MICROSERVICES FOR ASSET TRACKING BASED ON INDOOR POSITIONING SYSTEM

Dondi Sasmita[1] and Gede Putra Kusuma[2]

[1,2]*Computer Science Department, BINUS Graduate Program – Master of Computer Science, Bina Nusantara University, Jakarta, Indonesia. 11480*

E-mail address: dondi.sasmita@binus.ac.id, inegara@binus.edu

Received ## Mon. 20##, Revised ## Mon. 20##, Accepted ## Mon. 20##, Published ## Mon. 20##

Abstract: Indoor positioning system (IPS) is widely used for different use cases, but most of them are asset tracking and indoor navigation. Asset tracking for instance, might help industry have more efficient such as warehouse, stock recording, guest tracker and many more. Implementation of asset tracking need to have the IPS such as trilateration and fingerprinting. To have accurate location, it is not just the precise, but the data which will be consumed need robust services to process all those data in almost real time. Bluetooth low energy (BLE) is used to send the received signal strength indicator (RSSI) to the microservices based server. To support this, microservices architecture (MSA) is designed with Service-Oriented Modeling and Architecture (SOMA) framework to translates business goal into necessary services. We are implementing and comparing both MSA implementation strategies, which are orchestration and choreography strategies on the cloud computing with Kubernetes platform. These strategies compared to find the most resource efficient with biggest number of served requests. The bigger the served request number means more assets to be tracked in real time. Less resource usage could also mean the computationally is inexpensive. The study is finding that choreography strategy in MSA is better for IPS since the number of served requests are five times bigger with similar resources usage.

Keywords: Indoor Positioning System, Bluetooth Low Energy, Asset Tracking, Microservices Architecture, SOMA Framewor

1. INTRODUCTION

Global Positioning System (GPS) is being used everywhere for daily tasks, e.g., finding address, tracking package and others. GPS is powerful since it is using satellite technology, but GPS cannot be used for specific scopes such as indoor navigation or tracking assets inside a building or underground. These scenarios can be used in many industries such as hospitality, chain supply, museum and many more. In chain supply for instance, organization can track the location of their assets inside warehouse in real time. Another example in high security building, organization could track their guest's location. For the indoor navigation, it can be used inside a museum, mall, and university to get the location of the arts, shop or classroom. Also, in museum, IPS can be used to trigger the mobile application to show the explanation or descriptions of the arts.

The technology to get user specific location inside a building or room is called indoor positioning system (IPS) and it could be implemented using different technologies such as wireless fidelity (WiFi) [1], [2], [3], bluetooth low energy (BLE) [4], [5], [6], [7], [8], [9] radio frequency identification (RFID), etc.

BLE is a wireless technology that running in 2.4GHz frequency just like wireless fidelity (WiFi), but BLE's size is smaller, the price is cheaper, and it consume lower energy makes BLE should perform better in a long run and scale up efficiently in terms of cost. Mostly BLE powered by batteries, but some device like BLE Gateway will need more power than BLE Beacon so BLE Gateway mostly use electricity instead.

There are several techniques that quite common being used on indoor positioning system (IPS) like trilateration, fingerprinting, and hybrid. Trilateration is a technique to estimate distance by using 3 devices, in this experiment the device is BLE gateway. While fingerprinting is estimating distance by create a radio map database to collect data from several points and create a model using those data.

There are several methods to estimate asset's location like weighted k-nearest neighbor (WK-NN). Machine learning and deep learning are also quite common to be used to improve the IPS accuracy. With the data from Fingerprinting as a radio map database, those data will be

used to create a model based on neural network like artificial neural network (ANN) and others.

To get data in real time, the IPS need to be supported by great services behind it. There are few architectures to build a software such as monolith and microservices. Monolith's modules are dependent to each other while modules in microservices are independent. Since the modules are independent, services in microservices architecture (MSA) could be written in different programming languages and use messaging protocols to communicate like Protocol buffers (Protobuf) [10], and JavaScript Object Notation (JSON). Also, for the IoT itself can use message queuing telemetry transport (MQTT) [11], [12] to send telemetry information to the server.

For the IPS is accessible anytime and anywhere, its firewall needs to be opened to public. One of the easiest and most common is to host the software in public cloud instead of on premise. There is several global cloud hosting with reputable names such as Google Cloud Platform (GCP), Amazon Web Services (AWS), Alibaba Cloud, Microsoft Azure, Digital Ocean and many more.

In this experiment, BLE will be used as beacon and gateways, since it has several advantages like its size, price, and efficiency. The data from BLE will be sent to microservices based server. Researchers will design a microservices architecture with Service-Oriented Modeling and Architecture (SOMA) framework [13] that suit the asset tracking on IPS. The design will be using both MSA's implementation strategies, which are orchestration and choreography strategies. This MSA then will be deployed to cloud computing environment which have Kubernetes service. Finally, the comparison of served requests and resources usage between those strategies will be done to find the better approach for IPS.

2. Related Works

On this chapter, there are two previous works that will be discussed separately, which are indoor positioning methods and microservices architecture.

A. Indoor Positioning Methods

Implementation of IPS is dominated by Bluetooth and WiFi technologies, especially when Bluetooth Low Energy (BLE) is introduced because of its power consumption. BLE has a feature called advertisement packets, which being used for reporting the Received Signal Strength Indicator (RSSI).

In 2020, Assayag et al [4] did research and proposed a solution based on Bluetooth called PoDME (Positioning using Dynamic Model Estimation) where the model parameters will be calculated dynamically based on the RSSI value from best anchor nodes which received signal from users' smartphone.

The next year, Suseenthiran et al [5] also did research regarding BLE. They were using the RSSI and calculated the information with Trilateration and send it via LoRa (Long Range) technology. Their research can be continued in the future for IoT (Internet of Things) purpose, since LoRa can send signal in the long range.

