ISSN 1816-6075 (Print), 1818-0523 (Online)
Journal of System and Management Sciences
Vol. 13 (2023) No. 3, pp. 205-218
DOI:10.33168/JSMS.2023.0314

Implementations of Microservice on Self-service Application Using Service Oriented Modelling and Architecture: A Case Study

Hestu Widyo Hutomo, Abba Suganda Girsang

Computer Scient Department, Binus Graduate Program, Bina Nusantara University, Jakarta, Indonesia 11530

hestuwidyo@gmail.com, agirsang@binus.edu

Abstract. The impact of the pandemic has accelerated the pace of digital transformation, one of the sectors affected is self-service technology (SST), which allows the company's business activities to continue and can be accessed by customers without time and place restrictions. The increasing number of SST users, this is the basis for making applications that have good performance by comparing application development methods based on Monolithic Architecture (MA) with Microservice Architecture (MSA) using three Quality Attributes (QAs) based on the basic principles of Service Oriented Architecture (SOA) with the Service Oriented Modeling and Architecture (SOMA) method, namely scalability, performance, and availability. The scalability factor of the application becomes a reference because the application of MA has drawbacks in terms of scaling which requires quite a long time. Due to these deficiencies of MA, the application of MSA is a solution, especially by using containerization with Kubernetes as an orchestration platform that is deployed on a public cloud. Testing was carried out on QAs using Apache JMeter. From the test results, it can be concluded that MSA has high flexibility, able to accommodate large user requests. Moreover, it has good availability and agile development that can follow business needs.

Keywords: Digital Transformation, self-service technology, Monolithic Architecture, Microservice Architecture, Quality Attributes, Kubernetes, SOA, SOMA

1. Introduction

The pandemic that hit the world in early 2020 had impacted all economic activities, according to (Nousopoulou et al., 2022) business sectors affected by the pandemic have realized the importance of implementing digital transformation into their businesses. Digital transformation consists of at least two dimensions (Katsamakas, 2022) namely the application of digital technology and the level of transformation that focuses on products, business processes and business models. The application of self-service technology (SST) can improve operational efficiency and reduce labor costs (Hsu et al., 2021). The development of SST can also act as a channel that can be accessed without time restrictions by customers, making SST a new revenue channel for companies.

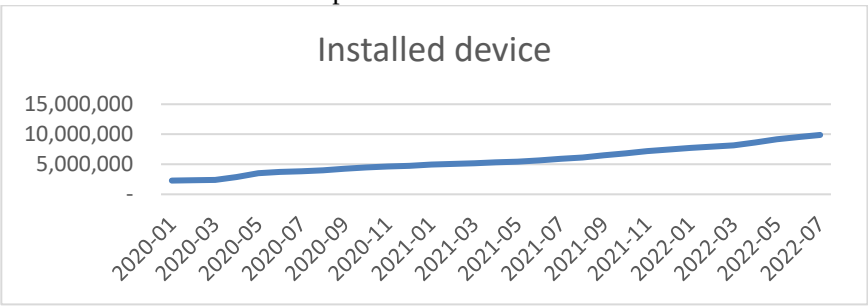


Fig. 1: Total downloaded user on Google Play.

Based on the graph in Fig. 1, users of applications obtained from the Google Play Store experienced a significant spike. Accordingly, the need for applications that have high availability becomes important, to satisfy the potential users of these applications. Previous application developments using MA encountered the following problems:

- Application scalability is less flexible, because the characteristic of MA does not support scalability.
- The use of large infrastructure resources, which to support business needs, a number of servers is added to support services.
- Application performance is not optimal due to the limited number of services that can process requests.
- The application development takes a longer time, due to complexity increase in business and service requirements, which must be developed in a single and comprehensive service.

Evolved from the above problems, flexible scalability is needed to meet the growing needs of users. In the application of MA, scalability is very difficult to do considering the basic concept of MA which is based on a single code, it is required to make a complete single code replication to perform scalability. This will have an impact on inefficient allocation of infrastructure resources. According to (Georgios et al., 2021) many enterprises decide to use a hybrid scheme in resource use to get good efficiency, as well as guaranteed infrastructure availability. The performance of the application is also one of the important parameters for the reliability of a service that is made to meet dynamic business needs. Based on the literature study conducted(Li et al., 2021), identified six Quality Attributes (QAs) that are most considered in the development of MSA, namely: scalability, performance, availability, monitorability, security, and testability.

