High-Performance and Resource-Efficient IoT-based Sleep Monitoring System

Nico Surantha
Computer Science Department, BINUS Graduate Program-Master of Computer Science
Jakarta, Indonesia, 11480
nico.surantha@binus.ac.id

Oei Kurniawan Utomo
Computer Science Department, BINUS Graduate Program-Master of Computer Science
Jakarta, Indonesia, 11480
oei.utomo@binus.ac.id

Sani Muhamad Isa
Computer Science Department, BINUS Graduate Program-Master of Computer Science
Jakarta, Indonesia, 11480
sani m.isa @binus.ac.id

***Abstract*— Recently, the number of people who have trouble sleeping is on the rise. It results in an increase in the number of car accidents, lower productivity and a higher risk of chronic disease. Advancing wearable and contactless health sensors offers the possibility of daily sleep quality monitoring that can be done on a regular basis by patients in their homes. The goal of this research is to propose the design of an Internet-of-Things (IoT) sleep quality monitoring system that can be integrated with a wearable or contactless sensor. The proposed platform architecture is based on micro-services and is event-driven. The simulation results show that the implementation of the IoT sleep quality monitoring system based on the proposed design decreases the response time by 55.85 per cent while increasing the throughput by 34.76 per cent compared to the traditional approach. On the other hand, the rate of increase in memory usage per instance replication for proposed method is lower than the traditional method.**

Keywords—Internet-of-Things, Microservices, Event-Driven, Sleep Quality, Health Monitoring

I. INTRODUCTION

Sleep is a condition where the human mind and body are at rest. Poor quality of sleep (sleep deficiency) will cause physical and psychological problems, from dizziness to work accidents [1]. Therefore, early detection of poor sleep quality needs to be done using a sleep quality monitoring system. A sleep quality monitoring system integrates various technologies, such as sensors, mobile technology and cloud computing with Internet of Things technology. IoT technology can maximize sleep quality monitoring systems in receiving data from sensors, processing the data, and displaying it on a dashboard. The sleep data should be accessible to users and health practitioners, who can later provide an analysis of the patient's health conditions in terms of sleep quality. Various types of architecture have been proposed as sleep monitoring systems [1]–[4]. In general, an IoT-based health monitoring system has three main parts: a sensor gateway to collect data, an IoT platform to store and process data from sensors on sleep quality, and a dashboard that can be accessed by patients/health practitioners.

In software architecture technology, microservices and event-driven are two different concepts which can be utilized as IoT backend services. The microservices architecture separates a whole system into several services that are independent of each other, depending on each function of each service. With low inter-service coupling, each component of the microservices can be developed and deployed separately to enhance the software development process, as well as increase the servers' resource usage efficiency [5]. On the other hand, event-driven architecture is an architecture where the flow of the system is built, determined, and triggered by events that occur within the system [6]. There are three main roles in event-driven architecture: as an event producer, an event consumer, and a message broker. The process in event-driven architecture occurs asynchronously, where the producer and the consumer are not connected directly. Therefore, implementing an event-driven architecture will also reduce coupling/dependency between services and maximize the capability of each service to serve its function.

In this study, an IoT platform for sleep quality monitoring is proposed. The main contributions of this paper are:

1) An IoT platform architecture for sleep quality monitoring system which is designed based on the microservices and event-driven architecture,
2) A sleep monitoring dashboard that can be accessed using a web browser, where each patient will have a personal account of their sleep quality data and health practitioners can see the sleep quality data of all patients.

By focusing of these contributions, this work aims to achieve good performance in throughput, response time, and memory allocation. The sleep apps provider will benefit from the proposed system because they can provide a reliable service while investing less on cloud computing services. The sleep stage classification was previously published in a conference [7].

The structure of this paper is organized as follows. Section II provides the proposed IoT platform and sleep stage classification algorithm. Section III presents the proposed sleep quality dashboard. The system's performance evaluation is presented in section IV. The last part, section V, ends this paper with a conclusion as well as future work.

II. THE PROPOSED SYSTEM ARCHITECTURE

In this paper, we propose a monitoring system using microservices and event-driven architecture that aims to achieve good performance in throughput, response time, and memory allocation. The proposed microservices and event-driven architecture of sleep quality monitoring system is presented on Fig. 1. There are three main parts, i.e. data acquisition, the IoT cloud server, and the monitoring application. Each block in the IoT platform is a microservices application that has its own source code which is separated from other applications and can be deployed independently. The monitoring app will be served by the web application gateway service from the IoT platform using single-page-applications technology [8]. This study will focus on the part of the IoT platform.