Other than Trilateration, Fingerprinting is also becoming more popular. This method can be used for BLE and WiFi technologies as well [2], [3], [7] Riady & Kusuma used Fingerprinting and pedestrian dead reckoning (PDR) combined with ANN.

Neural network is also becoming one of the many methods that being used during research. Sulaiman et al were using generalized regression neural networks (GRNN). Combined with fingerprinting technique, their research divides the phase into two, offline and online. The offline phase is to collecting data from radio map fingerprinting and create neural network model.

Cha & Lim also proposed a new framework based on neural network called hierarchical auxiliary deep neural network (HADNN). The idea is to simplify the calculation of building, floor, and users' locations at once. Compared with the same dataset from TUT2017 and TUT2018, HADNN gave a better result and improved the floor's accuracy to 94.58%.

When trilateration or fingerprinting only is not enough, there are research that combined those two to get a better result. Lie & Kusuma is one of them. In 2021, proposed an algorithm called *coarse-to-fine* [14]. The MPE is 0,874 meter and the computation to process it is between 0.4 – 0,7 millisecond.

Mehrabian & Ravanmehr, in 2023, also used hybrid approach which combined PDR and fingerprinting [15]. They used a novel filter called weight-based optimization (WBO) to optimize the RSSI value. Then they optimized the path loss parameter with particle swarm optimization (PSO) to convert RSSI to a distance value. The MAE value is 0.68 meter from a room with size 4.25 x 10.7 meter^2.

From the IPS works that have been done in the last couple of years, there are trends to use BLE over other technologies because its performance getting better each year. Also, BLE is one of the cheapest options to mass produce. Hybrid is also becoming a way to find a better result by combining the advantage of each methodology.

B. Microservices Architecture

"A monolith is a software application whose modules cannot be executed independently" [16]. One of the reason organizations moved from monolith because of the flexibility. When the system is scaling up, so does the cost. Because not every service needs to be added, but with monolith architecture, the whole system should be scaled up.

"A microservices is a cohesive, independent process interacting via messages" [16]. While "A microservices architecture (MSA) is a distributed application where all its modules are microservices".

In 2004, Ali Arsanjani introduced techniques to identify, specify, and realize services a service-oriented architecture (SOA), their flows and composition, as well as the enterprise-scale components needed to realize and ensure the quality of services required by SOA [17]. Then in 2004, Arsanjani et al proposed a method for developing service-oriented solutions called service-oriented modeling and architecture (SOMA) [13]. SOMA is used for different size of scopes in multiple industries worldwide. The idea of SOMA is to standardize on how to analyze, design, implement and deploy SOA effectively.

In 2021, Wang et al surveyed and interviewed professionals with different backgrounds like software engineers to Devops from different position as well [18]. This survey was conducted in 2 ways, face-to-face and online, focusing on 3 main concept, architecture, infrastructure, and code management. 81% correspondent says that business process is the main reason when defining MSA granularity from services to be developed. While 43% refer to data access and 24% from team's structure.

Still in the same year, de Toledo et al also interviewed 25 practitioners related to 16 architectural technical debt (ATD). There's a potential issue when MSA developed with shared databases if not well designed. This issue could be solved with database per service and sagas pattern [19] to maintain data consistency for each service.

Other than data consistency, there are distribution, portability, availability, and robustness which are few key characteristics in MSA [20]. These can be implemented using several ways, like orchestration with Docker. This will isolate problems in each service without affecting other services.

Docker is one of many tools to contain the microservices. The purpose is to make developer life's easier by automate the scale-up and scale-down processes. Dragoni et al did research to show how powerful containerization and orchestration microservices with Kubernetes [20]. With the tool, developer can focus on the services performances.

In MSA, performance can be affected by its design. Shadija et al, in 2017, did research that shows services' granularity could affect the performance if its design is too deep [21]. The granularity could be affected by the size of application, business processes, number of software engineers, database design and the re-use of the services. Infrastructure also affects the performance, like the latency and service's location in container.

When design the MSA, other than performance, we need to look at the business model. If the product will be a software as a service (SaaS), then the approach might differ from internal used. In 2020, Song et al research a SaaS that can modified [22]. Normally, customer can only customize small part of the SaaS, but in this research, they proposed method called deep-customization. The idea is to group the granularity into 4, which are class/function, components, services, and languages.

Portability and maintaining the services are also the MSAs advantages. But with the services that growing and getting bigger, sometimes it would be tricky to find the root cause of issues, therefor Brandón et al did research a topic related to RCA in MSA. They were using graph to help visualize the RCA [23]. From their finding, graph solution is better than machine learning approach with difference around 19.41%. They used Kafka and Spark to detect systems anomaly and saved it. The solution is trained by using the data's patterns and shows the RCA in graph. Their solution allow user to set priority on service, component, or database level.

SOMA is still a framework that most being used by many industries when they develop MSA. People tried to improve the details by add more tools, methods and techniques to make the MSA robust, maintainable and scalable.

3. Theory and Method

A. Bluetooth Low Energy

Bluetooth Low Energy (BLE) is a wireless technology that run on unlicensed frequency 2.4 GHz. Price of BLE is cheaper compared to other technologies and supported by most smartphones makes BLE quite common to be used [24]. BLE power consumption also quite efficient and its size relatively small makes this technology is easier to be deployed [5]. BLE also has higher sample rates, small network latency and its accuracy is better than WiFi [6].

For several scenarios, BLE can be grouped into two, as a beacon and a receiver. BLE beacon is a Bluetooth based device that can Bluetooth signal and powered by batteries [8]. While receiver is a Bluetooth based device which can received signal from transmitter or beacon that will be processed or sent to server.

