 44.8 s and on platform 2 for 18.111 s. The result of checkpoint on platform 1 has a value of 2.09 GB before doing a checkpoint and 1.99 GB after checking, 1.97 GB before restoring and 2.70 after restoring. On platform 2, the value is 2.62 GB before checking, 2.34 GB after checking and 2.34 GB before restoring and 2.76 GB after restoring. The amount of time needed to do a checkpoint takes 0.994 s for platform 1 and 1.037 s for platform 2 while restore requires 44.8 s for platform 1 and 18.111 s for platform 2. When the memory page contamination rate is faster than the replication rate for the pre-transcription procedures, all the pre-copy work will be inefficient and one has to stop the virtual machine immediately and copy all the pages from memory to the destination host as shown in Figures 8 and 9. Some strenuous memory workloads will not benefit from a pre-copy algorithm and downtime can increase to several seconds. This limitation makes the algorithm only applicable on high-speed local networks. Second, some semi-virtual optimization systems, such as rogue operations and unassigned page editing, can have some negative impact on the user experience, especially for some responsive interactive services. Finally, the pre-backup algorithm does not restore the data from the CPU cache. Although it may not fail on the target host, massive cache and TLB loss can cause performance degradation once the virtual machine takes over service.

(a) (b)

Figure 8. CPU usage scenario III, (a) Checkpoint, (b) Restore

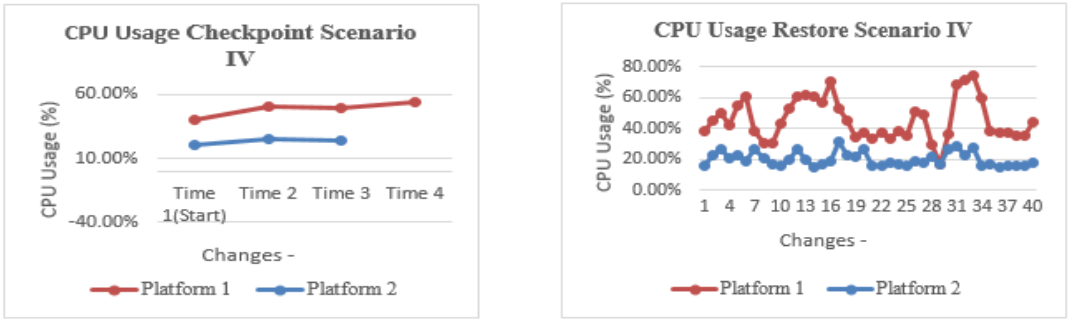


(a) (b)

Figure 9. Memory usage scenario III, (a) Checkpoint, (b) Restore

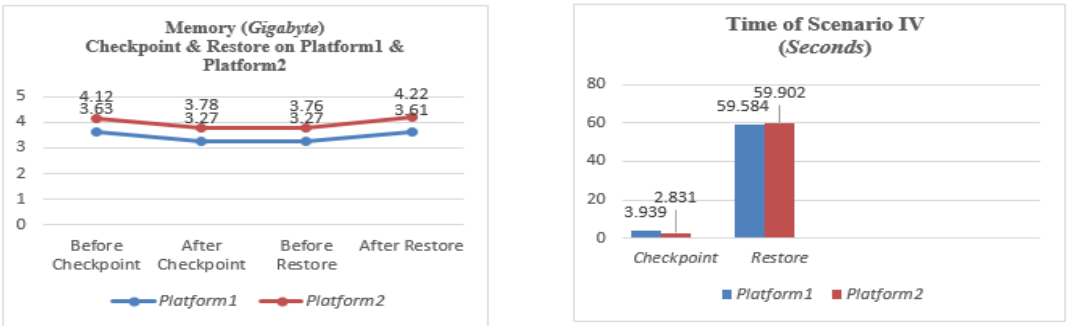
3.4. Fourth scenario; two-way live migration with three services

