IJCCS (Indonesian Journal of Computing and Cybernetics Systems)
Vol.14, No.3, July 2020, pp. 319~330
ISSN (print): 1978-1520, ISSN (online): 2460-7258
DOI: https://doi.org/10.22146/ijccs.57565

Optimizing Virtual Resources Management using Docker on Cloud Applications

Rendra Felani[1], Moh Noor Al Azam[2], Derry Pramono Adi[3], Agung Widodo[4], Agustinus Bimo Gumelar[*5]
[1,2,3,4,5]Fakultas Ilmu Komputer, Universitas Narotama; Surabaya, Indonesia
e-mail: [1]**rendra14@fik.narotama.ac.id**, [2]noor.azam@narotama.ac.id, [3]derryalbertus@ieee.org, [4]agung.widodo@narotama.ac.id, [*5]**bimogumelar@ieee.org**

Abstrak

Penelitian ini bertujuan untuk mengoptimalkan server yang memiliki tingkat utilitas rendah pada perangkat keras menggunakan teknik virtualisasi wadah dari Docker. Fokus utama dalam penelitian ini adalah memaksimalkan kerja, CPU, RAM dan Hard Drive. Penerapan teknik virtualisasi adalah untuk membuat banyak wadah karena masing-masing wadah adalah untuk aplikasi untuk menjalankan sistem penyimpanan cloud dengan konsep infrastruktur layanan CaaS (Container as a Service). Kontainer pada infrastruktur akan berinteraksi dengan kontainer lain menggunakan perintah konfigurasi di Docker untuk membentuk layanan infrastruktur seperti CaaS pada umumnya. Pengujian perangkat keras dilakukan dengan menjalankan lima aplikasi penyimpanan cloud Nextcloud dan lima aplikasi basis data MariaDB yang berjalan dalam wadah Docker dan diuji dengan pengujian acak menggunakan dataset multimedia. Pengujian acak memerintahkan pemrosesan dataset termasuk mengunggah dan mengunduh dataset secara bersamaan dan memantau sumber daya CPU, RAM, dan perangkat keras Disk saat memproses dataset menggunakan statistik Docker, HTOP, dan alat pemantauan Cockpit untuk menentukan kemampuan perangkat keras saat memproses dataset multimedia.

Kata kunci*— CaaS, Container, Docker, Virtualization*

Abstract

This study aims to optimize servers with low utility levels on hardware using container virtualization techniques from Docker. This study's primary focus is to maximize the work of the CPU, RAM, and Hard Drive. The application of virtualization techniques is to create many containers as each of the containers is for the application to run a cloud storage system with the CaaS service infrastructure concept (Container as a Service). Containers on infrastructure will interact with other containers using configuration commands at Docker to form an infrastructure service such as CaaS in general. Testing of hardware carried out by running five Nextcloud cloud storage applications and five MariaDB database applications running in Docker containers and tested by random testing using a multimedia dataset. Random testing with datasets includes uploading and downloading datasets simultaneously and CPU monitoring under load, RAM, and Disk hardware resources. The testing will be done using Docker stats, HTOP, and Cockpit monitoring tools to determine the hardware capabilities when processing multimedia datasets.

Keywords*— CaaS, Container, Docker, Virtualization*

Received July 4th,2020; Revised July 18th, 2020; Accepted July 31th, 2020

1. INTRODUCTION

The development of infrastructure technology in cloud systems, especially cloud storage, has increased significantly along with the emergence of public clouds and private clouds. Many companies also have shifted workloads to the cloud [1], [2]. In recent years, virtualization technology to support cloud infrastructure became popular. A common problem with virtualization is that the required hardware must be above average in terms of performance (which will also inevitably increase cost on hardware) to run the system, causing servers to experience many hardware changes. Over time, cloud storage technology relies heavily on infrastructure to run its operations. With the increasing need to build cloud storage infrastructure, a system administrator must be able to design infrastructure on an existing server to run cloud storage optimally. To meet the server's needs while simultaneously reducing cost, server administrators must "tweak" the technology to replace traditional virtualization techniques. Docker containers are present to provide solutions to traditional virtualization, such as full virtualization and paravirtualization, by saving resources on hardware, such as CPUs, RAMs, and Hard Drives. Chung et al. pointed out that the Docker container technology as a virtualization management operating system using Docker containers has improved scalability [3]. Docker can replace the performance of smaller and faster hypervisors to start virtualization.