BLE has a feature that report the RSSI where this data will be used to calculate the signal's strength between beacon and receiver. The power of the signal is dependent on distance and broadcast power value. From RSSI data, the distance between beacon and receiver can be calculated using several methods like Lateration and Fingerprinting.

B. Fingerprinting

Fingerprint is divided into two phases, offline and online. During offline phase, RSS data will be collected from several reference points as shown in Figure 1. With this information, radio map database will be created. For each row of the database contains two dimensions of x and y. The offline phase's goal is to form a model [24] Later,

on online phase, this data will be used to estimate the asset location using WK-NN.

To get the weight for this method, firstly we should calculate with (1).

$$W_n = \sum_{i=1}^{n} \frac{1}{d_i} \quad (1)$$

where W is weight's value and d is the result of calculation between asset location's RSS and RSS from radio map, which will be calculated using Euclidian distance as shown in (2).

$$d_i = \sqrt[2]{\sum_{j=1}^{n} \left(p_{ij} - \hat{p}_{ij}\right)^2} \quad (2)$$

where p is the RSSI value that saved in radio map and pˆis the measured RSSI. Once we have all the distances and weights, we should normalize the weights using (3).

$$NW_i = \frac{W_i}{\sum_{j=1}^{n} W_j} \quad (3)$$

By calculate each normalized weight, asset's location's estimation can be calculated using (4).

$$(x, y) = \sqrt{\sum_{j=1}^{n} \left(NW_j \,.\, (x_j, y_j)\right)} \quad (4)$$

Where (x,y) is estimated position while (x_j,y_j) is reference point.

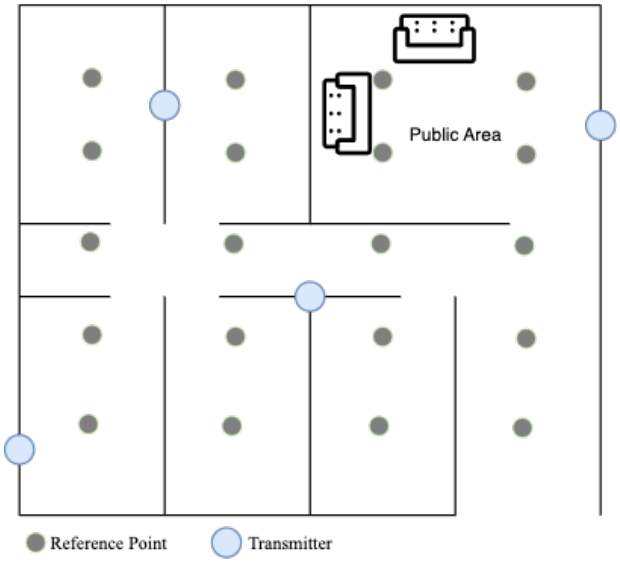

Figure 1. Fingerprinting Method

C. Message Queuing Telemetry Transport

This protocol is designed specifically for operation on low-cost and low-power sensor / actuator devices. MQTT runs over TCP with small data packet so the power consumptions is minimized [12].

MQTT uses publish/subscribe pattern. The publisher will send the message, while the subscriber will receive it. In order to filter messages, MQTT uses topic to group the messages, so the subscribers only get messages from topic which they subscribed as shown on Figure 2.

MQTT also provide a quality of service (QoS) where guarantees the reliability of messages delivery. There are three levels, level 0 messages are sent only once. Level 1 messages are sent at least once, so if the subscribers are not ready, the publishers could re-send the messages again. Lastly, Level 2 messages are needs to be received. Important or chained data might use level 2 QoS.

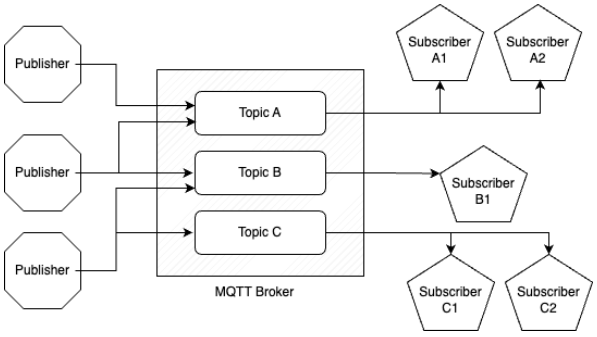

Figure 2. Publish subscribe pattern

D. Microservices

A microservice is a cohesive, independent process interacting via messages. And a microservice architecture is a distributed application where all its modules are microservices [16].

A few MSA characteristics that makes MSA more favorable than monolith architecture are flexibility, modularity, size, and independency. By flexibility means the system should be aligned with business dynamics and supported the modification that needed by the organization. The system's components need to be isolated where each component contributed to the whole function. In MSA, the service's size should be small, if it is too big, the service needs to be break down into 2 or more services so each service can focus on one thing only. Lastly, services in MSA should be operated independently from other services and can only communicate through published interfaces.

MSA can be implemented using 2 strategies which are orchestration and choreography as shown in Figure 3. Orchestration needs a conductor, centralized service which will send request to other services until response is received. While choreography assume that there is no

centralized service, so the strategy will use publisher subscriber mechanism to collaborate between services.

In Figure 4 shows deployment era from traditional to container. In traditional deployment, the application is running on top of the operating system which will be difficult if there are applications that needed run on different stacks. This issue was solved in virtualization era where the application will be running inside virtual machine. But this deployment also have an issue where there are multiple applications that runs on different versions of libraries. For instance, application A needs to be run on Java 8, while the other is using Java 11. These Java might need different libraries, and if the system was updated, the application might break. That's where the containerization deployment tries to address this issue, where each application will have its own environment. With this scope, developers will be less worry about breaking applications when there are updates applied.