In scenario IV the checkpoint process is carried out by platform 1 and platform 2, which there are 3 magento services. The checkpoint on platform 1 lasts for 3,939s and on platform 2 for 2,831 s. In scenario IV the restore process is done by platform 1 and platform 2 simultaneously, which the number of containers to be restored is 3 containers. The restore in scenario IV on platform 1 lasts for 59,584 s and on platform 2 for 59,902 s. In general, the running speed can be faster than the original execution with the log, because during normal execution, the process can prevent waiting for input and output events, while all the events can be instantly reproduced during operation due to the ability to bypass the downtime of the instructions. During the migration, physical memory pages are sent across the network to the new destination while the source host continues to operate. Pages modified during replication should be reintroduced to ensure consistency. After a limited iterative repeat phase, a very short stop and copy phase is implemented to move the remaining dirty pages. Figure 10 is the result of checkpoint and restores testing on platform 1 and platform 2. From the test results, it can be obtained a value of 3.63 GB before performing a checkpoint, 3.27 GB after checking, 3.27 GB before restoring and 3.61 GB after restoring on platform 1. On platform 2, 4.12 GB is obtained before checking, 3.78 GB after checking, 3.76 GB before restoring and 4.22 GB after restoring. The amount of time needed to do a checkpoint and restore. When checking, it takes 3,939 s for platform 1 and 1,831 s for platform 2 while for restore takes 59,584 s for platform 1 and 59,902 s for platform 2 as shown in Figure 11. In a LAN environment, since the migrated virtual machine maintains the same network address as before, any ongoing interactions at the network level will not be disabled.



(a) (b)

Figure 10. CPU usage scenario IV, (b) Checkpoint, (b) Restore



(a) (b)

Figure 11. Memory usage scenario IV: (a) Checkpoint, (b) Restore

Table 1 is a table that contains the difference in the number of files accessed by the process in scenario I, II, III, IV, which can be seen that when the checkpoint process there is a reduction in the system process files that are accessed, this is because the service is dead after performing a checkpoint. When the restore process there are additional system process files that are accessed. This is because there are additional containers resulting from the restore process.

Table 1. The difference of the accessed file in live migration process

Scenario	Checkpoint/restore	Process name					
		Platform 1			*Platform 2*		
		Dockerd (1305)	Dockerd (1230)	Docker-co (1478)	Dockerd (1403)	Dockerd (1447)	Docker-co (1747)
Scenario I	Checkpoint (system process file reduced)	8 file	-	-	-	-	-
	Restore (system process file increased)	-	-	-	10 file	-	-
Scenario II	Checkpoint (system process file reduced)	-	-	-	-	8 file	3 file
	Restore (system process file increased)	-	10 file	3 file	-	-	-
Scenario III	Checkpoint (system process file reduced)	-	9 file	3 file	-	8 file	3 file
	Restore (system process file increased)	-	8 file	3 file	-	8 file	3 file
Scenario IV	Checkpoint (system process file reduced)	-	24 file	9 file	-	24 file	9 file
	Restore (system process file increased)	-	30 file	9 file	-	30 file	9 file

3.5. Network analysis

3.5.1. Quality of service analysis

QoS analysis in the LM process has been carried out, the analysis summary is divided into 2, the analysis results on platform 1 and platform 2. Table 2 shows the delay, packet loss, and throughput on platform 1 based on the test scenarios that have been done. The data is obtained from the results of the analysis that has been done before. Table 3 shows the delay, packet loss, and throughput on platform 1 based on the test scenarios that have been done. The data is obtained from the results of the analysis that has been done before**.** The relatively large volume of storage data, such as the virtual disk for a virtual machine, and the small network bandwidth between data centers make storage data migration a bottleneck for direct VM migration over a WAN. A network file sharing system is implemented between the source and destination data centers to prevent the migration of storage data from generating high disk input and output latency. In fact, this architecture results in the double transfer of the virtual machine storage data during the migration, from the source data center to the shared storage system and from Shared storage system to the target data center. That generates a large amount of network traffic. Each data center uses a local storage system, and storage data migration performance is improved by using a variety of methods, such as deduplication or snapshots [10]. QoS should provide an online assurance to offer a range of measurable service features to end-to-end users and applications in terms of delay, instability, available bandwidth and packet loss.

