The Performance of Hotel Management System Using Microservices and Containerization Technology

Bekti Maryuni Susanto[1], Ery Setiyawan Jullev Atmadji[2], Lukman Hakim[3]

{bekti@polije.ac.id[1], ery@polije.ac.id[2], lukman.hakim@polije.ac.id[3]}

Information Technology Department, Politeknik Negeri Jember, Jl. Mastrip Kotak Pos 164 Jember Indonesia[1,2,3]

Abstract. Today, agile development of scalable applications that influence new forms of production and business organization is a requirement for organizations. Scalability and quick development requirements are no longer met by traditional monolithic architectures. Docker containerization is a new emerging technology bringing virtualization to software applications. In particular, lightness has brought higher profits to docker containers. This research aims to measure the performance of applications running on containerization architecture and compare it with conventional architecture namely monolithic architecture. The experiment was carried out on a computer with a Windows operating system on which the Docker desktop application was installed. Performance is measured using the Apache JMeter application to determine throughput, latency, packet loss, and delay. Analysis is carried out by comparing the results of parameter measurements. The results show that monolithic architecture has better latency values compared to microservices architecture.

Keywords: Virtualization, Docker Container, Cloud Computing, Hotel Management System.

1 Introduction

Today, agile development of scalable applications that influence new forms of production and business organization is a requirement for organizations. Scalability and quick development requirements are no longer met by traditional monolithic architectures [1]. Organizations want strong, technologically based solutions. To assist software products meet functional requirements and be resource-efficient, software engineers have created and implemented a variety of architectures over time. Certain architectures disperse their modules across multiple layers or tiers, or they may be arranged in a single layer. Since the invention of software systems, monolithic design, when combined with virtual machines, has shown to be a successful and efficient strategy for both small and large-scale projects. It is well known that when the volume of data to be handled grows or surpasses a particular capacity threshold, the performance of monolithic programs is impacted.

A new emerging technology called Docker containerization brings virtualization to software applications. It delivers an ultra-lightweight infrastructure technology for software applications

ICoSHIP 2023, November 18-19, Surabaya, Indonesia
Copyright © 2024 EAI
DOI 10.4108/eai.18-11-2023.2342548

resulting in significant take-up to develop, test , and deploy. Concerning this, a major issue in many online forums has occurred, especially about deploying distributed software applications on Docker-based containers in a more leveraged manner [2].

The mapping study results conducted by [3] show increasing interest and use of container-based technologies, such as Linux Container (LXC) or Docker as solutions of lightweight virtualization at the Infrastructure as a Service (IaaS) level, and as solutions of application management at the Platform as a service (PaaS) level. As observed, containers have a positive impact on several aspects, especially development and deployment. For example, architecture in the cloud is moving towards a DevOps-based approach, supporting continuous development and deployment pathways by considering cloud-native architectural solutions that are based on containers and their orchestration[3].

A single physical machine is deployed for limited applications and it results in hardware resources underutilization [4]. The idea of abstracting physical system resources into multiple virtual computing resources called virtualization originated from IBM. In 1990, it was commercialized for x86 computer systems. Virtualization techniques are considered as the cloud computing data center backbone as they allow deploying multiple virtual servers over a single physical server system. Thus, virtualization improves resource utilization and increases return on investment. It provides an abstraction over physical resources that can be shared by cloud users. The comparison among deployments of the application using traditional, hypervisor, and container architecture is shown in Figure 1.

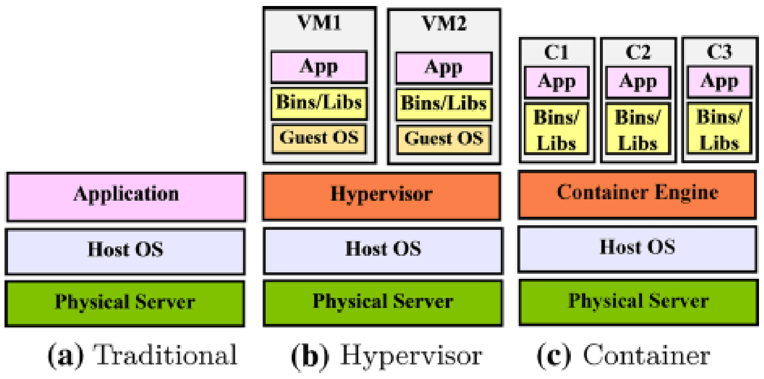

Fig. 1. Comparison of application deployment traditional, hypervisor, and container architecture)

The Microservices Architecture pattern possesses many pivotal benefits. First, it addresses complexity problems. It decomposes what might be a terrible monolithic application into a series of services. Second, it allows each service to be developed by a particular team independently. Those who develop the service are free to choose any technology that makes sense, as long as it respects the API contract. Most organizations prefer to avoid total anarchy by limiting choices of technology. Third, it allows each microservice to be used independently. Developers do not need to coordinate the local changes implementation to their services. Last, it allows each service to scale independently [5].