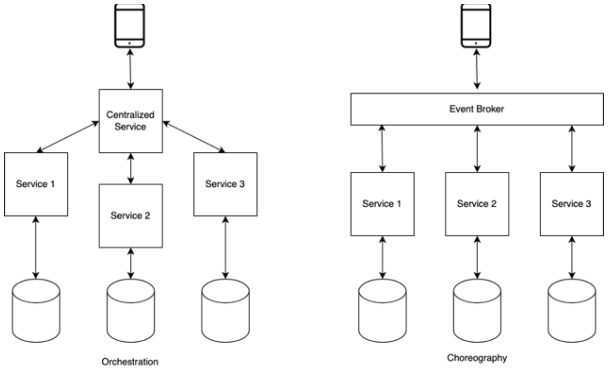

Figure 3. Orchestration and Choreography

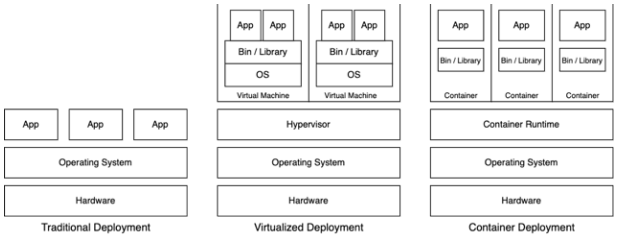

Figure 4. Comparison between deployments

E. Google Protocol Buffers and Remote Procedure Call

Protocol buffers (Protobuf) is developed by Google that introduced a mechanism to encode structured data in an efficient and extensible format [10]. Protobuf is human and machine friendly because it allows developers to create a Proto file which later will be compiled into native language. Protobuf is language and platform independent which in line with microservices behavior. It supports Go, Python, Kotlin and other programming languages.

Google Remote Procedure Call (gRPC) is an open-source remote procedure call framework that utilize Protobuf as interface description language and uses HTTP/2 protocol [25]. gRPC can use a single HTTP/2 connection for bi-directional communication between client and server. This key feature allows to stream messages in both directions.

Figure 5 shows the implementation of gRPC with Protobuf. gRPC server can be written in Go and will be called by gRPC Stub written in Python or Kotlin. These gRPC use Protobuf to send requests and receive responses.

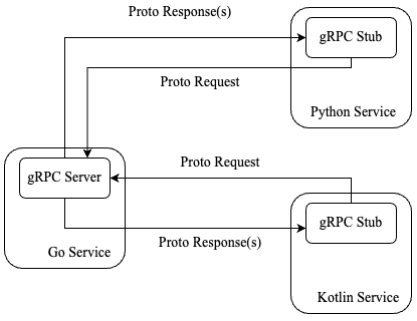

Figure 5. Google Remote Procedure Call

F. Cloud Computing

Cloud computing is a model that allows networks on demand for configurable resources like network, database, server, application, and services that can be provided and deleted instantly using user interface [7].

There are five elements in cloud computing that is on-demand self-service, broad network access, resource pooling, rapid elasticity, and measured service.

On-demand self-service gives flexibility to customers to instantiate new resources (CPU, Storage, etc) automatically without any interactions with other human that provided the services. With broad network access, the resources can be accessed from public devices such as laptop, personal computer (PC) and smartphone. The goal of resource pooling is to give economic scale to customers by implementing multi-tenancy and model virtualization. Next element is rapid elasticity that make sure customers can add and remove resources without commitment and long-term contract. And to give customers a way to evaluate their resources, cloud computing hosting give information based on measured resources that customer subscribed to.

In terms of services, there are four group that communities aware of which are Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS) and Data Storage as a Service (DaaS) [26].

4. PROPOSED MICROSERVICES DESIGN FRAMEWORK

We are using Service-Oriented Modeling and Architecture (SOMA) framework to interpret indoor positioning's business use case into services. There are six phases in SOMA which are business and modeling

TABLE I. GOAL-SERVICE MODELING

Goal	KPIs	Metrics	Services
1. Attract and retain customers	Increase new and retain existing users		
1.1 Enable assets tracking service	Increase new and retain existing users with indoor positioning system		
1.1.1 Enable assets tracking service in office complex	Increase new and retain existing users with indoor positioning system for office complex	Number of indoor positioning system users in office complex	- Account registration - Account management
		Number of indoor positioning system tracked assets in office complex	- Office management - BLE management - Asset management - Asset's location's estimation - Asset's location histories - Algorithm management
	Improve indoor positioning system's quality	Number of issues and solved issues of indoor positioning system	- Store log information - Monitoring response time

transformation, identification, specification, realization, implementation, and deployment & monitoring.

The research is focusing on estimating a location in a room within a building. During identification, we used goal-service modeling (GSM) to transform business goals into services based on its key performance index (KPI) and the metrics that needs to be measured. The goal-service modeling can be seen from Table I.

Once we identified basic services through GSM, we then detailing founded services with a technique called domain decomposition. This technique is a top-down analysis of business model and business process modeling to identify services, components, and flows. The result is on Table II.

During domain decomposition we focused on account management and asset location only. In here we are detailing the process of account registration, where the flow added by email verification and notification. We did the same thing with the office & BLE registration and asset location. We added more flows into those services.

Once the services are fixed, we then group those functions into services and explore technical feasibility through early prototype designed and developed. These steps are part of specification and realization. The designed architecture can be seen on Figure 8.