Table 2. Quality of Service of the live migration process of platform 1

Scenario	Delay	Packet loss	Throughput
1	1 ms	0.01%	0.35 MB/s
2	1 ms	0.03%	9.45 MB/s
3	1 ms	0.08%	9.95 MB/s
4	1 ms	0.59%	8.54 MB/s

Table 3. Quality of Service of the live migration process of platform 2

Scenario	Delay	Packet loss	Throughput
1	1 ms	0.98%	9.48 MB/s
2	1 ms	0.06%	0.34 MB/s
3	1 ms	0.40%	9.95 MB/s
4	1 ms	-0.49%	8.55 MB/s

3.5.2. System process in network analysis

Analysis of the system process is summarized based on the test scenario along with the data obtained. The protocols that are read by LSOF, the protocol used to transfer data from source platform to destination platform only TCP is based on the .pcap output file from Wireshark which shows the percentage of Transmission Control Protocol by 100% on all LM tests. However, each improvement faces a trade-off between the newly introduced overhead and the performance benefits of the posting. For example, the de-duplication feature increases hashed fingerprints of storage data to detect blocks that have already been introduced at the destination site to reduce the migration time. If the de-duplication process cannot find enough duplicate blocks at the destination site, it can extend the total migration time, since the computational burden of canceling the duplicate data exceeds its contribution to the data transfer. Therefore, there is a need to systematically rethink storage data migration to greatly improve performance. In other direction, it also can lead to the disruptions in service and communication overheads but the decisions regarding the time and

the location of transfer, which depend on many aspects, such as user mobility, communication channel characteristics and resource availability. The current cloud-based architecture has resulted in an overall great geographic separation between the user and the infrastructure. In such an arrangement, the end-to-end communication involves several network hops resulting in high latency while the incoming bandwidth to the cloud can also be saturated because it is accessed on a multiple-to-one basis [29]. As a result, service applications must remain relatively close to their end users to ensure low latency and high bandwidth connectivity. Basically, algorithms for decision-making should create equilibrium to these trade-offs, which usually need to anticipate the future service requests with some precision or queued buffer service requests so that they can be served in as many batches as possible after migration [30-33].

Table 4. System processes which uses TCP protocol in live migration

Command	PID	User	FD	Type	Device	Size/Off	Node	Name
Dockerd	1403	root	31u	sock	0.9	0t0	96280	Protocol: TCP
Dockerd	1230	root	31u	sock	0.9	0t0	92456	Protocol: TCP
Dockerd	1230	root	31u	sock	0.9	0t0	128883	Protocol: TCP
Dockerd	1230	root	31u	sock	0.9	0t0	270184	Protocol: TCP
Dockerd	1230	root	41u	sock	0.9	0t0	276728	Protocol: TCP
Dockerd	1230	root	51u	sock	0.9	0t0	281782	Protocol: TCP

4. CONCLUSION

Based on the testing and analysis that has been done on the scenario of Docker LM process using the CRIU, it can be concluded that this test makes it possible to backup multiple services and the platform uses more memory than the platform 1. The amount of RAM on a platform does not affect the speed of the backup process for LM. The number of cores on the CPU affects the amount of CPU usage at checkpoint and restore. The more cores used will be the lower CPU usage. The cause of the increase in CPU usage in the checkpoint process is a process that uses more resources when the checkpoint is running. This can be proven through the absence of additional system process files accessed by the process when performing a checkpoint. The cause of the increase in CPU usage in the restore process is the process of accessing more files when restoring. This can be proven through the addition of files that are accessed by the process when restore. The increased system process file is the result of restore container files. Based on the analysis of the process of LM in the Docker container with the CRIU, the LM process only uses TCP as the transport protocol on the Transport Layer computer network. Other protocols shown by the output of LSOF, namely UDP and NETLINK are protocols used by Docker container to provide services that are run. Future implementations can be done by defining an optional interface between the framework and the application, so that it can determine independently when it is preferable or rejected the migration process, thus improving the overall performance after all.

