International Journal of Technology (2017) 4: 611-621
ISSN 2086-9614

A PRELIMINARY STUDY ON SHIFTING FROM VIRTUAL MACHINE TO DOCKER CONTAINER FOR INSILICO DRUG DISCOVERY IN THE CLOUD

Agung Putra Pasaribu[1], Muhammad Fajar Siddiq[1], Muhammad Irfan Fadhila[1], Muhammad H. Hilman[1], Arry Yanuar[2], Heru Suhartanto[1*]

[1]*Faculty of Computer Science, Universitas Indonesia, Kampus UI Depok, Depok 16424, Indonesia*
[2]*Faculty of Pharmacy, Universitas Indonesia, Kampus UI Depok, Depok 16424, Indonesia*

(Received: January 2017 / Revised: April 2017 / Accepted: June 2017)

ABSTRACT

The rapid growth of information technology and internet access has moved many offline activities online. Cloud computing is an easy and inexpensive solution, as supported by virtualization servers that allow easier access to personal computing resources. Unfortunately, current virtualization technology has some major disadvantages that can lead to suboptimal server performance. As a result, some companies have begun to move from virtual machines to containers. While containers are not new technology, their use has increased recently due to the Docker container platform product. Docker's features can provide easier solutions. In this work, insilico drug discovery applications from molecular modelling to virtual screening were tested to run in Docker. The results are very promising, as Docker beat the virtual machine in most tests and reduced the performance gap that exists when using a virtual machine (VirtualBox). The virtual machine placed third in test performance, after the host itself and Docker.

Keywords: Cloud computing; Docker container; Molecular modeling; Virtual screening

1. INTRODUCTION

In recent years, cloud computing has entered the realm of information technology (IT) and has been widely used by the enterprise in support of business activities (Foundation, 2016). With its flexibility, cloud computing is the right choice for service expansion. Prime examples include e-commerce and sharing-based collaboration services such as online document editing and file sharing. The development of cloud computing itself cannot be separated from supporting technologies such as virtualization. Costly investments are required to provide dedicated servers for cloud computing, culminating in the development of virtualization. In short, virtualization allows a single host computer system (host) to be divided into several virtual computers called a virtual machine (VM).

VM success depends heavily on the hypervisor, which plays a vital role in the distribution of computing resources such as processors, RAM, and hard disk to each VM. Another feature of the hypervisor is its ability to isolate each VM so that it looks like the computer itself and does not intersect with another VM, even when it runs on the same host computer system. However, when problems arise, isolation presented by the hypervisor creates havoc in the overall system. Because of this isolation, each VM has one set of operating system (OS) complete with relevant kernel. This becomes a problem when the OS used by the entire VM is based on Linux, given that the Linux kernel can be used in all Linux distributions. A new technology called container

*Corresponding author's email: heru@cs.ui.ac.id, Tel. +62-21-786-3419, Fax. +62-21-786-3415
Permalink/DOI: https://doi.org/10.14716/ijtech.v8i4.7481

has emerged in response to these issues. Similar to the VM, the container can provide insulation to the users. This container technology is not entirely new, having been around since then. Over time, however, more mature and stable container technology has been developed, placing container on par with hypervisor-based VM. The latest data show that about 67% of the world's servers use a hypervisor, while 16% use container. About 49% of the enterprise is considering migrating to container (Foundation, 2016).

In the meantime, many well-developed and maintained applications such as molecular dynamics and virtual screening have been available to users. These applications are widely used in insilico drug discovery processes. However, preparing and maintaining these applications is inconvenient for most users without IT backgrounds. Several approaches were proposed to combat these challenges, including setting up the application in the cloud (Suhartanto et al., 2015; Wibisono & Suhartanto, 2012). We have also identified a need for an application to migrate between computing environments (Suhartanto et al., 2015). Other areas related to this Cloud problems are explained (Aparna & Nair, 2016; Bhushan & Reddy, 2016; Thiyagarajan & Ganesan, 2015).

Research reproducibility is one major issue for most scientific experiments. We will investigate how emerging Docker container technology can respond to our problems and how it performs. This paper presents our system performance benchmark and experiments on molecular modeling and virtual screening that run on Docker container technology. We integrate molecular modeling and docking software (Gromacs, Autodock, and Autodock Vina) into Docker to isolate the environment and thus make it is easier to migrate and reproduce the experiment. Further use of Docker for scientific reproducibility and migration in the cloud computing environment is expected as an alternative technology to the VM.