In many other computing environments, where traditional virtualization is still an eligible option, cloud computing is struggling using this old technology. Cloud servers generally are used to host multiple virtual machines in the same physical server [4]. However, in Karpoff and Lake's patented work, it is stated that using virtualization, the virtual disk image is known to the host computer can be larger than the actual consumed amount of physical storage [5]. Traditional virtualization will take a high toll on physical servers, especially on High-Performance Computer (HPC). Administrators will find themselves more likely to add new servers, which is not cost-effective [6], [7].

2. METHODS

In the past, computing operation operated using one physical server, and each server is running by one application [8], [9]. So, it would be demanding several physical servers if it runs several applications. Considering these limitations, we used the Docker as the "wrapper" of each self-managed and separate running application, which will later retrieve the resources needed [10]. In this section, the whole design of experimental scope settings is presented to determine the Docker container's performance when processing data with a multimedia dataset.

2.1 Container Virtualization

Docker is a container virtualization technology that behaves similarly to a lightweight virtual machine; Docker container has emerged as a complement to virtual machine technology; it also offers slightly less isolation between processes. They are lighter and easier to share with [11], [12].

Docker container virtualization is a virtualization method for running multiple applications isolated on the host using the main operating system's kernel sharing technique. Container virtualization is often called operating-system-level virtualization, which allows running multiple applications on one host. Figure 1 shows a simple illustration of the architectural differences between Docker containers and Virtual Machines; it also shows that the Docker container is more compact in running applications and does not require a guest OS. Therefore it should have lighter architectural advantages running on the server.

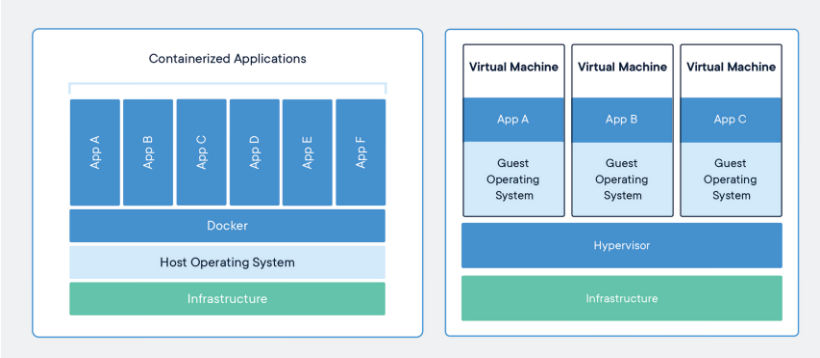

Figure 1 Comparison of Docker Container and Virtual Machine Architecture [13]

2.2 Cloud Storage Application

Nextcloud is a client-server application for creating and operating file hosting services, meaning its data can be accessed almost everywhere [14]. The use of the Nextcloud serves as a medium for entering data as a parameter testing the server's strength in serving the requested multiprocess on the client-side. Besides Nextcloud, there are other cloud storage applications, but Nextcloud officially supports Docker container technology, which is proven by the availability of images on Docker hub. In other words, Nextcloud is capable and ready to use in Docker container architecture [15].

2.3 Flow of The Experiment

The flow of the experiment carried out in this research is depicted in Figure 2. The first step is to collect multimedia datasets. This dataset contains video files, application files, and picture files. Further explanation of the dataset used in this research is presented in sub-section III.B "Multimedia Dataset." The second step is to put the collected dataset into the storage of a low-spec server. Briefly, the server has 2GB of RAM, Dual-core CPU, and 60GB of storage.

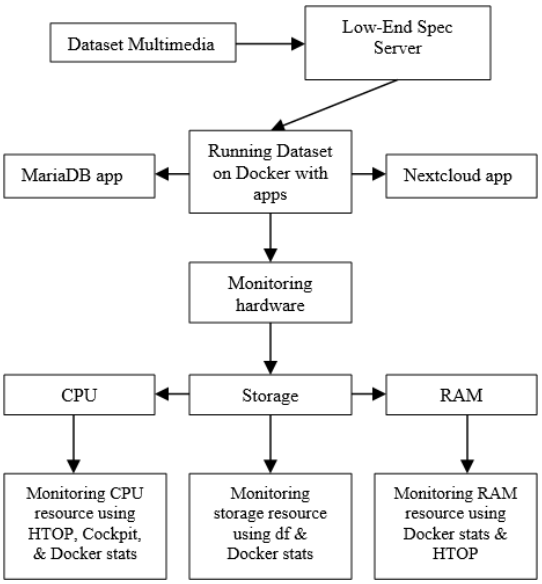

Figure 2 Block Diagram of Experiment's Flow

Using Docker, we run configuration for the server for MariaDB and Nextcloud application [16], [17]. These applications apply as the dataset processing media and monitoring for each of hardware performance while in different states. This step is done using various monitoring tools, namely HTOP, Docker stats, df, and Cockpit [18], [19]. Each hardware device can use the same monitoring tool due to the tool being able to read much information contained in the server's hardware.