In the part of data acquisition, the system will use sensor simulation data which is streamed from the MIT-BIH polysomnographic data. There are five different functions in

978-1-7281-5207-3/20/$31.00 ©2020 IEEE

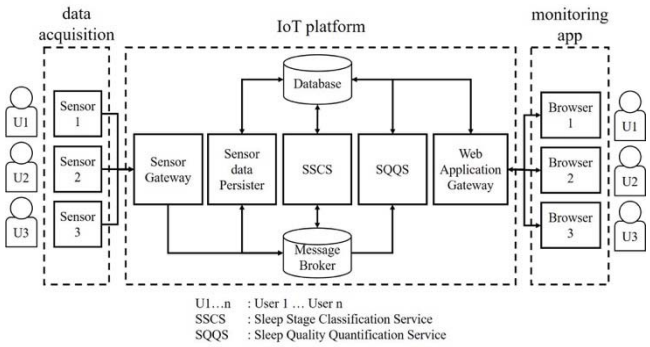

Fig. 1 The Proposed IoT Platform Architecture for Sleep Quality Monitoring System.

the proposed IoT platform for sleep quality monitoring system: receiving data from the sensor, storing/persisting the data to the database, classifying the sleep stages, quantifying sleep quality, and visualizing the sleep data to the dashboard.

There are two services to receive and store sensor data: the sensor gateway and sensor data persister, which uses message brokers to communicate. When sensor data is received, an event will be published by the sensor gateway and received by the sensor data persister. This service separation is aimed to make the process of data ingestion and data storing separate and run asynchronously.

The sleep stage classification process has one service to carry out when the sensor data has been completely received and stored to the database. The process of classifying the stages of sleep involves a scheduling system that checks the data in the database. An event will be published after the sleep stage classification is completed. The proposed combination of WELM and PSO will be implemented in this service.

The process of quantifying sleep quality uses one service, namely a fuzzy algorithm. The sleep quality quantification service will react to the event of the 'finished' sleep stages classification process.

To display sleep quality data on the dashboard, there is one service with the function of retrieving the sleep quality data from the database. This service also has administrative features for registering patients.

The following are the explanations of each service.

A. Sensor Gateway

The sensor gateway is the outermost part of the IoT platform that monitors the sleep quality related to sensors. The REST protocol [9] is used. The request body is the value of the ECG data and the patient's unique sensor token which identifies the patients. The received data will be forwarded to the message broker with an "*ecgDataInput*" topic, shown on Fig. 2. By implementing microservices and event-driven architecture, the data-storing process will be asynchronous and separated from the data-receiving process. Hence, the sensor gateway is expected to have higher throughput and lower response time.

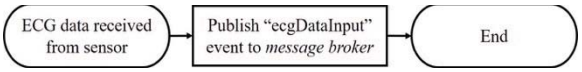

Fig. 2 The sensor gateway flowchart.

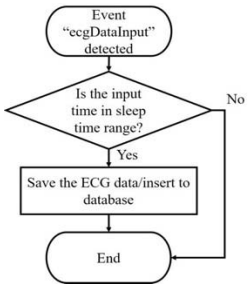

Fig. 3 The sensor data persister flowchart.

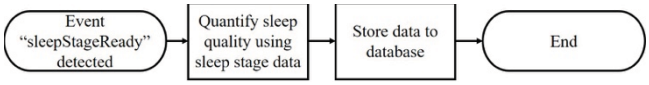

Fig. 4 The Sleep Quality Quantification Service event-based algorithm flowchart.

B. Sensor Data Persister

The sensor data persister has a function to react to the "*ecgDataInput*" event, and then validate and store the data to the database. The sensor data persister on the system is designed with the concept of the consumer group in the event-driven architecture, so that the sensor data persister is ready to be replicated to process large amounts of data from the sensor gateway.

There is only one set of ECG data entered in one day, which will be processed into sleep quality (one sleep quality data per patient per day). In the patient registration process, there is a time-range field that is used to determine the time which the ECG data will be processed. The process that occurs when an "*ecgDataInput*" event is received is shown on Fig. 3.

C. Sleep Stage Classification Service (SSCS)

The SSCS component is implemented using the proposed combination algorithm of WELM and PSO, which is an improvement over the ELM-PSO algorithm by Lesmana et al [10]. The WELM-PSO model was built using the MIT-BIH database which consists of human sleep ECG recordings [11]. The WELM-PSO is published in previous paper [7].