2. RELATED WORK

In the field of IT operations, container is known as an alternative to the VM. It is a lightweight operating system that runs on the host and executes the instruction directly to the processor core natively, providing insulation and resource management in Linux (Dua et al., 2014). At first glance, the container is no different from the VM. This is because both provide isolation and resource management. The difference is in the host environment. Container only runs on Linux, interacting in a host-guest relationship. Since container only exists in Linux, it is also referred to as Linux Container (LXC).

2.1. Container as a New Virtualization Hype

Container is considered an alternative virtualization technology for the cloud computing environment. Its promising performance will likely make it the backbone of cloud technology. Kozhirbayev and Sinnott (2017) explained that the performance comparison of several container technologies for the cloud environment, including Docker and Flockport, have overthrown native in some respects.

As a virtualization technology, container not only is useful for the cloud computing environment, but also shows promise in the high performance computing (HPC) arena. HPC-based performance benchmarking was done to determine the ability of container to execute Autodock3, a scientific application for virtual screening (Adufu et al., 2015). Container is superior to the VM due to significant differences in start-up times. In Chung et al. (2016), OpenMPI is used to evaluate container performance in the HPC environment. Container can reduce several challenges inherent to using VM.

Reproducibility is one promising benefit of container technology. Scientific experiments often must be repeated to prove their validity. Container stores the entire system in a repository, thus simplifying its reproducibility. Cito and Gall (2016), discussed container's usefulness in providing a reproducible environment for research artifacts in software engineering. This type of advantage will prove valuable in scientific research experiments.

This paper extends the aforementioned research and implements container for insilico drug discovery activities. To the best of our knowledge, insilico drug discovery has not been done by anyone in the Docker container platform.

2.2. Docker Container

Docker is an open-source engine that easily facilitates the deployment of applications into the container (Turnbull, 2014). It uses LXC to implement container solutions and is capable of managing images and files in a system called Union File System (Dua et al., 2014). Developers and sysadmins can build, ship, and run distributed applications (Docker Inc., 2016). Docker can also reduce the complexities encountered in an application's lifecycle by providing a platform equivalent to a part of development, quality assurance, and production in an organization.

Docker requires four major components to achieve its objectives. Turnbull (2014) describes these as client-server, image, registry, and container. In the Docker client and server component, the client program is represented by the "docker" command that can be executed via terminal. Later, the client will contact the server (in this case, the Docker daemon), which functions as both a regulator and a linker to the Docker container. All commands submitted by users via the client will be handled by the server. The second component is the image, which can be described as a template for the Docker container. Users can use the existing image or create their own image. The image-making process in Docker is called build. Later, the image is used as the basis for the container.

The next component is the registry, which functions as a repository where users can store and download Docker images. Docker supports private and public repositories called Docker Hub. Through the "docker pull" command, users can log in and directly download and store the image in the Hub.. The last component is a container, which appears if a user runs a command in an image. One or several processes can run at once insider the container. For example, users can run the command "bash" in the downloaded Ubuntu image. Once the container is automatically established, the user will be in the container and can do anything, including application installation, running a daemon, and system edits.

2.3. Molecular Dynamics

Molecular dynamics (MD) is the study of the dissolution of protein, ligand binding, and the complex interactions of proteins and DNA-protein folding (Mukesh & Rakesh, 2011). It was first used to study interaction on the ball hard (Alder and Wainwright, 1957). Development of the MD field began in 1964 with an MD simulation of fluid ambrgon by Rahman. Ten years later, Rahman and his colleagues performed MD simulations on the state of realistic systems (Stillinger & Rahman, 1974). The basic process of a M is determined by calculating the total force issued for several N atoms. The simulation then works to determine the acceleration owned by each atom and calculate the speed of each atom. Then, after the interaction between atoms, displacement occurs that moves every atom to a new position resulting from molecular simulations. The basic process consists of initialization (interaction and initialization of position), force calculation of atoms (atom position updates), and atom velocity calculation. Among other implementations, MD simulation helps researchers develop drugs for treating certain diseases (Mukesh & Rakesh, 2011).