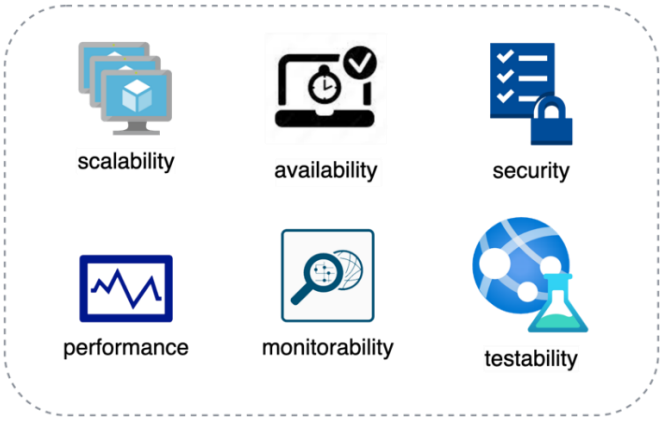
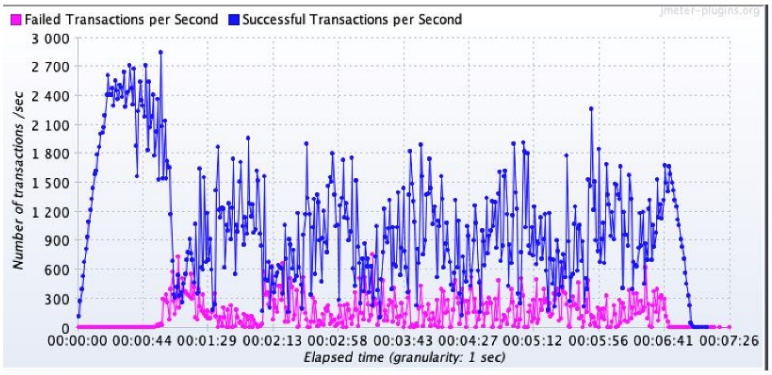


Fig. 2: QAs on microservices architecture.

The results of the load test using Apache JMeter and K6.io software on one of the purchasing services made based on MA, obtained the following information:



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received K...	Sent KB/sec	Avg. Bytes
/api/v1/...	106879	754	7	10568	1126.62	12.40%	254.8/sec	267.22	66.39	1074.0
/api/v1/...	105886	741	7	10926	1112.37	11.77%	252.9/sec	1636.85	195.50	6627.8
/api/v1...	104856	758	7	10606	1118.15	11.85%	250.7/sec	277.60	188.91	1133.7
/api/v1/...	103807	1652	7	39208	3398.84	12.76%	237.0/sec	242.51	591.50	1047.8
/api/v1/...	102369	1682	7	38030	3358.28	13.58%	231.0/sec	239.13	176.08	1059.9
TOTAL	523797	1111	7	39208	2335.22	12.47%	1175.9/...	2527.09	1180.34	2200.7

Fig. 3: Result load test.

Source: Researcher's Processed Data, 2022

The tests carried out as shown in Fig. 3 are based on testing using a simulation of 6,000 users, with the following test results:

- Total transaction per second (TPS) = 1117 TPS
- Average success rate (SR) = 88% SR
- Average response time = 1,11 second

By applying the MSA concept to the self-care application development, it is expected to increase availability, scalability, and improve performance based on SOA principles using the Service Oriented Modelling and Architecture (SOMA) method that focuses on Modularity and granularity with a relatively shorter development time. With limitations on the development of SST applications development with MA-

based applications within the scope of XYZ companies engaged in the telecommunications sector.

2. Related Works

2.1. Self Service Technology

Self-service applications have recently become increasingly popular along with rapid technological advances. In companies engaged in public service providers, it is an obligation to provide service help centers or physical outlets that are devoted to handling problems with the services provided to customers. This requires a large amount of money if not managed properly. Digital transformation (DT), which is currently developing, changes the conventional organizational design into a digital business ecosystem. The application of self-service technology (SST) within the company is expected to have a large impact on company expenses and customer satisfaction, as said (Hsu et al., 2021) that SST provides benefits to companies by increasing operational cost efficiency and reducing the number of workers, as well as reducing errors due to manual processes.

2.2.Monolithic Architecture

Monolithic architecture (MA) is a single software architecture in its implementation, which usually generally has three parts in its development, namely:

- Presentation: which is the top layer of an application, both the interface of the software and the response API.
- Business logic: In this layer all application logic is stored and executed according to application routines.
- Database: data storage layer or configuration of the application.

Usually, software built using this architecture has advantages in terms of development, because it only involves internal processes in communicating and managing data, either storing, changing, or deleting which is commonly called Create-Read-Update and Delete (CRUD). According to (Lauretis, 2019) MA is very easy to develop, test and deploy. However, when software becomes more complex, MA will tend to be more complicated in its development and operation.

2.3. Microservices Architecture

The application of the SOA concept in MSA design is very suitable for use in medium to large companies, because MSA can support complex software development. According to (Razzaq, 2020) MSA is at the heart of the internet of things (IoT) as an independent service. In the development of the concept of making a software framework, previously known as monolithic, which is commonly encountered in the application of software frameworks. Nowadays, it is increasingly being abandoned along with the development of the concept of a software development framework using the concept of microservices. In the application of the MSA concept (Rosen et al., 2008) the special characteristics of SOA consist of:

- Modularity and granularity.
- Encapsulation.
- loose coupling.
- Isolation of responsibility.
- Autonomy.