The primary aim of this experiment is to test low-spec server performance by running ten cloud applications simultaneously using Docker as an engine to manage several running applications. This experiment is expected to be used as a reference in selecting hardware in building a server that will be used to run cloud applications and optimize its hardware performance.

2.4 Multimedia Dataset

The experiment in this study uses a multimedia dataset of 4.12GB containing video files, image files, ISO files, and audio files with the testing process carried out by uploading and downloading simultaneously using the Nextcloud (a cloud storage application) that runs above the Docker container.

In the experiment session, uploading and downloading datasets from and to the server will be carried out through the Nextcloud, which results in changes in hardware resources on the server. These changes will be recorded and reviewed to determine the server's ability to process the dataset in the experiment. The design of the program for testing is made from a Docker compose script, which is immediately executed only once. The script settings contain a pair of applications between the Nextcloud and MariaDB that are interconnected, making it able to be run all at once as a cloud storage application.

Table 2 Docker Compose Script

	nextcloud:
	container_name: nextcloud **image: nextcloud**
1	ports: - 80:80
2	volumes:
3	- /mnt/vol/containers/cloud/nextcloud/apps:/var/www/html/apps
4	- /mnt/vol/containers/cloud/nextcloud/config:/var/www/html/config
5	- /mnt/vol/containers/cloud/nextcloud/data:/var/www/html/data
6	**depends_on:**
7	- db
8	db:
9	container_name: maria-db
10	image: mariadb
11	volumes:
12	- /mnt/vol/containers/cloud/mariadb:/var/lib/mysql

It can also be seen that each Nextcloud will be paired with the MariaDB, which is located in a different container. In total, there are ten Docker containers with details of five Nextcloud containers and five Docker containers. The dataset used in testing uses a multimedia dataset consisting of video files, image files, ISO files, and audio files. The details of the dataset used in the experiment are shown in Table 2.

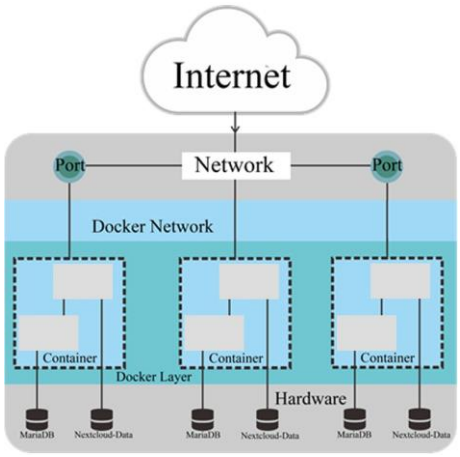

Figure 3 Architectural design within the Docker Container Scope

Before setting Nextcloud on the Docker, we added the "Docker-compose" tool to run orchestrate on containers and monitor tools to monitor server activity at idle and when processing multimedia datasets. To run and monitor CPU performance, we use three standard monitoring applications, namely Docker Stats, HTOP, and Cockpit monitoring.

Docker Stats is a monitoring tool specifically designed to monitor Docker containers. HTOP is the default Ubuntu Server monitoring tool, mainly used to monitor hosts or virtual Operating System resources and run applications. The Docker statistics command provides for observing running container status, resource memory, and I/O networks. HTOP is the main system monitor, commonly used on most Linux-based operating systems. With HTOP, we can see CPU Usage, Memory Usage, and Use of Swap Files, all distinguished in color graphics format. Performing tasks, the average workload is displayed at the top of the HTOP. Therefore, HTOP is an easy-to-use system monitoring tool, in a very efficient yet real-time, capable of displaying a complete list of ongoing processes. The third monitoring application is the Cockpit. The Cockpit can manage containers through Docker. This functionality is present in the Cockpit Docker package. Cockpit communicates with Docker daemons via the API via socket /var/run/docker.sock UNIX [18]. By doing this, we will be able to determine the server's ability to process multimedia datasets.

