IJCSE International Journal of Computer Sciences and Engineering Open Access

Research Paper Vol. 10, Issue.8, August 2022 E-ISSN: 2347-2693

Migration Architecture Monolithic to Microservice on Information Technology Consultant Company

Lutfi Ardiansyah[1*], Yuli Karyanti[2]

[1,2]Dept. of Computer Science and Information Technology, Gunadarma University, Depok, Indonesia

[*]Corresponding Author: lutfi_ardiansyah@ymail.com, Tel.: +62-81281-22136-2

DOI: https://doi.org/10.26438/ijcse/v10i8.914 | Available online at: www.ijcseonline.org

Received: 23/Jul/2022, Accepted: 08/Aug/2022, Published: 31/Aug/2022

Abstract— This research aims to migrate the architecture from a monolithic to a microservices architecture on applications that were originally built with a monolithic architecture by an IT consulting firm. The goal is to migrate from monolithic architecture to microservices architecture to overcome problems that occur in applications with monolithic architectural designs that have been delivered to clients by an IT consulting company to improve customer satisfaction by improving the quality of the application. Microservices is one of the most popular architectural styles today. It is an independent, usable service modeled around a business domain. The advantages of using a microservices architecture in developing systems are flexibility and system maintenance. One method that is widely used in system migration is the Strangler Fig Application. There are 3 main stages: 1. Identifying assets to be relocated; 2. transferring assets; and 3. rerouting relocated assets. The migration results in a monolithic architecture totaling 2491 records consisting of 87 columns taking 2 hours 59 minutes 18 seconds or 1 data point for 4.3 seconds and heap memory of 99.0 percent, while the microservices architecture with an increase in data of 384 records takes 1 minute 33 seconds or 1 data point for 0.03 seconds and heap memory of 12.8 percent after implementation of the new architecture.

Keywords— Microservices, Monolithic, Migration System, Strangler Fig Application

I. Introduction

This research aims to migrate the architecture from a monolithic to a microservices architecture on applications that were originally built with a monolithic architecture by an IT consulting firm. The goal is to migrate from monolithic architecture to microservices architecture to overcome problems that occur in applications with monolithic architectural designs that have been delivered to clients by an IT consulting company to improve customer satisfaction by improving the quality of the application.

Like many older enterprise applications, the FTGO application is a monolith, consisting of a single Java Web Application Archive (WAR) file. Over the years, this has become a large and complex application. Despite the best efforts of the FTGO development team, it became an example of the Big Ball of Mud pattern. Software delivery speed has slowed down. Worse still, the FTGO app has been written using some increasingly outdated frameworks. The FTGO app shows all the symptoms of a monolithic hell [1].

This architectural migration is carried out on one of the applications for project management that were developed using the Java framework, namely Java Spring, with a monolithic architecture and problems that occur when uploading data. The main problems are the decline in performance when uploading as data grows and problems when there are data upload activities in the task module and project. Memory usage will be high, other modules affected will be slow, and worst of all, the application cannot be accessed at all.

Usually, the response time should be as fast as possible, but it's also possible that the computer reacts so quickly that the user cannot keep up with the feedback. For example, a scrolling list can move so fast that the user cannot stop it in time to keep the desired element within the available window. 0.1 seconds is the limit for making the user feel that the system reacts instantly, meaning that no special feedback is required except to display the results [2].

Microservices are independently usable services modeled around a business domain. They communicate with each other over the network, and as an architectural choice, offer many options for solving any problems you may encounter. Therefore, the microservices architecture is based on several collaborating microservices. Migration was carried out using the Strangler Fig Application technique. A technique that is often used when rewriting a system is called "strangler fig application" [3].

II. Related Work

Application of Microservice Architecture in the Management System of Pos Indonesia Polytechnic Project Courses by Mohammad Harry Khomas Saputra and Luthfi

Muhammad Nabil. The problems in this research are the slow process of system development and handling of damage, and the difficulty of the development process and handling of system errors by the development team. The purpose of this research is to make an information system easier to break down into several parts and to make it easier for developers to develop information systems by separating the components of the application. The result of this research is that the effect produced when developing a system on a microservice architecture is less disturbing to other systems when developing to reduce the scope of development [4].

Refactoring Microservice Architecture on PT. Graha Usaha Teknik Attendance Application by Rizki Mufrizal and Dina Indarti. The problem with this research is the difficulty of the application maintenance process, the application performance is decreasing, and the update process is getting more difficult. The purpose of this research is to solve the existing problems. The results of this research are based on the results of software testing using a load test. Microservices architecture can be more optimal than monolithic architecture [5].