- Reuse.
- Dynamic discovery and binding.
- Stateless.
- Self-describing.
- Composable.
- Govern by policy.
- Independent of location, language, and protocol.

2.4.Virtualization

To support MSA, choosing the platform on which the application runs is important, especially the concept of MSA is more often referred to as containerization. Containerization is the development of OpenStack technology or virtualization of an operating system with minimal specifications only to be able to run software that has been made to be accessed individually. Containers and microservices are very closely related, nowadays almost every software development design uses the concept of microservices using containerization to run it. According to (Kim et al., 2021) Kubernetes is a container orchestration platform that is widely used in cloud computing, besides the open-source licence makes Kubernetes very popular.

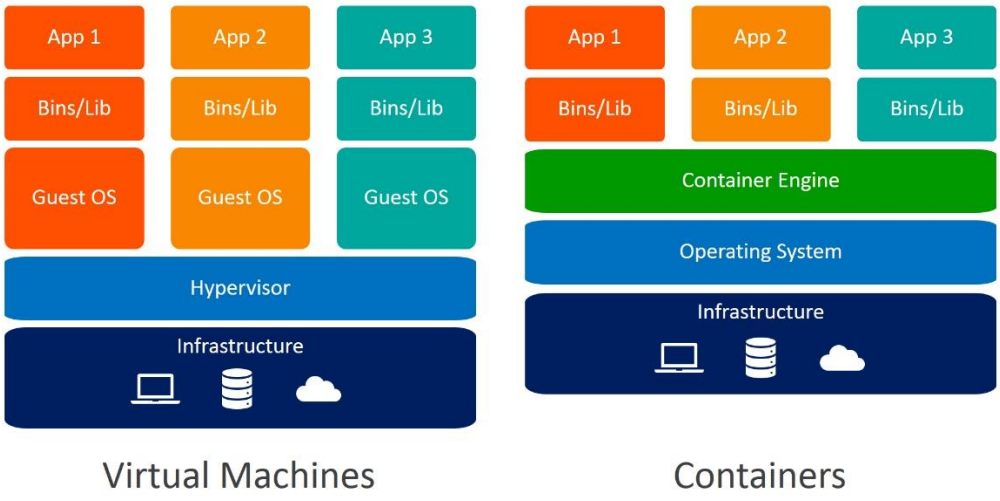


Fig. 4: Difference architecture between VM and containers.

Fig. 4 describes the architectural differences between Virtual Machine (VM) based virtualization and containers, the containers architecture shares each other at the operating system (OS) level while the VM architecture has its own OS in each application. One of the main innovations that can create scalability in the use of resource computing is cluster autoscaler (Tamiru et al., 2020) .Cluster autoscaler can allocate resource computing, fluctuating based on the workload of the active user number.

2.5.Service Oriented Modelling Architecture

SOMA was first published by IBM in 2008 through the IBM Systems Journal. SOMA is a software development lifecycle (SDLC) method developed by IBM for SOA-based software design. The SOMA

method focuses on how an application is made based on the running business side. Broadly speaking, SOMA (Arsanjani et al., 2008) consists of seven main phases as shown in Figure 4. depicting a patterned successive iteration process. However, SOMA phases are not linear, each phase is implemented in a risk-based, iterative, and gradually.

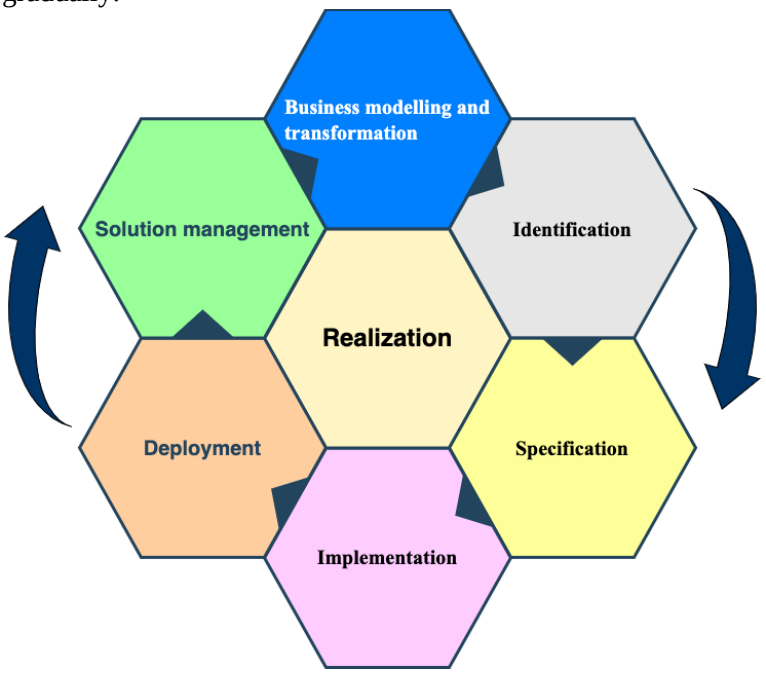


Fig. 5: SOMA fractal model of software development.