2.5 Hardware Specification

The servers used in the experiment with specifications shown in Table 3 and the scope for Nextcloud to run in the Docker container are shown in Figure 4.

Table 3 Hardware Specification of the Tested Server

Processor	Intel (R) Dual-core(R) CPU Dual-Core @2.20GHz
CPU Core (s)	2 Cores
RAM	2 GB
Harddisk	60 GB
Platform	Ubuntu Server 16.04.4 LTS, Docker 18.03.0-ce, Docker compose, Nextcloud, MariaDB
Monitoring Tools	Docker stats, HTOP, Cockpit monitoring

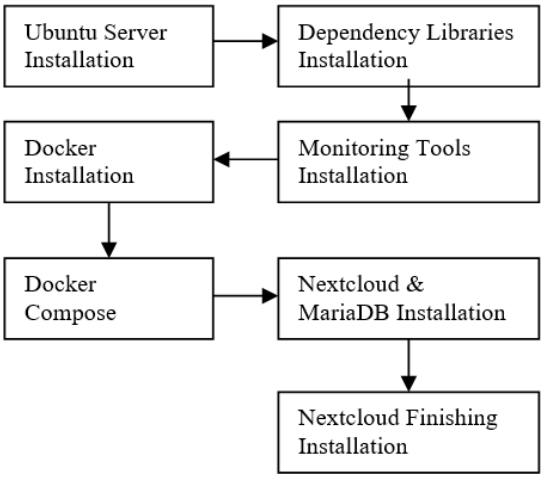

Figure 4 Block Diagram of Nextcloud Configuration on Docker

Optimization in this study focused on testing servers with low specifications to run multi-service cloud storage applications in Docker containers. The workload, when uploading and downloading multimedia datasets, certainly impacted the server hardware.

3. RESULTS AND DISCUSSION

The experiment begins by recording the initial condition of the server resource before running the multimedia dataset. Recording server conditions at idle is needed to determine the conditions under which the server runs the Docker container and the main operating system [20].

Figure 5 CPU Condition on Idle

Figure 5 shows the general patterns that occur during initiation or conditions before conducting experiments with multimedia datasets. The record of these conditions will be used as a parameter of resource changes when processing the datasets. The CPU resources needed to run the entire system, including the Docker container. The total resources needed by the server are around 4% and 9% of the resources that have two cores on the processor. The distribution of the CPU resource detail for each container will be described in Figure 6, with details of each running Docker container.

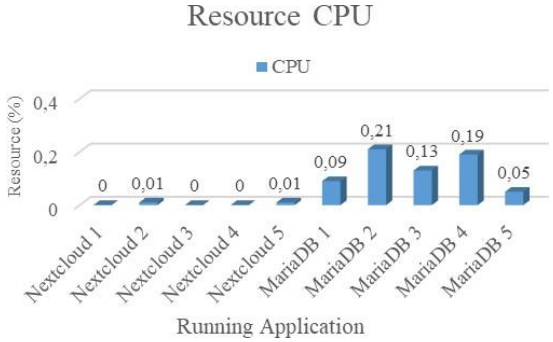

Figure 6 Conditions of Each Applications Running on Top of Docker

Figure 6 shows the resource container when idle or is not processing the dataset. Figure 6 shows that the MariaDB database container takes up more CPU resources, while the Nextcloud container does not consume much of the CPU resources. The use of significant resources occurs in MariaDB because the dataset is extensive and stored on a single storage device, which inevitably affects the global server performance conditions. Transfer data and loading are considered not in the ideal condition since they took too much time in a simple queuing system. The type of data used in this experiment is a vast set of structured data. In general, MongoDB is used in many cases to store unstructured documents or data. Data stored in MongoDB can then be reviewed and analyzed so that more structured information can be stored in other databases. The database in SQL format has always been a viable choice for big data architecture services. In MongoDB's internal architecture, relational databases would fail if the collected data is not standardized and organized into large objects, such as documents and multimedia clip objects.