Gromacs (short for Chemical Groningen Machine Simulation) is an MD application developed by the University of Groningen (Netherlands). The Gromacs application simulates the movement of molecules by using a Newton equation for a system containing hundreds of millions of molecules (Gromacs Manual 4.5.4, 2011). The simulation steps in Gromacs include pdb2gmx, editconf, genbox, genion, energy minimization, and production simulation. The first step, pdb2gm, is the process of building topology that contains information about the types, charges, bonds, and other characteristics obtained from the atomic coordinates. The next step, editconf, determines the model of the solution to be incorporated into the protein to be simulated, adjusts the relative distance between the edges of the box, and identifies the space between the molecules. The third step, genbox, is the process of adding the solution. Genion is the process of adding ions that form a neutral protein solution. The next step, energy minimization, is the process of eliminating local strains that can cause the Van der Walls effect of molecular structures resulting from the previous step. The final step is the production simulation, which is the process of running a reasonably balanced solution obtained within a certain period.

2.4. Molecular Docking

Molecular docking is a computational process used to search for the most suitable ligand in both geometry and energy when it is bound to a known receptor (protein). Molecular docking performed on a ligand and receptor will justify whether the ligand is a drug candidate that is being sought (Mukesh & Rakesh, 2011). The main aspects of molecular docking are computation interaction energy and conformation using a method from quantum mechanics to an empirical energy function. At a glance, problems in molecular docking can be lock-and-key issues, where a protein reception can be considered the lock and the ligand considered the key. In practice, however, molecular docking will seek the ligand that can adjust to the receptor. (Drug Design, 2014).

A scoring function is needed to restrict the best-fit of molecular docking. The function is usually based on a force field used in protein simulation. Some scoring functions add other computation aspects, such as entropy. The complexity of ligand and receptor affect the function evaluation (Mukesh & Rakesh, 2011). Virtual screening, a computational technique performed in drug discovery, automatically evaluates data bank collection receptors with a ligand to determine a drug candidate (Rester, 2008). Computing power is needed to run these executable commands quickly. Thus, virtual screening based on molecular docking is a method often used in practice. We also use the method contained in Autodock and Autodock Vina. Modeling of chemical structure uses molecular modeling to study the structure's phenomena.

Autodock is an application developed by The Scripps Research Institute (Drug Design, 2014). There are two types of applications: Autodock and Autodock Vina. Each differs from the other in terms of the scoring function used and optimization of parallel computing by Autodock Vina. Autodock uses empirical free of force field energy with a Lamarckian Genetic Algorithm in predicting the bonding coordinate between the ligand and the receptor (Morris et al., 2009). In Autodock Vina, optimization is carried out using global particle swarm optimization and Broyden-Fletcher-Goldfarb-Shanno (BFGS) locally (Trott & Olson, 2010).

3. METHODOLOGY

In this work, we conduct a system performance benchmark, molecular dynamic simulation using Gromacs, and virtual screening using Autodock. We then analyze the performance of the simulation.

3.1. System Performance Benchmark

We conduct a system benchmark to evaluate performance of the container, VM, and native computer. We examine the three main components of a computer system: the processor, memory (RAM), and storage (hard disk). All are tested by the application pure specific test with emphasis on the performance of each component. The overall system test is also conducted to determine the collaboration of all three components.

Processor performance is evaluated using p7zip and POV-Ray. p7zip is compression software that is considered a CPU-bound application, while POV-Ray is rendering software that calculates heavy processes in the processor. Memory performance is evaluated using RAMSpeed SMP, a standard, widely used memory benchmark. Storage performance is evaluated using fio (Flexible IO Tester), a standard IO-bound software that evaluates the read and write processes in the system.

3.2. Molecular Dynamics on Docker using Gromacs

In performing Gromacs experiment, we first prepare all the necessary libraries. It is important to ensure that every prerequisite was installed accordingly. The proposed steps are preparing the system environment with a supporting library, installing the Gromacs latest stable release, running and producing a virtual sites file, and then repeating the last two steps to produce lysozyme in the water. Two experiments from Gromacs need to be analyzed: the process of creating a virtual site and producing lysozyme in water. This material comes from a molecular modeling tutorial (GROMACS tutorials, 2015). In this work, we consider the Gromacs timestamp of each environment.