From Fig. 5, the seven main phases of the SOMA method are explained in detail which include: business modelling and transformation, solution management, identification phase, specification phase, realization phase, implementation phase, and deployment.

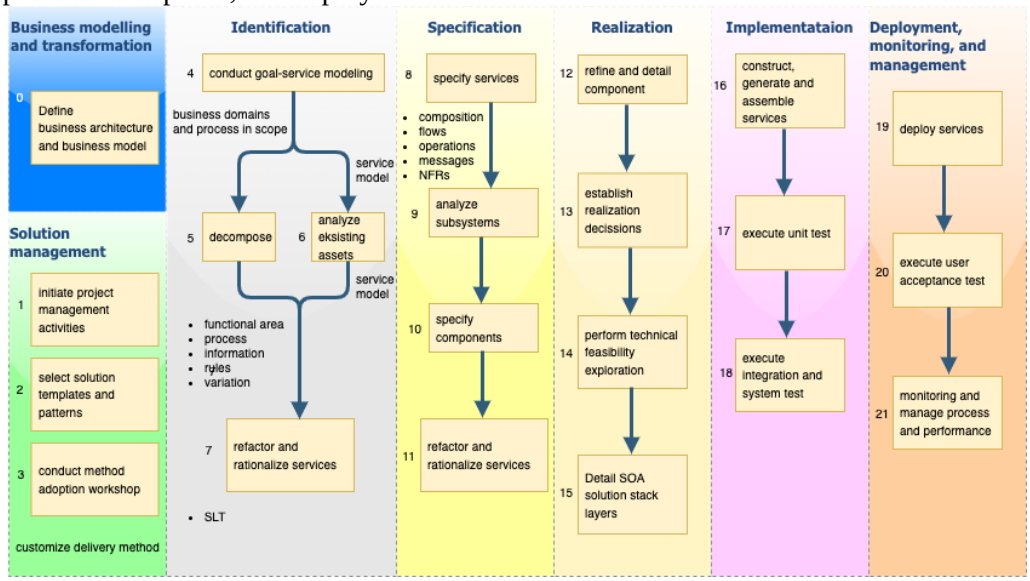


Fig. 6: SOMA lifecycle.

Fig. 6 explain in detail the seven main phases of the SOMA method which include: business modelling and transformation, solution management, identification phase, specification phase, realization phase,

implementation phase, and deployment.

3. Research Methodology

This research is divided into three phases, which are: planning or identification, specification or preparation, and development. First phase is to identify the problems, define the purposes and the scope of the research. Next is to comprehend the architecture and current service development methods in the scope of enterprise using SOA concept and MSA architecture design approach, focusing on six points of QAs as evaluation matrix to achieve satisfying results. Then, to prepare the needs for the research before doing the experiment. After preparation is the implementation phase of the methods related to this research in which some tests or experiments are executed to measure the performance and scalability of the services. Finally, it is time to make a conclusion and recommendations at the end of the research.

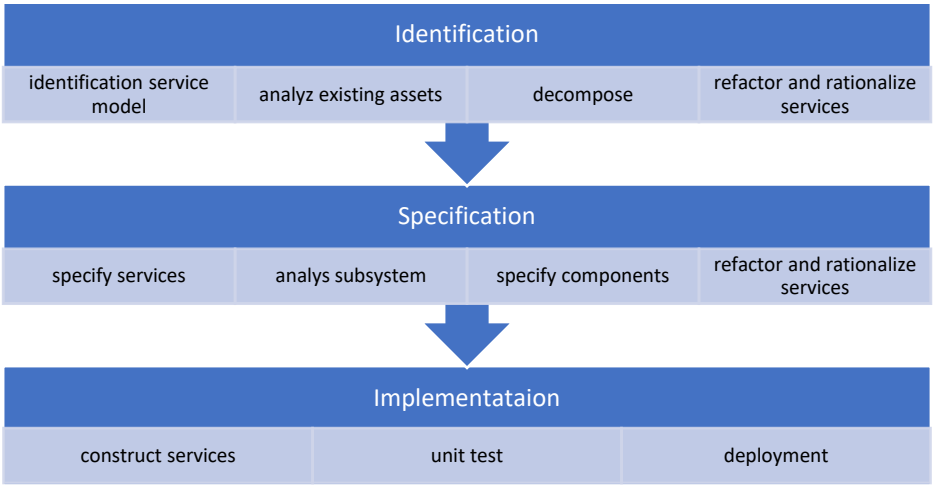


Fig. 7: Research Methodology.

3.1. Quality Attributes, Performance and Scalability

To be able to compare between the two architectures, QAs are needed to be able to describe the comparison between the two architectures (Blinowski et al., 2022) using three QAs specifications: scalability, performance, and availability.