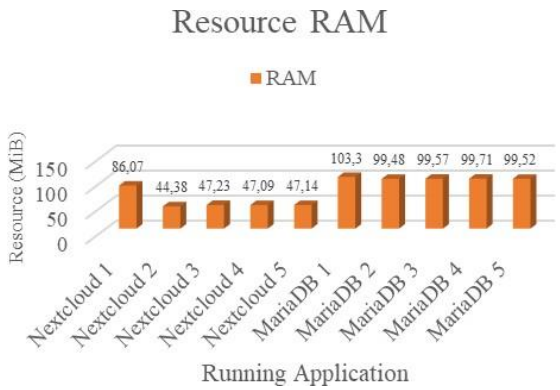

Figure 7 RAM Condition on Idle

Resources of RAM are the most significant database element because they help change the variables of the database program. Additional memory requires bigger keys and table caches stored in memory to allow disks to navigate; the order of magnitude would be decreased later. Figure 7 shows a chart of RAM activity on idle. It can be seen that RAM conditions in the MariaDB database container show a higher usage difference even though it is on idle. This phenomenon is likely to happen due to the use of memory swaps.

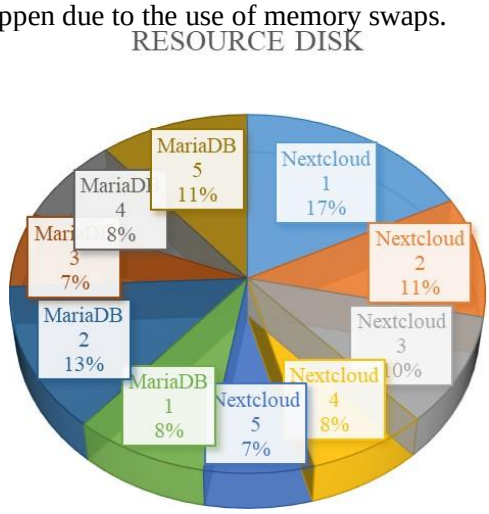

Figure 8 Pie Chart of Hard Drive's Capacity on Idle

Another feature that is not less important is the HyperThreading (HT) feature. HT involves two processing units that share cache on one hardware (single-core). If two cores are put to work on a similar task, then a cache will be quite useful. MySQL-based databases are lacking excellent performance while incorporating multiple cores. So, if the HT feature is

disabled, the remaining cores will run a bit faster. Figure 8 shows the sharing of capacity on the hard drive. Details of sharing capacity in the Docker container are shown in Table IV, recorded using Docker stats as its monitoring tools.

Table 4 Overall Resource on Docker Container on Idle

Name	CPU%	Mem. Usage	Mem%	Net I/O	Block I/O
nextcloud1	0.00%	86.07MiB	4.30%	2.26MB/ 1.96MB	36.4MB/ 1.6MB
nextcloud5	0.00%	47.14MiB	2.36%	148kB/ 658kB	14.3MB/ 0B
nextcloud3	0.00%	47.23MiB	2.36%	143kB/ 556kB	20.5MB/ 0B
nextcloud2	0.00%	44.38MiB	2.22%	141kB/ 556kB	22.7MB/ 0B
nextcloud4	0.00%	47.09MiB	2.35%	155kB/ 663kB	15.7MB/ 0B
maria-db3	0.13%	99.57MiB	4.98%	31.8kB/ 105kB	14.1MB/ 2.89MB
maria-db5	0.05%	99.52MiB	4.98%	31.3kB/ 104kB	22.9MB/ 2.83MB
maria-db4	0.19%	99.71MiB	4.99%	33.6kB/ 111kB	16.5MB/ 2.74MB
maria-db2	0.21%	99.48MiB	4.97%	32.2kB/ 105kB	27.3MB/ 2.83MB
maria-db1	0.09%	99.48MiB	4.97%	659kB/ 2.05MB	17.5MB/ 30.1MB

Table 4 shows all the resources needed by the Docker container: starting from the CPU, RAM, and Disk. It is seen in Table 4 is a wasteful container that consumes RAM, the MariaDB database container. For the Nextcloud container, it does not consume much RAM. Next is the recording of monitoring when randomly targeted, which starts recording CPU resources on the Cockpit monitoring.