3.3. Virtual Screening on Docker using Autodock and Autodock Vina

Our work is divided into three steps: pre-docking, docking, and analysis docking log results. Pre-docking consists of preparing receptor and ligand molecules used for molecular docking experiments. This step includes data pre-processing and data formatting. All the molecules are recorded in the form of a text file that is pre-processed to include the docking parameters and the potential location of docking activities. The outputs of this step are *.pdbqt and *.dpf files that contain the parameters needed for molecular docking. The second step is the main phase of molecular docking. Both Autodock and Autodock Vina are used in the experiments to provide a comprehensive analysis of system performance, as both applications are widely used in virtual screening. The last phase is docking log result analysis, which produces molecules that will be used as drug candidates in the next phase of drug discovery processes.

We create ten scenarios in this work, five involving Autodock and five involving Autodock Vina. Each scenario uses the same data and parameters. The difference in each scenario is the number of containers used and the number of receptors distributed into each container. The total number of receptors—the set of molecules that will be processed as a drug candidate—used in this experiment is 1,406 from herbaldb (Yanuar et al., 2011), a set of potential drug substances from Indonesian medicinal plants with one ligand as a targeted protein. The receptor distribution can be calculated by dividing the total number of receptors used by the number of containers in each scenario.

Table 1 Number of receptors in each container

Experiment	Container	Receptor
1	50	28
2	100	14
3	150	9
4	200	7
5	250	5

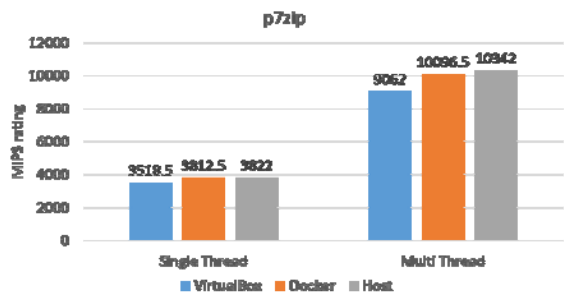
In measuring the performance of this work, we use running time as the parameter. This is the time needed to execute the entire set of molecules (screening), from the first molecule to the last. The running time parameter is measured using the embedded ‘date’ program available in the operating system (in this case, Debian Operating System). Furthermore, we record every running time for each container carried in each experiment and then study the longest, fastest, and average processing times. At the end of the experiment, we observe the running time for one molecule simulation in each scenario.

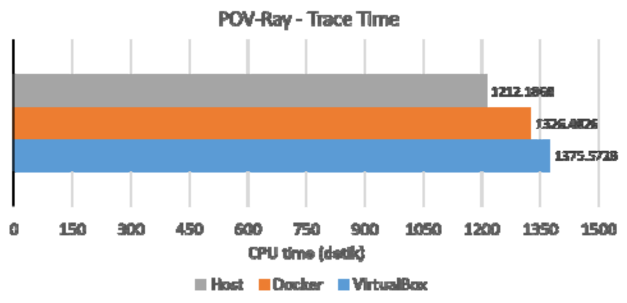
4. RESULTS AND DISCUSSION

4.1. System Performance Benchmark

The results are obtained from p7zip in the form of an average of MIPS rating for compression and decompression. This result can be seen in Figure 1a. On the left is the result of a single thread, while the result on the right is for multi-thread. For both parameters, the native host gets the highest yields, followed by Docker and VirtualBox. Even so, the difference between Docker and the native host is not as great as the difference between Docker and VirtualBox. The results show that from the perspective of p7zip, Docker can address the challenges commonly seen in the virtual machine, primarily related to the processor. The p7zip program that runs in Docker is seen as an ordinary process running on the host. This is clearly different from the VirtualBox. This explains why the results are almost equal to Docker's interaction with the native host.

POV-ray results can be seen in Figure 1b. This benchmark clearly shows that the native host is ahead of the other two: 13.48% faster than VirtualBox and 9.43% faster than Docker. This difference is quite significant considering that it is a long duration—about 20 minutes—which creates a difference in duration of more than 100 seconds. For scenes with a larger resolution (above 512×512 pixels), the duration will be greater, even with the difference in duration between the three instances.