3.2.Scalability

In general, scalability testing determines the maximum number of users that can be handled by the application. This is related to cost (Cheng et al., 2022), the application of hybrid cloud can increase efficiency in resource use to support applications. MSA deployments make it possible to deploy multi clouds. Because recently the need for business is increasingly dynamic, a reliable design for software development is expected to meet these needs. Therefore, these aspects must be planned and implemented thoroughly. Seeing the predecessors in software development using the MA method, it is very difficult to create reliable software, because monolithic concepts are usually implemented in a virtual machine or hardware separately, making it difficult to configure and add instances to be able to serve unexpected increasing requests.

3.2.1. Performance

Performance in MSA is a measure of the system's ability to meet time requirements in response to a request.

Performance measurement of a service can be calculated based on the time unit parameter in a request, this can be applied both in MA and MSA. (De, 2017) Application programmable interfaces (APIs) are no longer seen as mere integration mechanisms but have become the main route for the delivery of data and services to users through various digital channels. In testing the performance of a service, there are several types of tests that are often used, including:

- Baseline testing.
- Load testing.
- Stress testing.
- Soak testing.
- Automation testing.

In accordance with (Akbulut & Perros, 2019) Business interest in MSA is increasing since the MSA brings a lightweight, independent, reuse-oriented, and fast service deployment approach that minimizes infrastructural risks. Automation tests become better and more efficient and can provide convenience in carrying out a series of tests, especially tests that are often carried out to ensure changes do not affect other services. This will reduce the duration of testing, and the tester of the application only needs to focus on the new changes.
Several applications that can be used in testing MA and MSA based on SOA that are often used in testing, includes:

- Apache JMeter.
- K6.io
- SoapUI.
- Vegeta load testing.
- Blaze Meter.

3.2.2. Availability

Calculation of availability called Success Rate (SR) is generally displayed in percentage units (%). How to measure SR can be done in the same way as in equality 1:

$$SR = \frac{Success\ transaction}{Total\ transaction}\ x\ 100\% \quad \text{(Eq. 1)}$$

4. Results and Discussion

After implementing the SOA concept with SDLC method using SOMA in designing new systems, comparisons between the old and the new methods are needed to get an accurate and systematic analysis result. If there is any assumption-based analysis, it only applies for synchronous services only. The specification of a large target user application should be designed carefully since it might bring impact to the next architecture and development. MSA system provides more convenience during the system design and development.

4.1.Performance Test

Performance testing has a vital role (Németh & Sótér, 2021), doing the test during the system specification has been challenging (Hossain, 2018) as it needs more efforts during the time. In spite of additional resources, cost is also an important factor during specification (Baransel et al., 2021) because of the use of resources to execute the testing. It brings a good impact to the specification and development. In this case, testing load provides information of capacity which can be processed at one time to the system. As the result of the testing that has been done to the system, Fig. 8 shows the performance test of the new system.

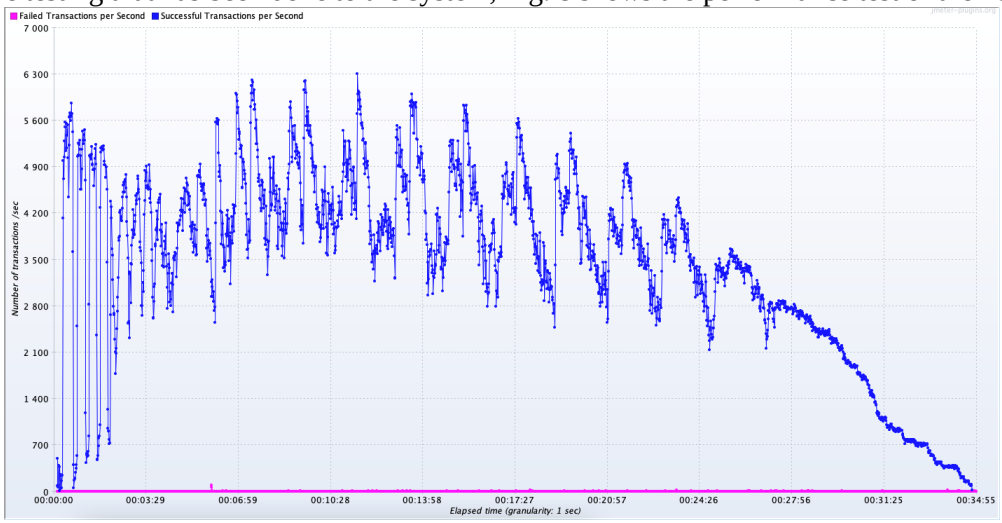


Fig. 8: Result Performance Test.

4.1.1. Stress Testing

The result of load test as shown at Fig. 9 shows the system capability to receive the ideal number of traffic at one time before a system failure happens.