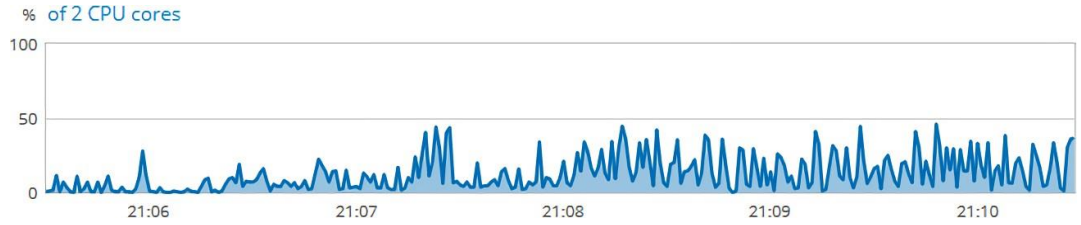

Figure 9 CPU Resources when Processing Datasets, Recorded by the Cockpit

Figure 9 shows the CPU resource movement when processing the dataset. The chart from Figure 9 shows that the CPU works around 50% of the available resources. The CPU still leaves many resources when random testing. In a randomized test, it took five minutes with the CPU working average at a maximum level of 50 out of 100, recorded by the Cockpit monitoring. The next step is to divide the resource container in the post-testing phase.

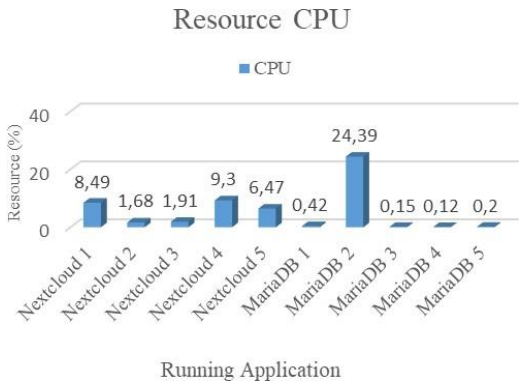

Figure 10 Chart of the Division of Labor on Randomized Test of Docker Containers

Figure 10 shows that the Nextcloud resource container has high resource spikes. Not all of the Nextcloud resource containers have a significant increase. However, from this case, it can be seen if the container is experiencing a heavy processing load, it will increase the resource needed by the container.

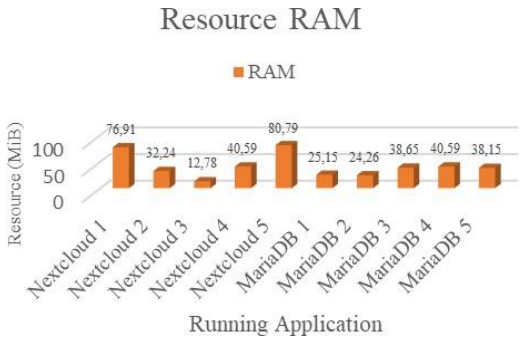

Figure 11 RAM Condition on Processing the Dataset

Figure 11 shows a chart containing resource RAM, which shows RAM activity increased significantly in the Nextcloud container. We encountered an increase in RAM when processing datasets to reach 89% of 100% of existing resources, as recorded by HTOP. Nevertheless, that did not last long, just a few seconds; then, RAM experienced a 70% to 80% decrease in the Nextcloud container. From this test, it can be seen that large RAM requirements are fundamental and inevitable in server building.

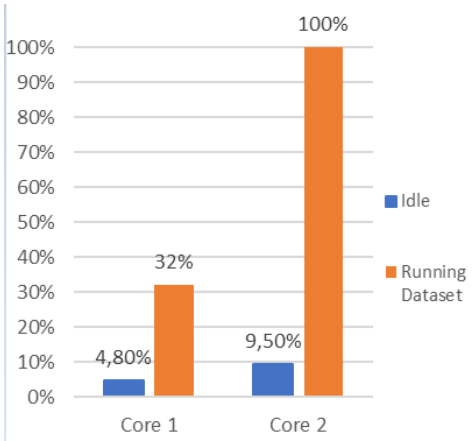

Figure 12 CPU Condition when Running the Dataset, Recorded by HTOP

Figure 12 shows the CPU resource change process when processing datasets where the CPU condition with core number 2 works optimally until it reaches 100%; it also can be seen that the core processor alternates in processing data when core number 1 runs optimally, the

core number 2 gives free space to the core processor, up to 40% and alternates continuously until the testing process is complete.