TABLE II. DOMAIN DECOMPOSITION

1. Account Management	1. Account Management
1.1. Account Registration	1.1. Account Registration
1.2. Office Registration	1.1.1. Email Verification
1.3. BLE Registration	1.1.2. Notification
2. Assets Location	1.2. Office Registration
2.1. Location Estimation	1.2.1. Building Registration
2.2. Location Histories	1.2.2. Floor Registration
	1.3. BLE Registration
	1.3.1. BLE Beacon
	1.3.2. BLE Gateway
	2. Assets Location
	2.1. Location Estimation
	2.1.1. RSSI Data Checking
	2.1.2. Estimation Methods Checking
	2.1.3. Estimation Process
	2.2. Location Histories

5. PROPOSED METHOD

A. Indoor Posititoning

We are going to use an Espressif board named ESP32-S3 shown on Figure 6. It has BLE 5.0, and Wi-Fi modules

embedded. We will need to write a custom firmware so it will scan the BLE Beacon with UUID that predefined. This device will be our gateway to receive beacon RSSI, then assemble a payload with several other information then send to the server. We will put 6 devices for a room with size of 19 x 12 meter^2.

Fingerprinting method with WK-NN will be used to estimate the location of BLE beacon. With fingerprinting there will be two phases, online and offline. For offline phase, we will collect RSSI data based on 46 reference points and 45 testing points. The floor plan ca be seen on Figure 7. Blue color is the BLE gateway, green color is the reference point and the red one is the testing point. We will pick 45 random testing points from the plan. The number of K that will be used is 6.

Figure 6. Espressif ESP32-S3

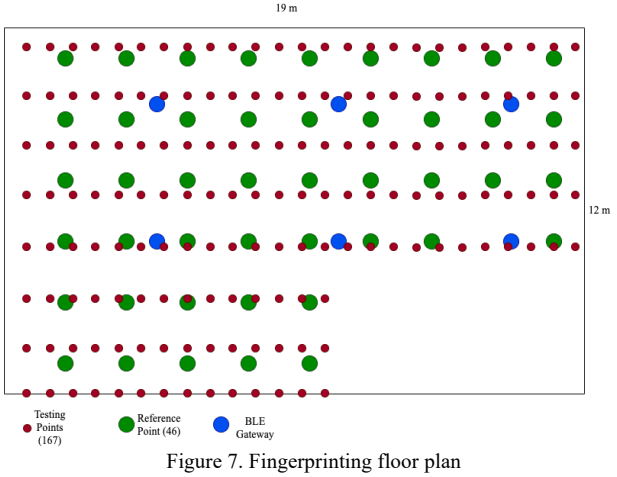

Figure 7. Fingerprinting floor plan

Once we got the radio map, we will estimate the location with WK-NN (4) and calculate the MPE with (5)

$$MPE = 1/n \sum_{i=1}^{n} Error_i \quad (5)$$

where error is a Euclidean distance calculated from real location of the beacon (x,y) with estimated location (x_p, y_p) shown in (6)

$$error = \sqrt{(x - x_p)^2 + (y - y_p)^2} \quad (6)$$

Together with the error, we will benchmark the estimation's process time, and Central Processing Unit (CPU) & Random Access Memory (RAM) usage as well.

B. *Microservices*

With SOMA, we have specified and decided the IPS functionalities. We translated those features and services into a chart shown on Figure 8. There will be seven services, which are account, authentication, log, location, subscriber, estimator, and API gateway services. To support those services, third party software will be installed as well. They are PostgreSQL as relational database management system, MongoDB as document data model, and EMQX as an MQTT broker.

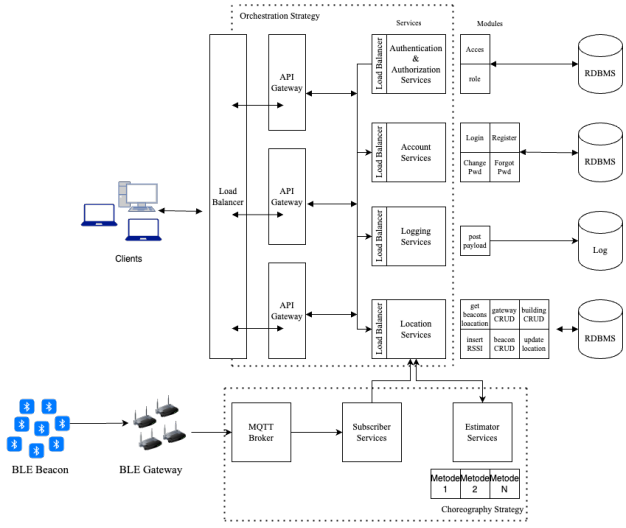

Figure 8. Proposed microservices architecture

Subscriber service is the first service that receiving RSSI information from BLE gateways. The information will be sent by BLE gateways use MQTT protocol through MQTT broker. The main program of subscriber service is to connected to MQTT broker, and subscribe to a certain topic. Every time a payload sent to a topic, subscriber service will receive it, and process the payload. Once the payload processed, subscriber will call location service using Google remote procedure call (gRPC) with protocol buffer.

Location service is designed to stored information related to location of the asset. Based on our business flow, we are storing several information, e.g. buildings, beacons, gateways, estimation algorithms, RSSI and the assets' locations itself.

Even though the estimated location stored in location service, the process of estimation itself happened in different service, which is estimator service. This service

job is only to estimate assets' locations based on methods or model that have been assigned to a building or a floor. There are possibilities that different buildings or floors might need different algorithm. This service will call location service's function periodically to check whether there are assets that needs to be estimated. If there is, estimator service will check the assigned algorithm, use it to estimate location, and return the coordinates of X and Y to location service.

For a security issue, all services are inaccessible from outside of private network, except API gateway service. API gateway service built as orchestrator, hence the name of the strategy. As an orchestrator, the flow inside microservices is transparent to the client. For instance, when client call an endpoint to retrieve list of buildings, API gateway communicate with several services, which are account, auth, log, and location.