This research aims to measure the performance of applications running on containerization architecture and compare it with conventional architecture namely monolithic architecture. The experiment was carried out on a computer with a Windows operating system on which the Docker desktop application was installed. Performance is measured using the Apache JMeter application to determine throughput, latency, packet loss, and delay. Analysis was carried out by making a comparison of the parameter measurement results.

2 Method

The research encompassed several stages, including requirements analysis, the design and implementation of a hotel management information system, and the testing of the system's architecture. Requirements analysis was carried out by identifying the software and hardware needed for system development. The object of this research was the Integrated Hospitality Laboratory of Politeknik Negeri Jember. Interviews with integrated hospitality teaching factory managers were conducted to identify the system needs. The requirement of software and hardware used to develop a hotel management information system is shown in Table 1.

Table 1. Requirement of software and hardware used to develop hotel management information system

Number	Requirement type	Specifications
1	Software	Web Server, PHP 8.2 Laravel framework Visual studio code Mysql database Docker Desktop Microsoft Windows 11 operating system
2	Hardware	CPU intel min 8th generation RAM DDR4 min 8 GB SSD min 256 GB Standard input output system

Unified Modelling Language (UML) was used to model the hotel management information system being developed. The UML diagram used in this research was a use case diagram. The next stage was the implementation of the development of a hotel management information system using the Laravel framework and MySQL database. System architecture testing used Apache JMeter software by measuring several parameters, namely latency, throughput, packet loss, and delay. Apache Jmeter software was chosen because it can carry out load tests and stress tests well[6]. The analysis was carried out by comparing the results of parameter measurements on monolithic and microservices architectures. Docker desktop software was used to develop microservices architecture. Testing was carried out on a computer with Windows 11 64-bit operating system, Intel(R) Core(TM) i7-8700 CPU @ 3.20GHz, 16 GB RAM and 256 GB SSD.

3 Results

The UML diagram used in this research was a use case diagram. To create it, an interview was previously conducted with the manager of the integrated hospitality teaching factory at

Politeknik Negeri Jember to get an idea of the system that will be developed. The use case diagram is shown in **Fig. 2**, which consists of managing customer data, managing food and drinks, managing laundry types, managing rooms, managing room types, managing food and drink orders, managing laundry orders, and managing room reservations.