Implementation of Microservices Architecture on E-Commerce Web Service by Juan Andrew Suthendra, and Magdalena Ariance Ineke Pakereng. The problem with this research is that the monolithic architecture of the application is more complex and larger, and if one part of the code is changed, it will affect other parts of the code. The result of this research is to use microservices architecture in developing easier system flexibility and maintenance. System performance can also be maximized because it can be built using different programming languages and databases. Service changes and improvements will not affect the work of other services if they are independent of each other. This is important because business processes will continue to evolve, and systems must adapt to changes [6].

Microservices and It's Applications : An Overview by Nupura Torvekar, and Pravin S. Game. The problem with this research is that traditional monolithic architecture was built as a single unit, which was difficult to scale and was not suitable for the development of complex processes. The purpose of this research is to get a pattern that is strong, flexible, and reliable. The result of this research is that the comparative analysis provided can help the readers in the identification of the contribution and an overall view of the utilization of this pattern for further studies. This work can be further enhanced by implementing the micro services for their real-time use cases and analyzing their performance [7].

eGovernment Integration Framework for Fragmented Systems by Sameer S. Paradkar. The problem with this research is interoperability for further development in eGovernment. The purpose of this research is to solve the problem of interoperability. It is resolved during the system design stage by leveraging service orientation, i.e., microservices and API. The result of this research is the proposed framework that delivers several eServices to the government, citizens, and business communities [8].

III. METHODOLOGY

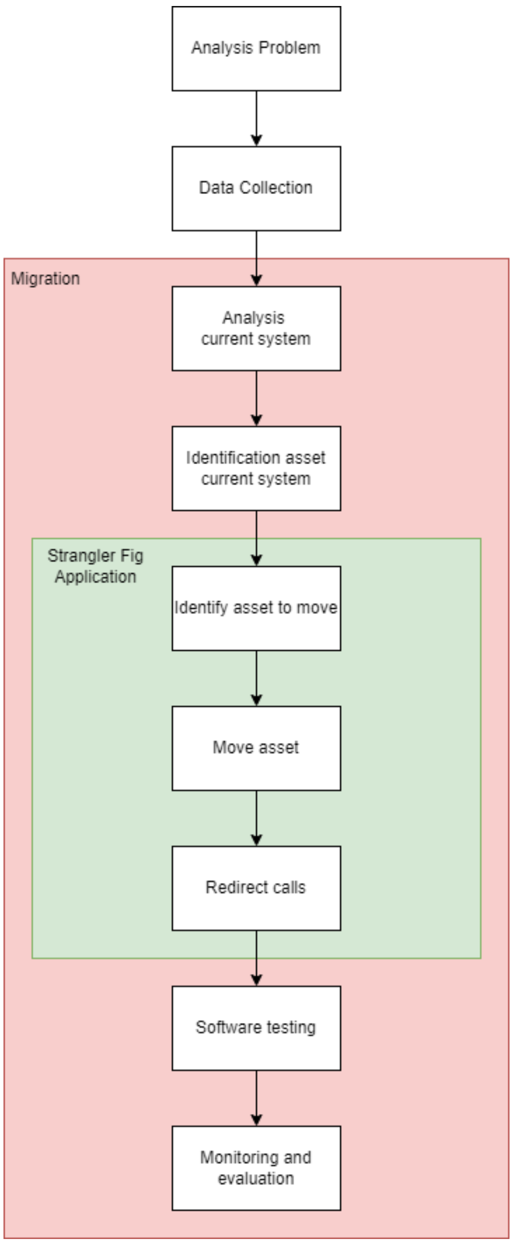

Figure 1. Flowchart of research methodology stages

Figure 1 shows the stages used in this research. consisting of 9 steps, 3 of which are designs from the Strangler Fig Application and 7 of which are migratory stages.

A. Analysis System

To find out what problems exist in the current monolithic architecture, in this research there are 3 problems, namely:

- When there is one module that is experiencing interference, it will have an impact on other modules causing it to be unable to access.
- The existing code will be more complicated and difficult as the existing features increase.

- Decreased data upload performance as the application size increases.

B. Data Collection

This stage is used to collect the data needed in the research. In this research, the data collected is a list of modules that exist in the monolithic application.