The other services, which are account, auth and log services are supported services on this research. Those services are developed to do basic things such as users' registration, checking password, store log information into databases and other things that not directly affected on location service's performance.

To host these services, we are using cloud provider named Digital Ocean in Singapore region for this research's infrastructure. They have a managed Kubernetes service so we could focus on developing the services instead of maintaining the Kubernetes platform. We will use their load balancers as well to balance the traffic for both orchestration and choreography strategies.

Digital Ocean has dashboard that shows metrics related to their servers and Kubernetes platform. This will help us to measure our services' performance. During the test, the metrics that will be measured are CPU and RAM. Other than those, we will measure the number of served requests too.

C. Evaluation

The microservices performance's test scenarios are to compare between 2 strategies, orchestration and choreography. For orchestration, there will be two sub-scenarios where we will test the API gateway with Post method only and Get & Post methods together. So, there will be three scenarios in total. Each scenario will be tested with the same settings. The settings are to test with different number of nodes and pods replication. We will test with 1 node & 3 pods, 1 node & 5 pods, 2 nodes & 5 pods, and 2 nodes & 8 pods. In terms of number of connections, the test will use 100, 500, and 1,000 concurrent connections.

The specification of Kubernetes cluster's machine is 8 vCPU and 16 GB RAM per node. For load balancer, we will use 2 nodes so it can accommodate up to 20,000 connections, 20,000 request per second, and 500 SSL connections concurrently.

6. Result and Discussion

A. Data Acquisition

Once BLE gateways installed and the microservices are deployed on the cloud, we started with data collection. Using fingerprinting, we collected RSSI based on reference points (RP). There are 46 RP with 100 sample data for each RP. Total RSSI data is 4600 records. From these records, we calculate the average of RSSI for each RP, so by the end the radio map based on reference points is only 46 records. This radio map then uploaded on estimator service for fingerprinting online phase. To calculate the accuracy, we collected another radio map based on testing points (TP) where the coordinates are different from RP.

The radio map format for both RP and TP is the same. There are eight columns on it which represents beacon coordinates with x and y. The other 6 are RSSI from different BLE gateway. The format is shown in Table III.

TABLE III. FINGERPRINTING RADIO MAP

X	Y	G1	G2	G3	G4	G5	G6
200	100	-61	-63	-65	-73	-70	-76
200	300	-59	-69	-68	-73	-71	-76
200	500	-57	-55	-61	-67	-75	-73
…	…	…	…	…	…	…	…
…	…	…	…	…	…	…	…
1800	500	-68	-62	-54	-54	-40	-47
1800	700	-66	-66	-59	-54	-54	-31

With these radio map, we calculated accuracy of WK-NN method and its resource usage. Using (4) and (6), the MPE of the method is 6.32 meter. The estimation process took about 1.98 millisecond with CPU usage around 1.2% and 0.5% RAM.

B. Microservices Architecture

After data collected with fingerprinting method completed, we are testing the microservices' performances. There are three scenarios which are orchestration strategy for Get & Post methods, orchestration strategy for Post method, and choreography strategy.

We used a modern benchmark tool called Bombardier to load test the orchestration strategy. It works by calling an endpoint for interval time with certain number concurrent connections. We called the endpoints for 120 seconds with 100, 500, and 1,000 connections to find the number of served requests and its CPU & RAM usage.

TABLE IV. ORCHESTRATION STRATEGY PERFORMANCE (GET & POST)

No. of Connections	No. of Node	No. of Pod	Average CPU (%)	Peak CPU (%)	Average RAM (%)	Peak RAM (%)	Total Served Request	Served Request per Second
100	1	3	22.2	36.4	20.0	20.2	26,627	221
	1	5	33.4	55.2	21.1	21.2	35,702	297
	2	5	15.1	29.1	14.1	15.1	36,239	301
	2	8	23.9	46.9	13.9	17.3	55,905	465
500	1	3	36.1	49.0	20.2	20.4	37,310	310
	1	5	32.7	47.6	22.0	22.0	51,721	431
	2	5	17.1	35.1	14.4	15.5	44,829	373
	2	8	25.1	46.0	14.1	17.5	79,120	659
1000	1	3	27.1	33.9	21.4	21.8	31,926	266
	1	5	36.5	41.2	22.4	22.9	51,876	432
	2	5	16.5	26.0	14.5	15.4	47,287	394
	2	8	33.3	63.6	15.2	17.2	84,367	703

TABLE V. ORCHESTRATION STRATEGY RESULT (POST)

No. of Connections	No. of Node	No. of Pod	Average CPU (%)	Peak CPU (%)	Average RAM (%)	Peak RAM (%)	Total Served Request	Served Request per Second
100	1	3	31.2	38.1	20.5	21.0	23,638	196
	1	5	42.7	57.2	22.1	22.8	34,538	287
	2	5	23.0	35.3	14.7	14.9	31,216	260
	2	8	31.6	46.8	15.5	16.8	38,292	319
500	1	3	32.1	40.1	20.5	21.1	29,346	244
	1	5	42.4	59.9	22.2	23.1	27,143	226
	2	5	23.7	39.2	14.6	15.1	33,844	282
	2	8	32.1	57.3	15.5	16.9	40,784	339
1000	1	3	33.4	46.1	21.4	21.8	18,310	152
	1	5	44.5	63.0	22.3	23.0	23,652	197
	2	5	25.5	41.4	14.4	15.4	34,984	291
	2	8	34.3	65.6	15.3	17.1	43,580	363