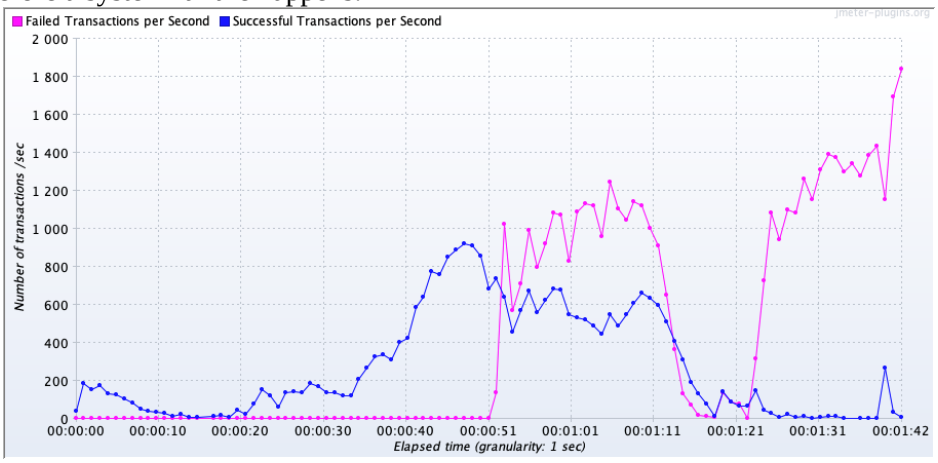


Fig. 9: Result Stress Test.

4.1.2. Soak Testing

There is also a longer period of testing that is executed to test (Khurshid et al., 2021) the system ability in handling real traffic assuming the number of active users. This test is also known as soak test due to the long duration of the test.

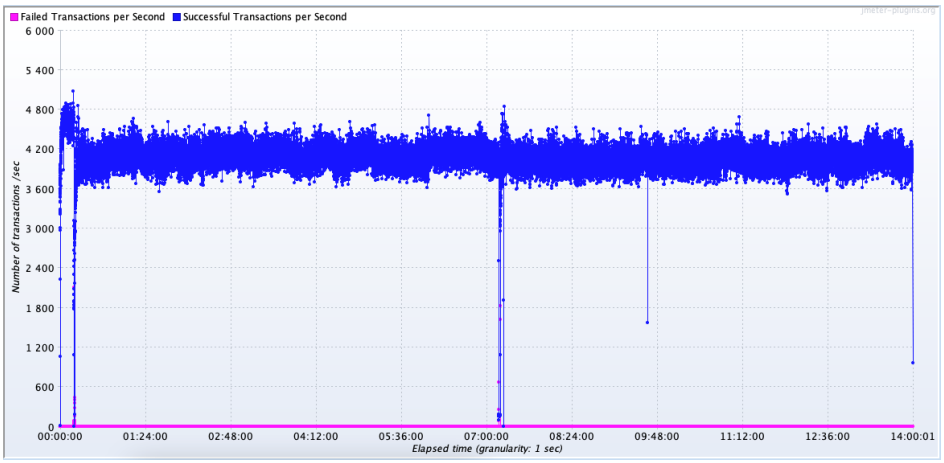


Fig. 10: Result Soak Test.

Based on a 14-hour soak test, the result is shown in Fig. 10, in which a pattern of increasing errors can be shown on overall testing. It can be a reference to create a system scale to avoid another system error in the future.

4.1.3. Automation Testing

During a system development, the testing on the system should be adequate to make sure the system runs well. In the research (Zafar et al., 2022) concluded that the implementation of a combinatorial testing between manual and automation testing improves the efficiency of the test suite which brings such a good impact like shorter testing execution time and faster validation process of the features developed in the system.

4.2.Scalability

Scalability is a measurement of a system's ability to add resources used to handle some amount of requests in a system. In this way, scalability plays an important role in a system. One of them is horizontal scalability which can help reduce the operational costs (Perri et al., 2022) and guarantees high fault tolerance and the ability to scale its computational power according to the number of incoming requests from users.

4.3.Cost Base Analysis

A good specification of a system should enclose the total cost ownership (TCO). TCO can be used by a developer as a report to the high-level management. The technique to a more efficient offloading is as explained by (Ali & Iqbal, 2022) which is to divide them into smallest parts or microservices.

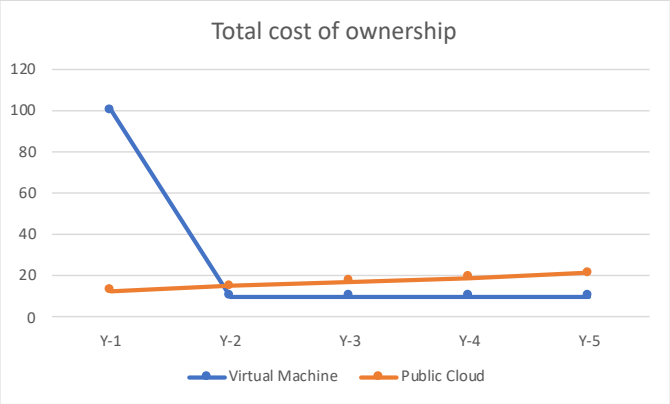


Fig. 11: TCO Yearly.