REFERENCES

[1] L. Shu, "The design and implementation of cloud-scale live migration," *Rutgers University Libraries*, Master Thesis, Graduate School New Brunswick, New Jersey, pp. 1-51, 2014.
[2] I. Foster, Y. Zhao, I. Raicu, and S. Lu, "Cloud computing and grid computing 360-degree compared," *2008 Grid Computing Environments Workshop*, Austin, TX, pp. 1-10, 2008.
[3] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud container technologies: a state-of-the-art review," in *IEEE Transactions on Cloud Computing*, vol. 7, no. 3, pp. 677-692, 1 July-Sept 2019.
[4] M. Abdelbaky, J. Diaz-Montes, M. Parashar, M. Unuvar, and M. Steinder, "Docker containers across multiple clouds and data centers," *2015 IEEE/ACM 8th International Conference on Utility and Cloud Computing (UCC)*, Limassol, pp. 368-371, 2015.
[5] D. S. Linthicum, "Moving to autonomous and self-migrating containers for cloud applications," in *IEEE Cloud Computing*, vol. 3, no. 6, pp. 6-9, Nov-Dec 2016.
[6] A. Strunk, "Costs of virtual machine live migration: A Survey," *2012 IEEE Eighth World Congress on Services*, Honolulu, HI, pp. 323-329, 2012.
[7] A. J. Elmore, S. Das, D. Agrawal, and A.E. Abbadi, "Zephyr: Live migration in shared nothing databases for elastic cloud platforms categories and subject descriptors," *Sigmod*, pp. 301-312, 2011.
[8] H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, "Live migration of virtual machine based on full system trace and replay," *ACM HPDC*, pp. 101-110, 2009.
[9] B. Kavitha and P. Varalakshmi, "Performance analysis of virtual machines and docker containers," *Data Science Analytics and Applications*, pp. 99-113, 2017.
[10] F. Zhang, "Challenges and new solutions for live migration of virtual machines in cloud computing environments," *Doctoral Thesis Georg-August University School of Science (Gottingen*), pp. 1-69, 2018.