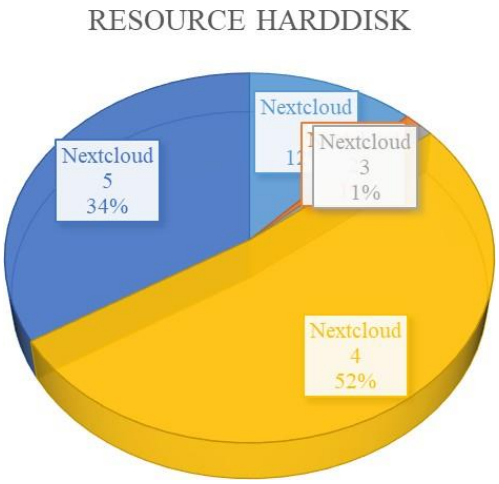

Figure 13 Pie Chart showing Hard Drive's Capacity when Processing Datasets

Figure. 13 shows the resource disk described in the form of a pie chart when processing a dataset. In Figure. 11, it can be seen that the most moving resource is the resource of the Nextcloud container. To find out the details of all the resources used by the Docker container, see Figure. 12.

Table 5 Overall Resource on Docker when Processing Dataset

Name	CPU%	Mem. Usage	Mem%	Mem. Avail.	Block I/O
nextcloud1	8.49%	76.91MiB	3.85%	600MB/ 4.2MB	621MB/ 1.61MB
nextcloud5	6.47%	80.79MiB	4.04%	1.57GB/ 58.6MB	1.73GB/ 168kB
nextcloud3	1.91%	12.78MiB	0.64%	11.8MB/ 1.98MB	53.6MB/ 9.45MB
nextcloud2	1.68%	32.24MiB	1.61%	6.86MB/ 1.82MB	63.6MB/ 5.13MB
nextcloud4	9.30%	97.28MiB	4.86%	2.31GB/ 608MB	2.65GB/ 0B
maria-db3	0.15%	38.65MiB	1.93%	672kB/ 2.03MB	17.9MB/ 47.3MB
maria-db5	0.20%	38.15MiB	1.91%	2.44MB/ 7MB	26.2MB/ 123MB
maria-db4	0.12%	40.59MiB	2.03%	3.24MB/ 9.45MB	20MB/ 139MB
maria-db2	24.39%	24.26MiB	1.21%	576kB/ 1.79MB	36.3MB/ 35MB
maria-db1	0.42%	25.15MiB	1.26%	1.79MB/ 5.53MB	23.3MB/ 92.7MB

Table 5 shows the resource containers recorded from Docker stats. In Table 5, it can be seen that the performance of the Nextcloud container consumes the most RAM resources. While the disk and CPU, resources do not experience many increases. For details on disk resource performance, see Figure. 14.

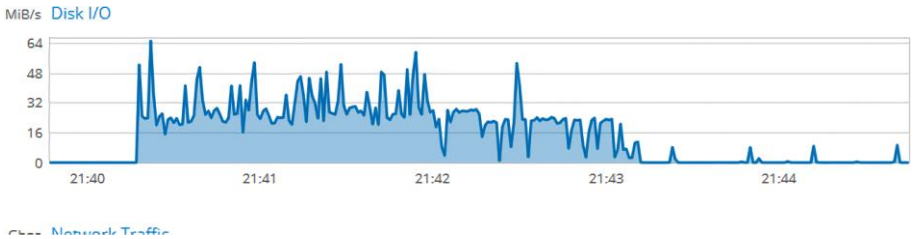

Figure 14 Chart of Disk trends during Random Testing using datasets

Figure 14 shows the resource movement on the Disk when processing the dataset. In Figure 14, it can be seen that there is an increase in the disk with a maximum number of 64% of 100% of the available disks. In the chart, it can be seen that there is a time window for 5 minutes when the resource processes the dataset; resources experienced a significant increase. Then, the resource conditions are gradually back to idle.

4. CONCLUSIONS

The test results show that optimization means the server can run multiple applications all at once. It takes about five servers with the traditional infrastructure to build cloud storage compared to the optimized one. Servers with Virtual Machines require high-end hardware. Contradicted and proved with this experiment's result, the Docker container virtualization can tackle this high-cost hardware. These are proven by resource monitoring of CPU, showing a randomized dataset test resulting in 15.5% and 18.8% percentage of performance, respective to its cores. For RAM, it shows 1.3GB is in use while processing the dataset, with 61.8% of total usage. The remaining Hard Drive capacity is around 46GB of a total of 56GB. The test takes up to 14% of hard drive resources, including the main operating system (Ubuntu Server 16.04 LTS), Docker, monitoring tools, and Docker's image application.

Future work is expected to use SSD or NVMe-based storage for faster response time and a bigger memory size. We also planned to utilize load balancing and scaling to coordinate and manage their execution and handle issues related. Furthermore, this model is beneficial for the development in swarm and orchestration mode.