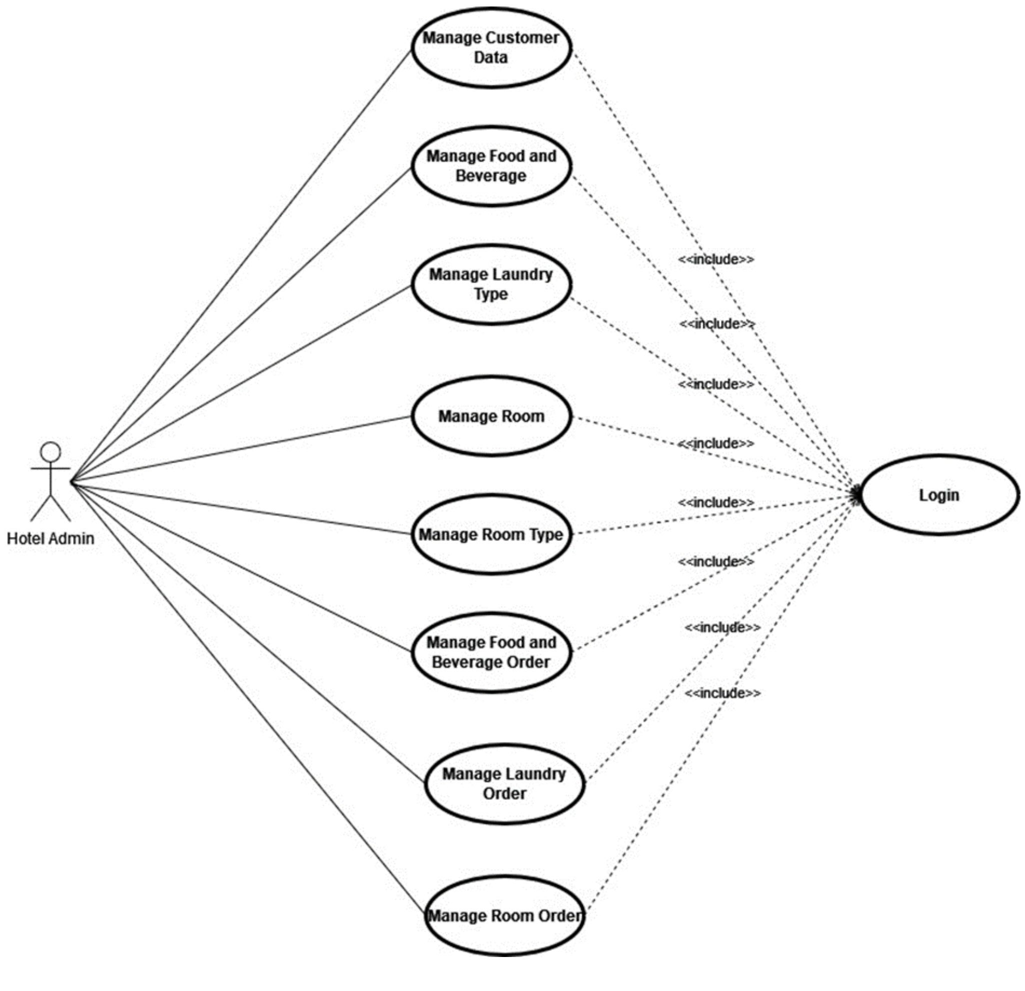

Fig. 2. Use case diagram hotel management system

Furthermore, the applications that have been developed are implemented on two different system architectures, namely monolithic and container-based microservices. To implement the system using a microservices architecture, the docker-compose file is used which is shown in Figure 3. Based on program figure 3, there are four types of microservices, namely web server, PHP, database, and php-myadmin.

Docker desktop software is used to run this microservices architecture. Docker Desktop is secure, ready-to-use containerization software that offers developers and teams a powerful hybrid toolkit for building, sharing, and running applications anywhere [7].

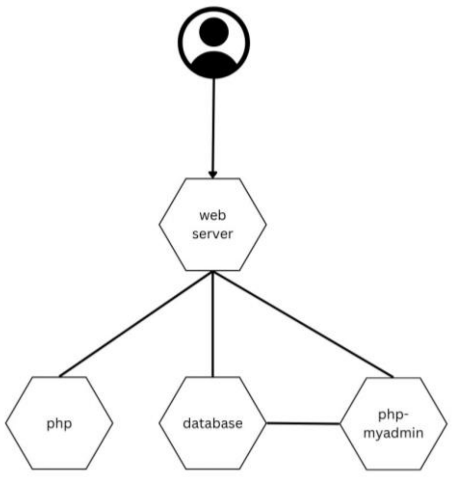

Fig. 3. Architecture of microservices technology used in the hotel management system.

The next step is to carry out architectural system testing. Testing is carried out locally, which is different from that where testing is carried out by [8] in a cloud computing environment. The testing activity was carried out using Apache Jmeter software with latency, throughput, packet loss, and delay parameters. The test results are shown in Figures 4 to 7. Figure 4 shows a comparison of latency measurement results between monolithic and microservices. Monolithic architecture has better latency values compared to microservices architecture. The same results are also obtained for the throughput parameters shown in Figure 5, and respectively in Figures 6 and 7. The results of measuring all parameters show that monolithic architecture is better than microservices architecture.

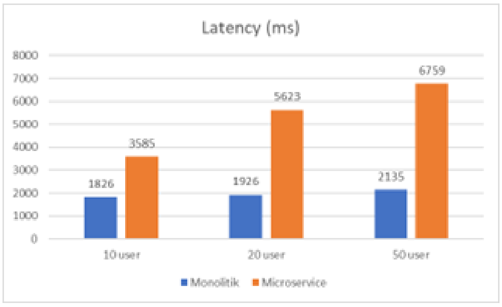

Fig. 4. Comparison latency parameter between monolithic and microservices architecture

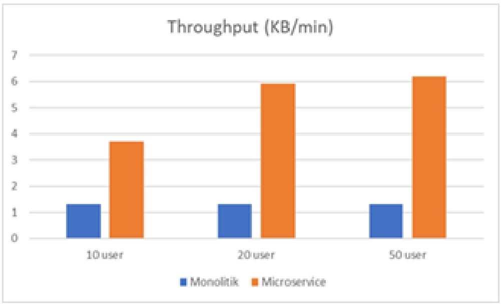

Fig. 5. Comparison throughput parameter between monolithic and microservices architecture