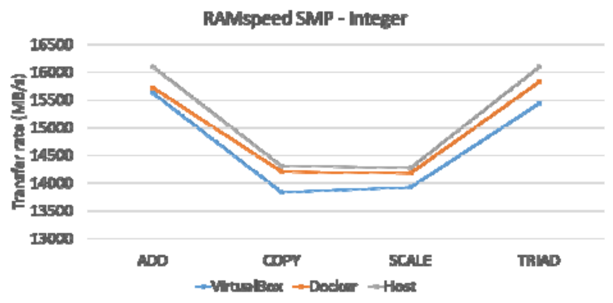


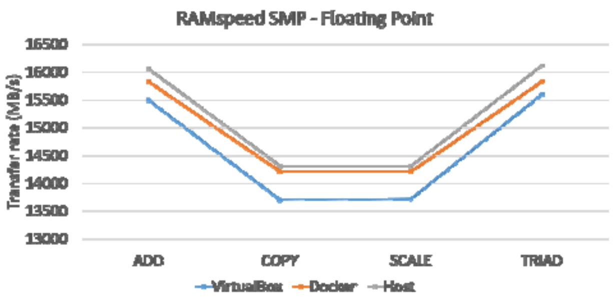


(a) (b)

Figure 1 Processor performance result

RAMSpeed SMP benchmark uses two benchmark aspects, integer and floating point overall. Though not as the same as the processor, memory is not degraded significantly in Docker and VirtualBox. The advantage of the host over Docker is less than 1% for the COPY operation and SCALE, and less than 2% for the ADD operation and TRIAD. Exceptions occur for surgery ADD in integer, where the hosts surpass Docker by a margin of 2.34%. For VirtualBox itself, the difference with the hosts is always below 5%. The result of memory performance benchmark can be seen in Figure 2.

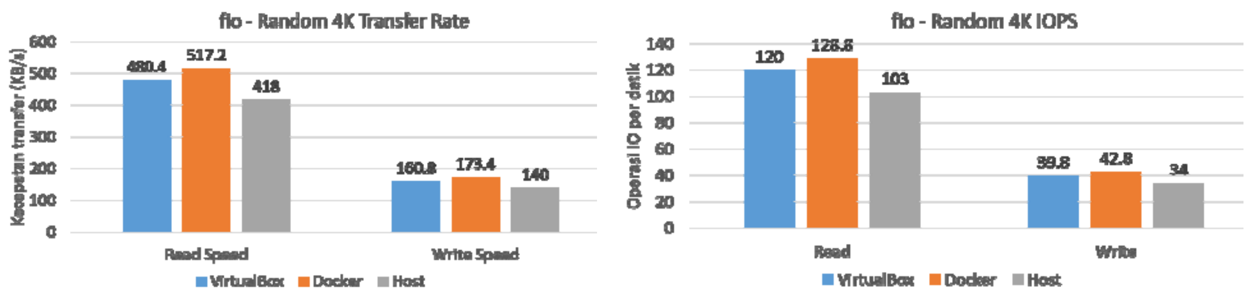




(a) (b)

Figure 2 Memory performance result

From the two charts in Figure 3, we can see the correlation between the IOPS and the transfer rate. The greater the IOPS or IO operations per second, the greater the speed of data transfer. In addition, it is seen here for the first time that the hosts receive the worst result among the three. The hosts got the lowest result because the condition of storage is not as good as in the other instances. At the time of testing, internal storage was used in each instance. At that time, the storage host had pretty much filled, leaving only about 30% of its total capacity empty. As for VirtualBox, there was about 50% empty storage. Implementation of storage by default for containers in Docker itself seems to resemble VirtualBox, where there is one file that is allocated as a virtual storage.



(a) (b)

Figure 3 Storage performance result

4.2. Molecular Dynamics Results

The results of the timestamp in processing virtual sites are presented in Table 2, while the results of lysozyme in the water are presented in Table 3. The results of virtual sites show that Docker has the best performance, slightly better than native and followed by VirtualBox. The results of lysozyme in the water show the same behavior as the results of virtual sites. The time spent by Docker and native to run virtual sites is around 4.475 and 4.534 seconds, respectively, while VirtualBox needs about 10.559 seconds. As for the time spent for running lysozyme in the water, Docker needs 1 hour, 38 minutes, and 52 seconds, while native needs 1 hour, 39 minutes, and 1 second, respectively. On the other hand, VirtualBox needs 8 hours, 16 minutes, and 29 seconds to complete the task. These results of the time spent is represented by wall time in the above tables. While core time is more about CPU time taken to complete the task, the performance parameter of ns/day and hour/ns provides broad information about how many molecular dynamics steps each system can perform in certain processes.