REFERENCES

[1] J. Kumar and A. K. Singh, "Workload Prediction in Cloud using Artificial Neural Network and Adaptive Differential Evolution," *Futur. Gener. Comput. Syst.*, vol. 81, pp. 41–52, Apr. 2018.

[2] M. Attaran and J. Woods, "Cloud Computing Technology: Improving Small Business Performance using the Internet," *J. Small Bus. Entrep.*, vol. 31, no. 6, pp. 495–519, Nov. 2019.

[3] M. T. Chung, N. Quang-Hung, M.-T. Nguyen, and N. Thoai, "Using Docker in high performance computing applications," in *2016 IEEE Sixth International Conference on Communications and Electronics (ICCE)*, 2016, pp. 52–57.

[4] F. Sabahi, "Secure Virtualization for Cloud Environment Using Hypervisor-based Technology," *Int. J. Mach. Learn. Comput.*, vol. 2, no. 1, pp. 39–45, 2012.

[5] W. Karpoff and B. Lake, "Storage virtualization system and methods." Google Patents, 2009.

[6] A. Rosenthal, P. Mork, M. H. Li, J. Stanford, D. Koester, and P. Reynolds, "Cloud Computing: A New Business Paradigm for Biomedical Information Sharing," *J. Biomed. Inform.*, vol. 43, no. 2, pp. 342–353, Apr. 2010.

[7] J.-H. Huh, "Server Operation and Virtualization to Save Energy and Cost in Future Sustainable Computing," *Sustainability*, vol. 10, no. 6, p. 1919, Jun. 2018.

[8] I. Mohiuddin and A. Almogren, "Workload Aware VM Consolidation Method in Edge/Cloud Computing for IoT Applications," *J. Parallel Distrib. Comput.*, vol. 123, pp. 204–214, Jan. 2019.

[9] W. Wu, W. Lin, and Z. Peng, "An Intelligent Power Consumption Model for Virtual Machines under CPU-intensive Workload in Cloud Environment," *Soft Comput.*, vol. 21, no. 19, pp. 5755–5764, Oct. 2017.

[10] J. Turnbull, *The Docker Book*. Turnbull Press, 2014.

[11] M. Nardelli, C. Hochreiner, and S. Schulte, "Elastic Provisioning of Virtual Machines for Container Deployment," in *Proceedings of the 8th ACM/SPEC on International Conference on Performance Engineering Companion - ICPE '17 Companion*, 2017, pp. 5–10.

[12] I. Mavridis and H. Karatza, "Performance and Overhead Study of Containers Running on Top of Virtual Machines," in *2017 IEEE 19th Conference on Business Informatics (CBI)*, 2017, pp. 32–38.

[13] L.-H. Hung, D. Kristiyanto, S. B. Lee, and K. Y. Yeung, "GUIdock: Using Docker Containers with a Common Graphics User Interface to Address the Reproducibility of Research," *PLoS One*, vol. 11, no. 4, p. e0152686, Apr. 2016.

[14] T. Ngo, "Cloud Security: Private Cloud Solution with End-to-end Encryption," Search Results Web result with site links Haaga-Helia University of Applied Sciences, 2018.

[15] R. Yasrab, "Platform-as-a-Service (PaaS): The Next Hype of Cloud Computing," *CoRR*, vol. abs/1804.1, 2018.

[16] S. Kariyattin, S. Marru, and M. Pierce, "Evaluating NextCloud as a File Storage for Apache Airavata," in *Proceedings of the Practice and Experience on Advanced Research Computing*, 2018, pp. 1–4.

[17] W. Wood, "MariaDB Solution," in *Migrating to MariaDB*, Berkeley, CA: Apress, 2019, pp. 59–71.

[18] A. B. Gumelar, D. A. Lusia, A. Widodo, and R. Felani, "Using Neural Networks on Cloud Container's Performance Comparison By R on Docker (ROCKER)," *2018 Int. Symp. Adv. Intell. Informatics*, p. 5, 2018.

[19] L. Huang, K. Milfeld, and S. Liu, "Tools for Monitoring CPU Usage and Affinity in Multicore Supercomputers," 2020, pp. 69–86.

[20] A. B. Gumelar, "An Anatomy of Machine Learning Data Visualization," in *2019 International Seminar on Application for Technology of Information and Communication (iSemantic)*, 2019, pp. 1–6.