[11] Z. Kozhirbayev and R. O. Sinnott, "Performance comparison of container-based technologies for Cloud," *Future Generation Computer System*, vol. 68, pp. 175-182, 2017.
[12] F. Zhang, G. Liu, X. Fu and R. Yahyapour. "A survey on virtual machine migration: challenges, techniques and open issues," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 2, pp. 1206-1243, 2018.
[13] R. Morabito, "Virtualization on Internet of Things edge devices with container technologies: A performance evaluation," in *IEEE Access*, vol. 5, pp. 8835-8850, 2017.
[14] P. Karhula, J. Janak, and H. Schulzrinne, "Checkpointing and migration of IoT edge functions," *EdgeSys '19: Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking*, pp. 60-65, 2019.
[15] F. Aïssaoui, G. Cooperman, T. Monteil, and S. Tazi, "Smart scene management for IoT-based constrained devices using checkpointing," *2016 IEEE 15th International Symposium on Network Computing and Applications (NCA)*, Cambridge, MA, pp. 170-174, 2016.
[16] R. Amrutha and V. Nithya, "Curbing of TCP incast in data center networks," *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO) (Trends and Future Directions)*, Noida, pp. 1-5, 2015.
[17] K-C. Leung, V. O. Li and D. Yang, "An overview of packet reordering in transmission control protocol (TCP): problems, solutions, and challenges," in *IEEE Transactions on Parallel and Distributed Systems*, vol. 18, no. 4, pp. 522-535, April 2007.
[18] T. Li, W. Zhu, J. Xu, and Y. Cheng, "The analysis and implementation of UDP-based cross-platform data transmission," *2012 2nd International Conference on Consumer Electronics, Communications and Networks (CECNet)*, Yichang, pp. 628-630, 2012.
[19] S. R. U. Kakakhel, L. Mukkala, T. Westerlund, and J. Plosila, "Virtualization at the network edge: A technology perspective," *2018 Third International Conference on Fog and Mobile Edge Computing (FMEC)*, Barcelona, pp. 87-92, 2018.
[20] C. Cicconetti, L. Lenzini, E. Mingozzi, and C. Eklund, "Quality of service support in IEEE 802.16 networks," in *IEEE Network*, vol. 20, no. 2, pp. 50-55, March-April 2006.
[21] H. Li and W. Zhang, "QoS routing in smart grid," *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, Miami, FL, pp. 1-6, 2010.
[22] D. Chen and P. K. Varshney, "QoS support in wireless sensor networks: A survey," *Proceedings of the International Conference on Wireless Networks*, (ICWN '04), Las Vegas, pp. 227-233, 2004.
[23] K. Chandran, S. Raghunathan, S. Venkatesan, and R. Prakash, "A feedback-based scheme for improving TCP performance in ad hoc wireless networks," in *IEEE Personal Communications*, vol. 8, no. 1, pp. 34-39, Feb 2001.
[24] D. A. Menasce, "QoS issues in Web services," in *IEEE Internet Computing*, vol. 6, no. 6, pp. 72-75, Nov-Dec, 2002.
[25] E. E. Hassan, T. K. A. Z. Zakaria, N. Bahaman, and M.H. Jifri,"Maximum loadability enhancement with a hybrid optimization method," *Bulletin of Electrical Engineering and Informatics*, vol. 7, no. 3, pp. 323-330, 2018.
[26] N. M. N. Mathivanan, N. A. M. Ghani, and R. M. Janor, "Improving classification accuracy using clustering technique," *Bulletin of Electrical Engineering and Informatics*, vol. 7, no. 3, pp. 465-470, 2018.
[27] R. Fauzi, M. Hariadi, M. Lubis, and S. M. S. Nugroho, "Defense behavior of real time strategy games: Comparison between HFSM and FSM," *Indonesia Journal of Electrical Engineering and Computer Science*, vol. 13, no. 2, pp. 634-642, 2019.
[28] Julham, H. A. Adam, A. R. Lubis, and M. Lubis, "Development of Soil Moisture Measurement with Wireless Sensor Web-Based Concept," *Indonesia Journal of Electrical Engineering and Computer Science*, vol. 13, no. 2, pp. 514-520, 2019.
[29] A. Machen, S. Wang, K. K. Leung, B. J. Ko, and T. Salonidis, "Live service migration in mobile edge clouds," in *IEEE Wireless Communications*, vol. 25, no. 1, pp. 140-147, February 2018.
[30] A. Machen, S. Wang, K.K. Leung, B. J. Ko, and T. Salonidis, "Poster: Migrating running application across mobile edge clouds," *MobiCom '16: Proceedings of the 22nd Annual International Conferemce on Mobile Computing and Networking*, pp. 435-436, 2016.
[31] J. Sinti, F. Jiffry and M. Aiash, "Investigating the impact of live migration on the network infrastructure in enterprise environments," *2014 28th International Conference on Advanced Information Networking and Applications Workshops*, Victoria, BC, pp. 154-159, 2014.
[32] J. Fesl, V. Gokhale, M. Dolezalova, J. Cehak and J. Janecek, "Cloud infrastructures protection technique based on virtual machines live migration," *Proc. ACIT*, 2019.
[33] A. Riaz, H. F. Ahmad, A.K. Kiani, J. Qadir, R.U. Rasool, and U. Younis "Intrusion detection systems in cloud computing: A contemporary review of techniques and solutions," *Journal of Information Science and Engineering*, vol. 33, pp. 611-634, 2017.

BIOGRAPHIES OF AUTHORS

Adityas Widjajarto received his Bachelor and Master degree from Institut Teknologi Bandung at 1999 and 2007 respectively. He joined as a Lecturer in the School of Industrial Engineering, Telkom University, in 2013. He teaches some of subjects related to networking such as basic operation system, network service management, information system security and computer forensics and cyber security. His research interests include simulation and model, business intelligence, data communication and networking.

Deden Witarsyah Jacob received his bachelor from Universitas Bhayangkara Surabaya in Electrical and Information Engineering at 1997 while his master in Curtin University of Technology Australia in Computer Engineering at 2006. Lastly, he was graduated his Doctor of Philosophy from University Tun Hussein Onn at 2018. He joined as a Lecturer in the School of Industrial Engineering, Telkom University, in 2013. He teaches some of subjects related to cyberlaw and ethics such as introduction to information system, profession ethics, regulation and international culture as well as research methodology and academic works writing. His research focused on Open Data, Internet of Things and Decision Support System.

Muharman Lubis received his Doctor of Philosophy and Master degree in IT (Information Techology) from International Islamic University Malaysia at 2011 while his Bachelor degree in Information Technology from University Utara Malaysia at 2008. He joined as a Lecturer in the School of Industrial Engineering, Telkom University, in 2017. He taught several subjects related to design and network management such as network design, user experience design and customer relationship management. His research interests include privacy protection, information security awareness, knowledge management and project management.