TABLE VI. CHOREOGRAPHY STRATEGY RESULT

No. of Connections	No. of Node	No. of Pod	Average CPU (%)	Peak CPU (%)	Average RAM (%)	Peak RAM (%)	Total Served Request	Served Request per Second
100	1	3	21.3	25.9	20.9	21.0	54,835	456
	1	5	19.1	23.1	25.2	25.4	59,805	498
	2	5	14.7	28.3	16.8	22.8	63,570	529
	2	8	20.0	32.1	20.4	25.7	68,990	574
500	1	3	21.2	28.8	21.9	21.9	151,419	1,261
	1	5	21.6	49.6	25.5	25.7	202,211	1,685
	2	5	18.8	32.5	17.4	23.2	237,448	1,978
	2	8	23.8	37.3	20.4	25.9	233,798	1,948
1000	1	3	23.4	33.7	22.0	22.0	166,682	1,389
	1	5	44.8	62.8	26.1	26.3	313,122	2,609
	2	5	19.1	33.5	17.8	23.5	338,781	2,823
	2	8	26.5	38.6	21.2	25.9	457,970	3,816

While for choreography strategy, we wrote a custom Python script to imitate real life scenario where BLE gateway push the RSSI payload through MQTT broker for each interval time. In this scenario, we sent six payloads per second for each connection for 120 seconds. We used the same number of connections as the orchestration, which are 100, 500, and 1,000 concurrent connections.

We are using four settings to test the microservices' performance. 1 node with 3 replications, 1 node with 5 replications, 2 nodes with 5 pod replications, and 2 nodes with 8 pod replications. The result are shown on Table IV, Table V, and Table VI. The numbers are available from cloud dashboard shows on Figure 9.

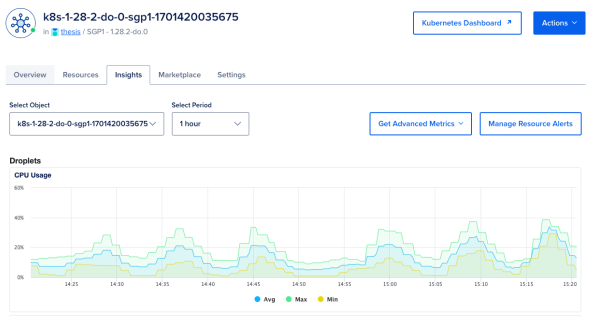

Figure 9. CPU usage on Kubernetes platform

Table IV shows the result for orchestration strategy for Get & Post methods. This scenario shows the biggest served request per second (RPS) happened when there were 1,000 concurrent connections called the API. The RPS is 703 with the average CPU usage is 33.3% and RAM with 15.2%.

Second scenario on Table V shows that calling Post method shows smaller RPS. The RPS is 363 with average CPU 34.3% and RAM 15.3%. Compared to first scenario, this scenario used bigger CPU power but not the RAM. It seems the Post method mainly used CPU, since the RAM of 100, 500, and 1,000 connections are quite the same.

Last scenario, choreography strategy on Table VI gave us the best result with 3,816 RPS. With 1,000 connections, this scenario served 457,970 requests for 120 seconds with average CPU 26.5% and RAM 21.2%.

7. Conclusion and Future Work

In this paper, we have implemented indoor positioning system using BLE and microservices on the cloud environment. The microservices were designed with SOMA framework and implemented in choreography and orchestration strategies. This MSA then deployed on a Kubernetes platform. After services ready, both strategies were called with multiple scenarios and choreography strategy shows a better result with 457,970 served requests or five times bigger than the other scenarios. The CPU usage is smaller compared to orchestration strategy's scenarios, but RAM usage is 50% more.

The location estimation method used in this research is a simple WK-NN. With the neural network thrived, the IPS could be improved by implementing other location estimation methods such as Graph Neural Network (GNN) to have more accurate estimated location in the future.

References

[1] O. I. Mustafa, H. L. Joey, N. A. AlSalam, and I. Z. Chaloob, "Accurate indoor positioning system based on modify nearest point technique," *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 12, no. 2, p. 1593, Apr. 2022, doi: 10.11591/ijece.v12i2.pp1593-1601.

[2] B. Sulaiman, E. Natsheh, and S. Tarapiah, "Towards a better indoor positioning system: A location estimation process using artificial neural networks based on a semi-interpolated database," *Pervasive Mob Comput*, vol. 81, p. 101548, Apr. 2022, doi: 10.1016/j.pmcj.2022.101548.

[3] J. Cha and E. Lim, "A hierarchical auxiliary deep neural network architecture for large-scale indoor localization based on Wi-Fi fingerprinting," *Appl Soft Comput*, vol. 120, p. 108624, May 2022, doi: 10.1016/j.asoc.2022.108624.

[4] Y. Assayag, H. Oliveira, E. Souto, R. Barreto, and R. Pazzi, "Indoor Positioning System Using Dynamic Model Estimation," *Sensors*, vol. 20, no. 24, p. 7003, Dec. 2020, doi: 10.3390/s20247003.

[5] K. Suseenthiran *et al.*, "Indoor positioning utilizing bluetooth low energy (BLE) RSSI on LoRa system," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 23, no. 2, p. 927, Aug. 2021, doi: 10.11591/ijeecs.v23.i2.pp927-937.

[6] G. P. Kusuma and M. M. K. Lie, "A review of indoor positioning system techniques using bluetooth low energy," *ICIC Express Letters*, vol. 13, no. 12, pp. 1139–1147, 2019, doi: 10.24507/icicel.13.12.1139.

[7] A. Riady and G. P. Kusuma, "Indoor positioning system using hybrid method of fingerprinting and pedestrian dead reckoning," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 7101–7110, 2022, doi: https://doi.org/10.1016/j.jksuci.2021.09.005.