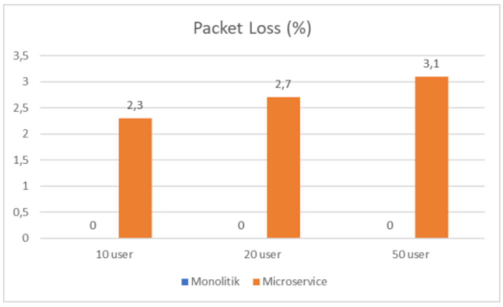

Fig. 6. Comparison packet loss parameter between monolithic and microservices architecture

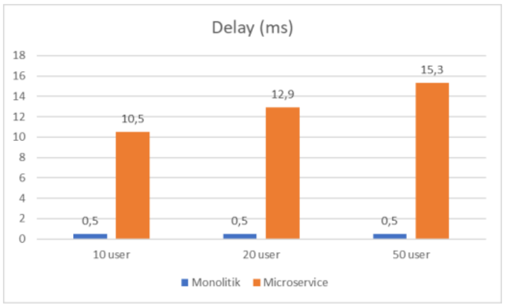

Fig. 7. Comparison delay parameter between monolithic and microservices architecture

4 Discussion

In the test results, it was found that the monolithic architecture had better values for all parameters measured, namely latency, throughput, packet loss, and delay. This is because the microservice architecture has not been optimized and only relies on basic settings. Moreover, microservice architecture is implemented in containers that have small computing resources. A form of optimization that can be carried out on a microservice architecture is the addition of load balancing and scaling, both manual and automatic as conducted by [9]. By scaling the quality of service can be improved even though the resources owned by the container are small because by scaling you will get a larger number of containers, especially in horizontal scaling. In this research, the microservice only consists of four, namely web server, databa se, PHP, and PHP-my admin, and has not implemented event-driven architecture as done by[10].

5 Conclusion

Microservice architecture offers several advantages over monolithic architecture, namely, solving complex problems, each service can be developed independently without any dependency on other parties and can be scaled. However, in this study, the monolithic architecture had better parameter values because the microservices had not been optimized. Future research can implement load balancing and scaling in this hotel management information system application.

Acknowledgment

A big thank you to the Politeknik Negeri Jember for providing funds to conduct this research through PNBP (*Penerimaan Negara Bukan Pajak*/Non-Tax State Revenue) funding sources.

References

[1] F. Tapia, M. ángel Mora, W. Fuertes, H. Aules, E. Flores, and T. Toulkeridis, “From monolithic systems to microservices: A comparative study of performance,” *Appl.*

Sci., vol. 10, no. 17, 2020, doi: 10.3390/app10175797.

[2] W. M. C. J. T. Kithulwatta, K. P. N. Jayasena, B. T. G. S. Kumara, and R. M. K. T. Rathnayaka, "Integration With Docker Container Technologies for Distributed and Microservices Applications," *Int. J. Syst. Serv. Eng.*, vol. 12, no. 1, pp. 1–22, 2022, doi: 10.4018/ijssoe.297136.

[3] C. Pahl, A. Brogi, J. Soldani, and P. Jamshidi, "Cloud container technologies: A state-of-the-art review," *IEEE Trans. Cloud Comput.*, vol. 7, no. 3, pp. 677–692, 2019, doi: 10.1109/TCC.2017.2702586.

[4] A. Bhardwaj and C. R. Krishna, "Virtualization in Cloud Computing: Moving from Hypervisor to Containerization—A Survey," *Arab. J. Sci. Eng.*, vol. 46, no. 9, pp. 8585–8601, 2021, doi: 10.1007/s13369-021-05553-3.

[5] C. Richardson and F. Smith, "Microservices - From Design to Deployment," *Nginx*, p. 80, 2016.

[6] D. I. Permatasari, "Pengujian Aplikasi menggunakan metode Load Testing dengan Apache JMeter pada Sistem Informasi Pertanian," *J. Sist. dan Teknol. Inf.*, vol. 8, no. 1, p. 135, 2020, doi: 10.26418/justin.v8i1.34452.

[7] Anonim, "Docker Desktop: The # 1 containerization software for developers and teams The fastest way to containerize applications," Docker Inc. [Online]. Available: https://www.docker.com/products/docker-desktop/

[8] G. Blinowski, A. Ojdowska, and A. Przybylek, "Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation," *IEEE Access*, vol. 10, pp. 20357–20374, 2022, doi: 10.1109/ACCESS.2022.3152803.

[9] H. Suryotrisongko, "Arsitektur Microservice untuk Resiliensi Sistem Informasi," *Sisfo*, vol. 06, no. 02, pp. 231–246, 2017, doi 10.24089/j.sisfo.2017.01.006.

[10] H. F. Oliveira Rocha, *Practical Event-Driven Microservices Architecture*. Ermesinde, Portugal: Apress, 2022. doi: 10.1007/978-1-4842-7468-2.