Source: Researcher's Data and Projection Plan, 2022

Fig. 11 describes the total investment projection needed in the use of compute resources. There is a significant difference in the first year if using virtual machine resources. Meanwhile, the use of public cloud is gently sloping, the total usage is stable and adjustable as the need and load of the service.

4.4.Discussion

In reference to the result of this study, MSA development will impact the infrastructure and platform used. This is related to resource availability and capability to scale and handle dynamic requests from end users. The role of platforms, such as Kubernetes, provide solutions to scaling problems which will have an impact on increasing the availability of the applications. To give it best performance, the test on application ability must be done, to decrease system failure in the production stage.

Table. 1: Result Analysis.

Previous system					New system				
Function	Request sent	SR %	Max tps	Throughput	Function	Request sent	SR %	Max tps	Throughput
login	106.979	88,6	2.795	255 /s	get token	100.304	99,98	6.291	433 /s
					validate token	102.201	99,4	6.291	433 /s
					logout	100.139	99,98	6.291	433 /s
dashboard	105.886	88,33	2.795	255 /s	quota	100.142	99,98	6.291	433 /s
					balance	100.108	99,98	6.291	433 /s
					vas	108.021	99,99	6.291	433 /s
					balance transfer	100.211	99,98	6.291	433 /s
store	104.856	89,15	2.795	255 /s	list product	105.025	99,99	6.291	433 /s
					purchase	100.704	99,99	6.291	433 /s
					payment	100.031	99,99	6.291	433 /s

profile	103.807	88,34	2.795	255 /s	add	105.008	99,99	6.291	433 /s
					update	100.109	99,99	6.291	433 /s
					get profile	100.021	99,99	6.291	433 /s
care	102.369	87,42	2.795	255 /s	chat	100.341	99,99	6.291	433 /s
					faq	103.631	99,99	6.291	433 /s
					store location	109.302	99,99	6.291	433 /s

Source: Researcher's Processed Data, 2022

According to Table 1, the data shows that MSA implementation can improve the application performance. It brings a good impact to the application performance which runs above it. Comparison between MA and MSA architecture to the application performance is aimed to accommodate larger requests.

4.4.1. Resource Consumption

The use of previous Virtual Machine resources is quite different from the public cloud principle because the public cloud resource fee will be charged at the first time of resource provisioning, even if the resource is not being used or idle (Chaurasia et al., 2021). It becomes one factor to consider at the development stage which can be projected through performance test data (Aldossary et al., 2019) combined with projected application user numbers.

Table. 2: Autoscaling comparation.

	Monolithic Architecture			Microservice Architecture		
Virtual Users	cpu	memory	SR	cpu	memory	SR
500	32	64	100%	4	8	100%
1.000	32	64	100%	4	8	100%
1.500	32	64	100%	6	10	100%
2.000	32	64	99,80%	12	24	99,99%
2.500	32	64	98,07%	12	24	100%
3.000	32	64	95,80%	16	32	99,97%
3.500	32	64	91,23%	16	32	100%
4.000	32	64	90,83%	24	48	99,78%
4.500	32	64	87,42%	24	48	100%
5.000	32	64	85,74%	32	48	99,82%
5.500	32	64	80,01%	32	48	100%
6.000	32	64	73,70%	32	48	100%

Source: Researcher's Processed Data, 2022

According to Table. 2 the result of scalability testing using JMeter comparing the use of VM infrastructure's resources experienced a very large decrease in SR due to limited additional resources which took time to prepare. Meanwhile, microservices can easily scale to receive incoming traffic.

5. Conclusion and Future Work

The implementation of MSA is much more complex than MA development, but if the execution is right and is using containerize with Kubernetes platform which has auto-scaling feature, the result will be an agile system development which has good performance. Kubernetes platform requires the availability of certain infrastructure, so that the use of public cloud is also a key to satisfying performance results. The implementation of performance testing can also increase the level of reliability of the product being developed because it will increase the potential for system failure. The implication of MSA to enterprise companies is ideal because it will reduce excessive costs in purchasing Enterprise Service Bus product licences which are widely offered on an enterprise scale.

Acknowledgements

The authors would like to thank everyone, grateful to all of those with whom I have had the pleasure to work during this and other related projects has provided me extensive personal and professional guidance and taught me a great deal about both scientific research and life in general.