Table 2 Gromacs virtual sites timestep and performace results

	Time			Performance	
	Core t (s)	Wall t (s)	(%)	(ns/day)	(hour/ns)
VirtualBox	9.591	10.559	90.8	8182.753	0.003
Docker	4.426	4.475	98.9	9654.654	0.002
Native	4.477	4.534	98.7	9528.400	0.003

Table 3 Gromacs lysozyme timestep and performance results

	Time			Performance	
	Core t (s)	Wall t (s)	(%)	(ns/day)	(hour/ns)
VirtualBox	29473.294	29789.647	98.9	2.900	8.275
Docker	4247.023	5932.965	96.3	14.563	1.648
Native	4414.133	5941.477	98.0	14.542	1.650

From these results, it can be concluded that for processing or producing a bigger system, the performance gap will get wider between Docker, native, and VirtualBox.

As for graphical visualization, the result of virtual sites cannot be displayed due to the number of very small molecules. However, the result of lysozyme in the water is presented in Figure 4 below to verify that three environments produce the same result.

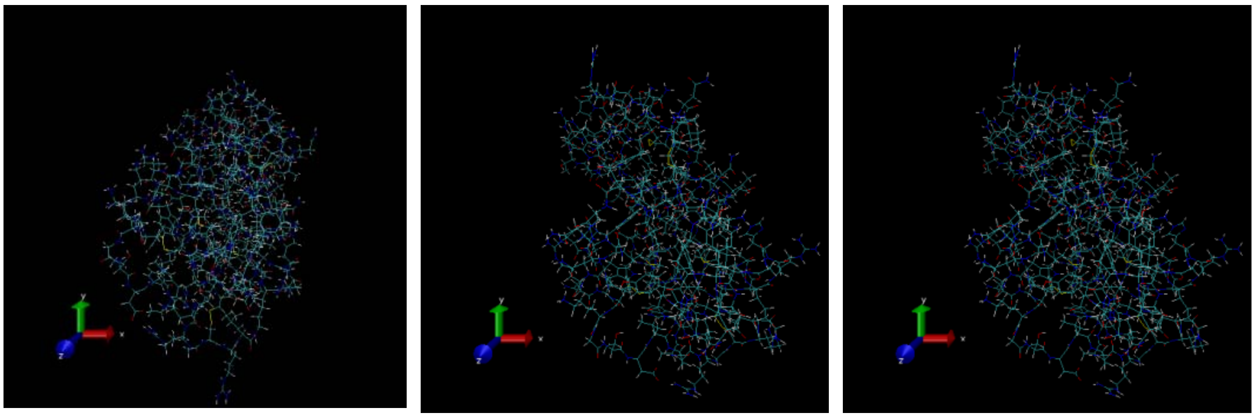


Figure 4 Lysozyme graphical visualization results: (a) VirtualBox; (b) Docker; and (c) Native

4.3. Virtual Screening Results

There are ten scenarios, five using Autodock and five using Autodock Vina. A different number of container is used for each scenario. The total number of receptors used is divided equally into containers. In each scenario, a different number of receptor processed in each container is based on the division between the number of receptors and the total number of containers used. Once the preparation is complete, the container application is run (Autodock or Autodock Vina) and the time is recorded from the beginning of the first data execution to the end of the time when the application finishes executing the last data. Table 4 presents the shortest, longest, and average execution times per container in every scenario. The following are data resulting from five scenarios using Autodock and Autodock Vina.

Table 4 Experiment results on virtual screening

Number of Containers	Execution Time (minutes) Autodock Vina			Execution Time (minutes) Autodock		
	Shortest	Longest	Average	Shortest	Longest	Average
50	625	1359	968	1044	1344	1255
100	407	1434	891	835	1386	1259
150	247	2615	745	710	1405	1216
200	129	1496	705	482	1276	1007
250	193	1431	720	358	1071	811

Execution in Autodock Vina is much faster than in Autodock. This is because Autodock Vina uses multithreading to optimize the performance of many CPUs (Trott & Olson, 2010). The “Average” column in Table 4 shows a decreasing trend as the value of the container increases. This indicates that the number of containers and the data affect the processing time. As the number of containers increases, the number of data in the container decrease, resulting in a shorter processing time.