Table 1. List modules

No.	Label	Name
1.	Action Logs	action_logs
2.	Activity	activity
3.	Activity Logs	activity_logs
4.	Administration Logs	administration_logs
5.	Approval Task	approval_task
6.	Assignee	assignee
7.	Async Managers	async_managers
8.	Attachment	attachment
9.	Branchs	branchs
10.	Comment	comments
11.	Company	company
12.	Configs	ref_email_list
13.	Cron Configurations	cron_configurations
14.	Cron Logs	cron_logs
15.	Dashboard	dashboard
16.	Dashboard Executive	executive_dashboard
17.	Data Permissions	record_role
18.	Demo	demo
19.	Document Category	category_document
20.	Documents	documents
21.	Flow	flow
22.	Form	form
23.	Generate Numbers	generate_number
24.	Geographic	geographic
25.	Groups	groups
26.	Home	home
27.	Http Request Logs	http_request_logs
28.	Iframe	iframe
29.	Integrate System	Integrate_system
30.	Kanban	kanban
31.	Kota	kota
32.	Locations	locations
33.	Logging File	logging_file
34.	LogSummary	log_summary
35.	Maps	maps
36.	Maps Layers	maps_layer
37.	Master Data	master_data
38.	Member	member
39.	Member Assignee	member_assignee
40.	Menus	menus
41.	Module Permissions	module_permissions
42.	Modules	modules
43.	Notifications	notifications
44.	Outboxs	mail_outbox
45.	Project	project
46.	Project Category	project_category
47.	Project Detail	project_detail
48.	Report Project	report_project
49.	report_detail	report_detail
50.	Sent Items	mail_sentitem
51.	Settings	settings
52.	Status	status
53.	Sub Branchs	sub_branchs
54.	Sub Wilayah	sub_wilayah
55.	Task	task
56.	Task Detail	task_detail
57.	Task Progress	task_progress
58.	Templates	ref_template
59.	Todo	todo
60.	Users	users
61.	Wilayah	wilayah
62.	Workspace	workspace
63.	Workspace Member	workspace_member
64.	Workspace Panel	workspace_panel

C. Analysis Current System

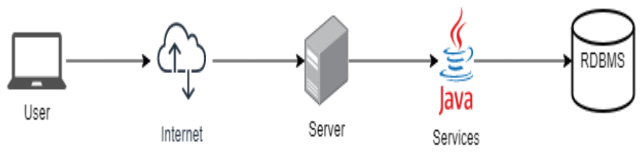

Figure 2. Design architecture monolithic

This stage has entered the stage of system migration, at this stage the aim is to find out an overview of the running system and as a basic reference for building microservices architecture in determining what tools will be used. In this monolithic architecture, this application has a general design as shown in Figure 2. when the user accesses the application, it will be directed directly to a java service, and from the java application it will access the data in the database.

D. Identification Asset Current System

Modules are identified on the running system by querying directly to the module table in the database, which is obtained during the data collection stage. The purpose of this stage is so that no module is missed or left behind when determining which module to move.

E. Identify Asset to Move

After identifying the existing modules on the running system, the next step is to identify which modules will be moved. The determination of which modules will be moved in the case of this application is based on the problems that occur, namely the existence of activities that make the system inaccessible. To find out what activities cause the system to be inaccessible, monitoring is carried out on the server monitoring dashboard. In addition, the determination of which module will be moved is also based on the slowing performance of the module. To see which performance begins to slow down, it can be seen from the length of time it takes to upload several data on the module.

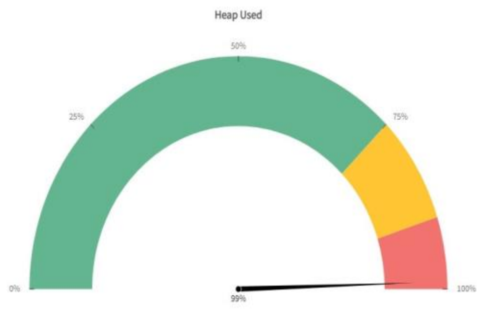

Figure 3. Memory consumption when uploading project

Figure 3 is the condition of the server when there is an upload project, which causes the server heap memory to be 99.0% and causes the application to be inaccessible.

Table 2. Detail data upload project

No.	Name	Description
1.	Columns	87
2.	Records	2491
3.	Start Time	14:57:39
4.	End Time	17:56:57
5.	Total Time	10758 seconds
6.	Record / s	1 record / 4.3 second

Table 2 is detail data of proses upload project totalling 2491 records consisting of 87 columns taking 10758 seconds or 1 data for 4.3 second.