References

Akbulut, A., & Perros, H. G. (2019). Performance Analysis of Microservice Design Patterns. IEEE Internet Computing, 23(6), 19–27. https://doi.org/10.1109/MIC.2019.2951094

Aldossary, M., Djemame, K., Alzamil, I., Kostopoulos, A., Dimakis, A., & Agiatzidou, E. (2019). Energy-aware cost prediction and pricing of virtual machines in cloud computing environments. Future Generation Computer Systems, 93, 442–459. https://doi.org/10.1016/j.future.2018.10.027

Ali, A., & Iqbal, M. M. (2022). A Cost and Energy Efficient Task Scheduling Technique to Offload Microservices Based Applications in Mobile Cloud Computing. IEEE Access, 10, 46633–46651. https://doi.org/10.1109/ACCESS.2022.3170918

Arsanjani, A., Ghosh, S., Allam, A., Abdollah, T., Ganapathy, S., & Holley, K. (2008). SOMA: A Method for Developing Service-Oriented Solutions. IBM Systems Journal, 47(3), 377–396. https://doi.org/10.1147/sj.473.0377

Baransel, B. A., Peker, A., Balkis, H. O., & Ari, I. (2021). Towards Low Cost and Smart Load Testing as a Service Using Containers (pp. 292–302). https://doi.org/10.1007/978-3-030-71711-7_24

Blinowski, G., Ojdowska, A., & Przybylek, A. (2022). Monolithic vs. Microservice Architecture: A Performance and Scalability Evaluation. IEEE Access, 10, 20357–20374. https://doi.org/10.1109/ACCESS.2022.3152803

Chaurasia, N., Kumar, M., Chaudhry, R., & Verma, O. P. (2021). Comprehensive survey on energy-aware server consolidation techniques in cloud computing. The Journal of Supercomputing, 77(10), 11682–11737. https://doi.org/10.1007/s11227-021-03760-1

Cheng, W., Feng, H., & Liang, G. (2022). Design of IT Infrastructure Multicloud Management Platform Based on Hybrid Cloud. Wireless Communications and Mobile Computing, 2022, 1–12. https://doi.org/10.1155/2022/9227948

De, B. (2017). API Management. Apress. https://doi.org/10.1007/978-1-4842-1305-6

Georgios, C., Evangelia, F., Christos, M., & Maria, N. (2021). Exploring Cost-Efficient Bundling in a Multi-Cloud Environment. Simulation Modelling Practice and Theory, 111, 102338. https://doi.org/10.1016/j.simpat.2021.102338

Hossain, Md. S. (2018). Challenges of Software Quality Assurance and Testing. International Journal of Software Engineering and Computer Systems, 4(1), 133–144. https://doi.org/10.15282/ijsecs.4.1.2018.11.0044

Hsu, P.-F., Nguyen, T. K., & Huang, J.-Y. (2021). Value Co-Creation and Co-Destruction in Self-Service Technology: A Customer's Perspective. Electronic Commerce Research and Applications, 46, 101029. https://doi.org/10.1016/j.elerap.2021.101029

Katsamakas, E. (2022). Digital Transformation and Sustainable Business Models. Sustainability, 14(11), 6414. https://doi.org/10.3390/su14116414

Khurshid, S., Shrivastava, A. K., & Iqbal, J. (2021). Effort based software reliability model with fault reduction factor, change point and imperfect debugging. International Journal of Information Technology, 13(1), 331–340. https://doi.org/10.1007/s41870-019-00286-x

Lauretis, L. De. (2019). From Monolithic Architecture to Microservices Architecture. 2019 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 93–96. https://doi.org/10.1109/ISSREW.2019.00050

Li, S., Zhang, H., Jia, Z., Zhong, C., Zhang, C., Shan, Z., Shen, J., & Babar, M. A. (2021). Understanding and Addressing Quality Attributes of Microservices Architecture: A Systematic Literature Review. Information and Software Technology, 131, 106449. https://doi.org/10.1016/j.infsof.2020.106449

Németh, G. Á., & Sótér, P. (2021). Teaching performance testing. Teaching Mathematics and Computer Science, 19(1), 17–33. https://doi.org/10.5485/TMCS.2021.0518

Nousopoulou, E., Kamariotou, M., & Kitsios, F. (2022). Digital Transformation Strategy in Post-COVID Era: Innovation Performance Determinants and Digital Capabilities in Driving Schools. Information, 13(7), 323. https://doi.org/10.3390/info13070323

Perri, D., Simonetti, M., & Gervasi, O. (2022). Deploying Efficiently Modern Applications on Cloud. Electronics, 11(3), 450. https://doi.org/10.3390/electronics11030450

Razzaq, A. (2020). A Systematic Review on Software Architectures for IoT Systems and Future Direction to the Adoption of Microservices Architecture. SN Computer Science, 1(6), 350. https://doi.org/10.1007/s42979-020-00359-w

Rosen, M., Lublinsky, B., T. Smith, K., & J. Balcer, M. (2008). Applied SOA: Service-Oriented Architecture and Design Strategies. Wiley Publishing, Inc.

Tamiru, M. A., Tordsson, J., Elmroth, E., & Pierre, G. (2020). An Experimental Evaluation of the Kubernetes Cluster Autoscaler in the Cloud. 2020 IEEE International Conference on Cloud Computing Technology and Science (CloudCom), 17–24. https://doi.org/10.1109/CloudCom49646.2020.00002

Zafar, M. N., Afzal, W., & Enoiu, E. (2022). Evaluating system-level test generation for industrial software. Proceedings of the 3rd ACM/IEEE International Conference on Automation of Software Test, 148–159. https://doi.org/10.1145/3524481.3527235