Table 5 Amazon EC2 specifications and virtual screening results (Hilman, 2012)

Amazon EC2 Specification			
Parameter	SCluster	LCluster	XLCluster
Memory	1.7 GB	7.5 GB	1.5 GB
Compute Unit	1×1.2 GHz	4×1.2 GHz	8×1.2 GHz
Platform	32-bit	64-bit	64-bit
Cost/Hour	USD 0.085	USD 0.34	USD 0.68
Virtual Screening Result			
Total Exec Time (s)	375,624	108,179	48,421
Execution Time per Molecule			
Avg Exec Time per molecule (s)	189.9	54.76	24.18

In this work, we also compare the performance of Docker with the work of Hilman (2012). He used Autodock Vina applications for virtual screening in the Amazon EC2 environment, as shown in Table 5. Comparisons were made to determine the average time to process one molecule (cost per molecule). Table 5 also shows the time in thousands of seconds needed to process one molecule.

Table 6 Execution time per molecule on Docker (S)

Container	Autodock Vina	Autodock
50	41.30	53.55
100	38.02	54.01
150	32.19	52.29
200	30.08	43.37
250	31.02	35.00

The results of our experiment using both Autodock Vina and Autodock in Docker are provided in Table 6. It is clear that the results given by Docker on a desktop PC are not much different from those provided by Amazon EC2 as reported in Hilman (2012). Table 6 makes obvious that the execution of Docker with the smallest container is faster than the execution of LCluster. The fastest time is still held by XLCluster, but the results in Docker are still acceptable compared to those of virtual cluster in Amazon EC2.

5. CONCLUSION

In general, Docker performs better than VirtualBox and is competitive to native approaches. The greater the number of containers are used, the less time needed in the experiment. This occurs because a smaller amount of data is distributed to each container. The number of containers that can be made without having to run will be limited to the amount of memory storage capacity (hard disk), while the number of containers that can perform specific tasks will be limited by the existing RAM size. We also observe that the larger the number of containers, the smaller the relative weight value of each container sharing the CPU resources. The relative weight value will increase for each container if there is an idle container. When the idle container is given a task, then the relative weight value will be divided evenly to the other container.

In virtual screening experiments, Docker also has better performance than the VM represented by SCluster and LCluster of Amazon Elastic Compute Cloud. Docker is only slightly behind XLCluster, the most expensive VM instance used in the experiments. Among two widely used virtual screening applications, Autodock Vina is superior to Autodock in every scenario. Autodock Vina is recommended as a molecular docking application to be used bundled with the Docker container environment. Our results show that Docker container is suitable for bundling with MD and virtual screening applications. Docker performs much better than VirtualBox in terms of handling overhead delays, as the difference with native environment is not significant. Bundling applications in Docker makes it easier to reproduce and migrate them to different computational infrastructures.

6. ACKNOWLEDGEMENT

This research is supported by the 2015 PUPT (Penelitan Unggulan Perguruan Tinggi) Grants from The Indonesian Ministry of Research and Higher Education.

7. REFERENCES

Adufu, T., Choi, J., Kim, Y., 2015. Is Container-based Technology a Winner for High Performance Scientific Applications? *In*: IEEE-Network Operations and Management Symposium (APNOMS), 2015 17th Asia-Pacific, pp. 507–510

Alder, B.J., Wainwright, T.E., 1957. Phase Transition for a Hard Sphere System. *The Journal of Chemical Physics*, Volume 27(5), p. 1208–1209

Aparna , K., Nair, M.K., 2016. Incorporating Stability and Error-based Constraints for A novel Partitional Clustering Algorithm. *International Journal of Technology*, Volume 4, pp. 691–700

Bhushan, S.B., Reddy, C.H., Pradeep. 2016. A Four-level Linear Discriminant Analysis Based Service Selection in the Cloud Environment. *International Journal of Technology*, Volume 5, pp. 859–870

Chung, M.T., Quang-Hung, N., Nguyen, M.T., Thoai, N., 2016. Using Docker in High Performance Computing Applications. *In*: IEEE Sixth International Conference on Communications and Electronics (ICCE), pp. 52–57