Based on this problem, there are 2 modules to be moved. The first is the project module (project, project_category, project_detail) and the second is the task module (task, task_detail).

Table 3. List of modules to be moved

No.	Label	Name
1.	Project	project
2.	Project Category	project_category
3.	Project Detail	project_detail
4.	Task	task
5.	Task Detail	task_detail

F. *Move Asset*

In this research, the transfer from the current system to the new system requires adjustments first. The transfer of this module is done by creating a new project and moving some of the required source code to the module, because there are additional supporting tools, namely Rabbit MQ as a broker that will be used for communication between services to make it easier. During the module move, some new source code was added for the Rabbit MQ configuration.

G. *Redirect Calls*

After moving the specified modules, the next step is to reroute the split services. This project uses an additional tool, namely Rabbit MQ, to be able to communicate with each other between main services and other services.

There is an additional RabbitMQ supporting library as a message broker with the aim of minimizing memory usage by utilizing the queuing system features of RabbitMQ and as routing services to communicate using the channel features available RabbitMQ.

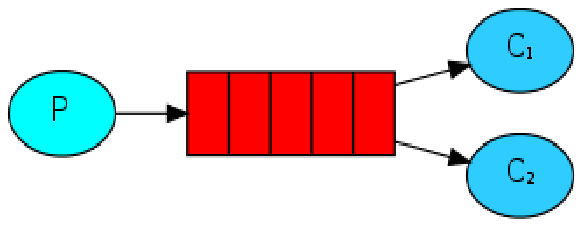

Figure 4. RabbitMQ work queue

P = Producer
C_n = Consumer (n).

Figure 4 shows the flow of the process in the RabbitMQ queuing system. P send a message to the channel, the first incoming process will be processed first on the channel by the C on the channel. The next process will be carried out by the consumer until the previous process has been completed.

H. *Software Testing*

The next stage is software testing. Testing is carried out to ensure the modules on the new system are running well and there are no errors. Testing is done by doing the same thing with the running system by doing data upload activities.

I. *Monitoring and Evaluation*

The next and final stage is the monitoring and evaluation stage. After testing the software on the new architecture, monitoring is carried out on the new architecture, for monitoring the performance of the new system.

IV. RESULTS AND DISCUSSION

A. *Arsitektur New System*

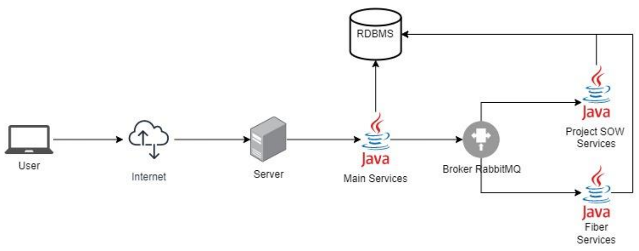

Figure 5. Design architecture new system

Figure 5. is a new design with a microservices architecture. In this microservices architecture design, the migration is done at the backend level of Java Spring Boot. The implementation of database as a service is not carried out because the problem occurs at the backend level of Java Spring Boot because the heap memory is based on server monitoring.

Services are divided into 3 (Main Services, Project Services, and Task Services). Communication between services is done using RabbitMQ. When the user uploads project or task data, it will first go through the server and then to the main services. From the main services, the process will be forwarded by sending a message to RabbitMQ. After the message is sent by Main Services, project services, or task services as consumers, will run the process.

With this architectural design, when there is a problem or excessive memory usage in a project or task module, it will not interfere with the main services or applications because they are in separate services. so that the application can run normally and only project or task modules cannot run.

B. Performance Upload Result

Table 4. Detail data upload project architecture microservices

No.	**Name**	**Description**
1.	Columns	87
2.	Records	2875
3.	Start Time	14:44:19
4.	End Time	14:45:54
5.	Total Time	93 seconds
6.	Record / s	1 record / 0.03 second

Table 4. is the result of the upload process to the module project after moving to the microservices architecture design. The data used is the same data as the system on the monolithic architecture, as many as 87 columns, because there is a time lag between the uploads of the data module project and the upload of the data module project on the microservices architecture. There is an increase in data by 384 records, so the total data that will be uploaded during the microservices architecture is 2875 records. Although there is an increase in data when uploading on the microservices architectural design, the performance shows better results with a total time required of 93 seconds, or 1 record for 0.03 seconds.