[8] G. Pau, F. Arena, M. Collotta, and X. Kong, "A practical approach based on Bluetooth Low Energy and Neural Networks for indoor localization and targeted devices' identification by smartphones," *Entertain Comput*, vol. 43, p. 100512, Aug. 2022, doi: 10.1016/j.entcom.2022.100512.

[9] F. J. Aranda, F. Parralejo, F. J. Álvarez, and J. A. Paredes, "Performance analysis of fingerprinting indoor positioning methods with BLE," *Expert Syst Appl*, vol. 202, p. 117095, Sep. 2022, doi: 10.1016/j.eswa.2022.117095.

[10] S. Popic, D. Pezer, B. Mrazovac, and N. Teslic, "Performance evaluation of using Protocol Buffers in the Internet of Things communication," in *2016 International Conference on Smart Systems and Technologies (SST)*, IEEE, Oct. 2016, pp. 261–265. doi: 10.1109/SST.2016.7765670.

[11] U. Hunkeler, H. L. Truong, and A. Stanford-Clark, "MQTT-S — A publish/subscribe protocol for Wireless Sensor Networks," in *2008 3rd International Conference on Communication Systems Software and Middleware and Workshops (COMSWARE '08)*, IEEE, Jan. 2008, pp. 791–798. doi: 10.1109/COMSWA.2008.4554519.

[12] R. A. Atmoko, R. Riantini, and M. K. Hasin, "IoT real time data acquisition using MQTT protocol," *J Phys Conf Ser*, vol. 853, p. 012003, May 2017, doi: 10.1088/1742-6596/853/1/012003.

[13] A. Arsanjani, S. Ghosh, A. Allam, T. Abdollah, S. Ganapathy, and K. Holley, "SOMA: A method for developing service-oriented solutions," *IBM Systems Journal*, vol. 47, no. 3, pp. 377–396, 2008, doi: 10.1147/sj.473.0377.

[14] M. M. K. Lie and G. P. Kusuma, "A fingerprint-based coarse-to-fine algorithm for indoor positioning system using Bluetooth Low Energy," *Neural Comput Appl*, vol. 33, no. 7, pp. 2735–2751, Apr. 2021, doi: 10.1007/s00521-020-05159-0.

[15] H. Mehrabian and R. Ravanmehr, "Sensor fusion for indoor positioning system through improved RSSI and PDR methods," *Future Generation Computer Systems*, vol. 138, pp. 254–269, Jan. 2023, doi: 10.1016/j.future.2022.09.003.

[16] N. Dragoni *et al.*, "Microservices: yesterday, today, and tomorrow," Jun. 2016, [Online]. Available: http://arxiv.org/abs/1606.04036

[17] A. Arsanjani, "Service-Oriented Modeling and Architecture," *IBM developer works*, Mar. 2004.

[18] Y. Wang, H. Kadiyala, and J. Rubin, "Promises and challenges of microservices: an exploratory study," *Empir Softw Eng*, vol. 26, no. 4, p. 63, Jul. 2021, doi: 10.1007/s10664-020-09910-y.

[19] H. Garcia-Molina and K. Salem, "Sagas," *ACM SIGMOD Record*, vol. 16, no. 3, pp. 249–259, Dec. 1987, doi: 10.1145/38714.38742.

[20] N. Dragoni, I. Lanese, S. T. Larsen, M. Mazzara, R. Mustafin, and L. Safina, "Microservices: How To Make Your Application Scale," Feb. 2017, [Online]. Available: http://arxiv.org/abs/1702.07149

[21] D. Shadija, M. Rezai, and R. Hill, "Microservices: Granularity vs. Performance," Sep. 2017, [Online]. Available: http://arxiv.org/abs/1709.09242

[22] H. Song, F. Chauvel, and P. H. Nguyen, "Using Microservices to Customize Multi-tenant Software-as-a-Service," in *Microservices*, Cham: Springer International Publishing, 2020, pp. 299–331. doi: 10.1007/978-3-030-31646-4_12.

[23] Á. Brandón, M. Solé, A. Huélamo, D. Solans, M. S. Pérez, and V. Muntés-Mulero, "Graph-based root cause analysis for service-oriented and microservice architectures," *Journal of Systems and Software*, vol. 159, p. 110432, Jan. 2020, doi: 10.1016/j.jss.2019.110432.

[24] R. P. Ghozali and G. Putra, "Indoor Positioning System using Regression-based Fingerprint Method," *International Journal of Advanced Computer Science and Applications*, vol. 10, no. 8, 2019, doi: 10.14569/IJACSA.2019.0100829.

[25] C. Nimpattanavong, I. Khan, T. Van Nguyen, R. Thawonmas, W. Choensawat, and K. Sookhanaphibarn, "Improving Data Transfer Efficiency for AIs in the DareFightingICE using gRPC," Mar. 2023.

[26] T. Dillon, C. Wu, and E. Chang, "Cloud Computing: Issues and Challenges," in *2010 24th IEEE International Conference on Advanced Information Networking and Applications*, IEEE, 2010, pp. 27–33. doi: 10.1109/AINA.2010.187.

Dondi Sasmita received B.Sc. degree in Computer Science major from Bina Nusantara University in 2009 and received M.Sc. degree in Master of Computer Science in 2024. System architecture, cloud computing and machine learning are his research interests.

Gede Putra Kusuma received PhD degree in Electrical and Electronic Engineering from Nanyang Technological University (NTU), Singapore, in 2013. He is currently working as a Lecturer and Head of Department of Master of Computer Science, Bina Nusantara University, Indonesia. Before joining Bina Nusantara University, he was working as a Research Scientist in I2R – A*STAR, Singapore. His research interests include computer vision, deep learning, face recognition, appearance-based object recognition, gamification of learning, and indoor positioning system.