Cito, J., Ferme, V., Gall, H.C. 2016. Using Docker Containers to Improve Reproducibility in Software and Web Engineering Research. *In*: International Conference on Web Engineering, Springer International Publishing, pp. 609–612

Docker, Inc. 2016. *What is Docker*? Available online at: https://www.docker.com/whatisdocker/, Accessed on 22 December 2016

Drug Design. 2014. Available online at: http://strbio.biochem.nchu.edu.tw/classes/special%20topics%20biochem/course%20ppts/rational%20drug%20design-2014.pdf, Accessed on 22 December 2016

Dua, R., Raja, A.R., Kakadia, D., 2014. Virtualization vs Containerization to support PaaS. *In*: 2014 IEEE International Conference on Cloud Engineering, pp. 610–614

Foundation, T.L., 2016. *2014 Enterprise End User Report*. Available online at: https://www.linux.com/publications/2014-enterprise-end-user-report, Accessed on 22 December 2016

Gromacs Manual 4.5.4., 2011. Available online at: ftp://ftp.gromacs.org/pub/manual/manual-4.5.4.pdf, Accessed on 22 December 2016

GROMACS Tutorials., 2015. Available online at: http://www.bevanlab.biochem.vt.edu/Pages/Personal/justin/gmx-tutorials/, Accessed on 22 December 2016

Hilman, M.H., 2012. Analisis Teknik Data Mining dan Kinerja Infrastruktur Komputasi Cloud Sebagai Bagian dari Sistem Perancangan Obat Terintegrasi. *Graduate Thesis*. Faculty of Computer Science, Universitas Indonesia

Kozhirbayev, Z., Sinnott, R.O., 2017. A Performance Comparison of Container-based Technologies for the Cloud. *Future Generation Computer Systems*, Volume 68, pp. 175–182

Morris, G.M., Huey, R., Lindstrom, W., Sanner, M.F., Belew, R.K., Goodsell, D.S., Olson, A.J., 2009. AutoDock4 and AutoDockTools4: Automated Docking with Selective Receptor Flexibility. *Journal of Computational Chemistry*, Volume 30(16), pp. 2785–2791

Mukesh, B., Rakesh, K., 2011. Molecular Docking: A Review. *International Journal of Research in Ayurveda & Pharmacy*, Volume 2(6), pp. 1746–1751

Rester, U., 2008. From Virtuality to Reality—Virtual Screening in Lead Discovery and Lead Optimization: A Medicinal Chemistry Perspective. *Current Opinion in Drug Discovery & Development*, Volume 11(4), pp. 559–568

Stillinger, F.H., Rahman, A., 1974. Improved Simulation of Liquid Water by Molecular Dynamics. *The Journal of Chemical Physics*, Volume 60(4), pp. 1545–1557

Suhartanto, H., Wibisono, A., Yanuar, A., 2015. Current Progress on the Development of Cloud Computing Platform to Support Drug Design based on Medical Plants: Is it Possible to have Cloud Service Migration? *In*: PRAGMA28 Workshop, Nara Institute of Science and Technology, Nara, Japan. Available online at: http://pragma28.pragma-grid.net/dct/page/1

Thiyagarajan, D., Ganesan, R., 2015. Data Security Model Employing Hyperelliptic Curve Cryptography (HECC) and Secure Hash Algorithm-3 (SHA-3) in Cloud Computing. *International Journal of Technology*, Volume 3, pp. 327–335

Trott, O., Olson, A.J., 2010. AutoDock Vina: Improving the Speed and Accuracy of Docking with a New Scoring Function, Efficient Optimization, and Multithreading. *Journal of Computational Chemistry*, Volume 31(2), pp. 455–461

Turnbull, J., 2014. *The Docker Book*. Available online at: https://www.dockerbook.com/, Accessed on 22 December 2016

Wibisono, A., Suhartanto, H., 2012. Cloud Computing Model and Implementation of Molecular Dynamics Simulation using Amber and Gromacs. *In*: 2012 International Conference on Advanced Computer Science and Information Systems (ICACSIS). Depok, Indonesia

Yanuar, A., Mun'im, A., Lagho, A.B.A., Syahdi, R.R., Rahmat, M., Suhartanto, H., 2011. Medicinal Plant Database and Three-Dimensional Structure of the Chemical Compounds from Medicinal Plants in Indonesia. *International Journal of Computer Science Issues*, Volume 8(5), pp. 180–183