C. Memory Consumption

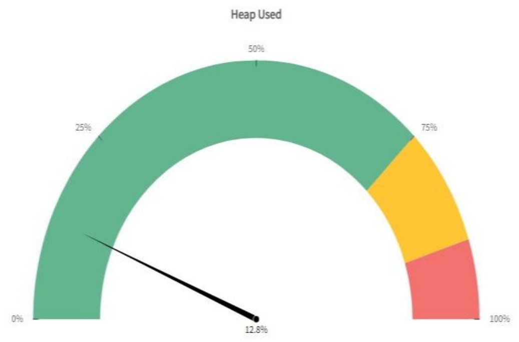

Figure 6. Memory consumption architecture microservices

When software testing is carried out on services that have been separated from main services, monitoring is carried out on servers with a microservices architecture. Based on this monitoring, it was observed that memory consumption was at 12.8 percent, and in the last 30 days, after the migration to the microservices architectural design, it was monitored to be safe and stable.

V. Conclusion and Future Scope

Based on the results of the research, it can be concluded that the migration process from monolithic architecture to microservices architecture based on the stages of the Strangler Fig Application method can be carried out for the system migration process from monolithic to microservices. And by solving problems that exist in applications with a monolithic architectural design, an IT consultant can overcome the decline in application performance as data increases and the development of the application causes the page to be unable to access when the load is high. From the upload time, which previously took 10758 seconds with 2491 records, became faster with a total time of 93 seconds with the addition of 384 records for a total of 2875 records and more optimal memory usage.
In this research, the tools used are limited based on the problems that exist in the application with the monolithic architectural design. Suggestions for further research can be migrated with more complete tools such as the application of API gateway, services discovery, database as a service to get maximum performance and results.

References

[1] Chris Richardson, Microservices Patterns With Examples in Java. Shelter Island, **NY**: Manning Publications Co., **2019.**

[2] Jakob Nielsen, Usability Engineering. California: Elsevier, **1993**. doi: 10.1016/C2009-0-21512-1.

[3] Sam Newman, Monolith to Microservices Evolutionary Patterns to Transform Your Monolith, Second Edition. **Gravenstein Highway North**: O'Reilly Media, Inc., **2020.**

[4] Mohammad Harry Khomas Saputra and Luthfi Muhammad Nabil, "PENERAPAN ARSITEKTUR MICROSERVICE PADA SISTEM TATA KELOLA MATAKULIAH PROYEK POLITEKNIK POS INDONESIA," Teknik Informatika, vol. **13**, no. **3**, **pp. 22–28**, Aug. 2021, Accessed: Aug. 1**, 2022.** [Online]. Available: https://ejurnal.poltekpos.ac.id/index.php/informatika/article/view/1667

[5] R. Mufrizal and D. Indarti, "Refactoring Arsitektur Microservice Pada Aplikasi Absensi PT. Graha Usaha Teknik," Jurnal Nasional Teknologi dan Sistem Informasi, vol. **5**, no. **1**, **pp. 57–68**, Apr. **2019**, doi: 10.25077/TEKNOSI.v5i1.2019.57-68.

[6] J. A. Suthendra and M. A. I. Pakereng, "Implementation of Microservices Architecture on E-Commerce Web Service," ComTech: Computer, Mathematics and Engineering Applications, vol. **11**, no. **2**, **pp. 89–95**, Dec. **2020**, doi: 10.21512/comtech.v11i2.6453.

[7] N. Torvekar and P. S. Game, "Microservices and Its Applications An Overview," International Journal of Computer Sciences and Engineering, vol. **7**, no. **4**, **pp. 803–809**, Apr. **2019**, doi: 10.26438/ijcse/v7i4.803809.

[8] S. S. Paradkar, "eGovernment Integration Framework for Fragmented Systems," International Journal of Computer Sciences and Engineering, vol. **9**, no. **1**, **pp. 51–55**, Jan. **2021**, doi: 10.26438/ijcse/v9i1.5155.

AUTHORS PROFILE

Mr. Lutfi Ardiansyah Completed his Bachelor of Information Systems from Gunadarma University, Indonesia in 2017. Currently working at Information Technologi Company as Software Engineer. His focusing on development product and starter pack back end java framework..

Mrs. Yuli Karyanti Currently Working as lecturer at the Faculty of Information Technology at Gunadarma University and as the person in charge of the UPT Gunadarma University.