D. Sleep Quality Quantification Service (SQQS)

The SQQS component is implemented with the Fuzzy algorithm [12] that quantifies the level of sleep quality based on the percentage of deep sleep, awake, and total sleep duration. The event-driven data flow of SQQS is shown on Fig. 4. When SQQS receives the "*sleepStageReady*" event published by SSCS, the process of sleep quality quantification will be executed. The durations of deep sleep, awake, and total sleep are taken from the sleep stages from SSCS, and will be

Fig. 5 The Daily Sleep Stage Dashboard

used as fuzzy input by Ang et al. [12] using the jFuzzyLogic library. In SQQS, a set of data for metrics/additional information on the dashboard are also calculated from the sleep stages, e.g. awake duration, deep sleep duration, light sleep duration, REM duration, total sleep duration, time to sleep, wake time, and sleep efficiency (percentage of sleep compared to total sleep duration). The final result will be stored back to database.

E. Web Application Gateway

The web application gateway not only acts as a gateway for the web application but also facilitates health practitioners and patient registration using Angular single page application technology. In the patient registration process, a sensor token will be generated using UUID [13]. For the web application gateway, this service provides REST endpoints to retrieve ECG, sleep stage and sleep quality data for each patient.

F. Monitoring Application

The monitoring application component is a client-side web application that can be accessed by patients through a web browser. Sleep quality data can be accessed in the form of tables and graphs that provide information about the patient's sleep details on the dashboard.

III. Proposed Monitoring Dashboard

The proposed monitoring dashboard is implemented using Java, HTML, CSS, and Angular javascript frameworks. The dashboard that was developed is daily sleep stage dashboard, which are presented on Fig. 5. Each dashboard can be accessed by two types of user access, namely patients and health practitioners. Basically, both the patients' and health practitioners' dashboards share the same features. The difference lies in the patient filter for searching patients, which is only available on the health practitioners' dashboard. The daily sleep stage and sleep quality dashboards provide information on sleep stage, sleep quality, and sleep metrics data. All data on this dashboard is displayed according to the selected date/time filter.

The sleep stage is represented in the form of a 2D-array which is divided into several colors according to the stages of sleep. The sleep metrics contains information on the duration of awake time, duration of deep sleep, duration of light sleep, duration of REM, total sleep duration, time to sleep, wake time, sleep efficiency, and sleep quality.

In Fig. 5, the sleep quality is displayed as "Level 7" which means that the patient's sleep quality at the time was Level 7 of 9 determined by Ang et al. [12]. The duration of deep sleep for patients at that time was 34.46% (good), the duration of awake was 4 minutes (enough), and the total sleep duration was 2 hours 57 minutes (enough).

Information about overall patient sleep quality is displayed on the sleep stage and overall quality dashboards. The data displayed is represented in the form of a bar chart, where the y-axis is sleep quality and the x-axis is the day. The data is displayed according to the specified start and end date filters. There is also a sleep metrics which shows the awake duration, deep sleep duration, light sleep duration, REM duration, total sleep duration, time to sleep, wake time, sleep efficiency, and sleep quality.

IV. Results and Discussions

In this study, two evaluations were made to measure the performance of the proposed IoT platform using microservices and event-driven. The evaluations were performed on a single machine with 4 CPU with a clock rate of 2GHz and 15 GB RAM.

1) Sensor gateway throughput and response time evaluation, which was aimed to measure the number of transactions that can be handled by the system in a certain period of time (transaction per second or request per second). Meanwhile, response time evaluation was aimed to measure how long a transaction/request is handled by a system, which is measured since a request is sent to the server until it receives a response [14]. The sensor data will be simulated with a simulator that streams the MIT-BIH database. The results will be compared with monolith architecture and microservices architecture without event-driven/directly to the database with 1 and 2 replica applications [2]-[4].
2) Memory resources/RAM usage evaluation, which was aimed to test the memory resources consumption of the proposed architectures. The test will be compared with monolith architecture and microservices without event-driven/directly to the database using seven replica applications [2]-[4].

Table 1 Throughput and Response Time Evaluation

	Throughput (transaction/s)	Response time (ms)
1 McsED (proposed)	803	253 34
2 McsED (proposed)	1508	68.8
1 McsDB [2]	695	374.72
2 McsDB [2]	1119	155.84
1 MnlDB [3], [4]	476	596 22
2 MnlDB [3], [4]	783	280.6

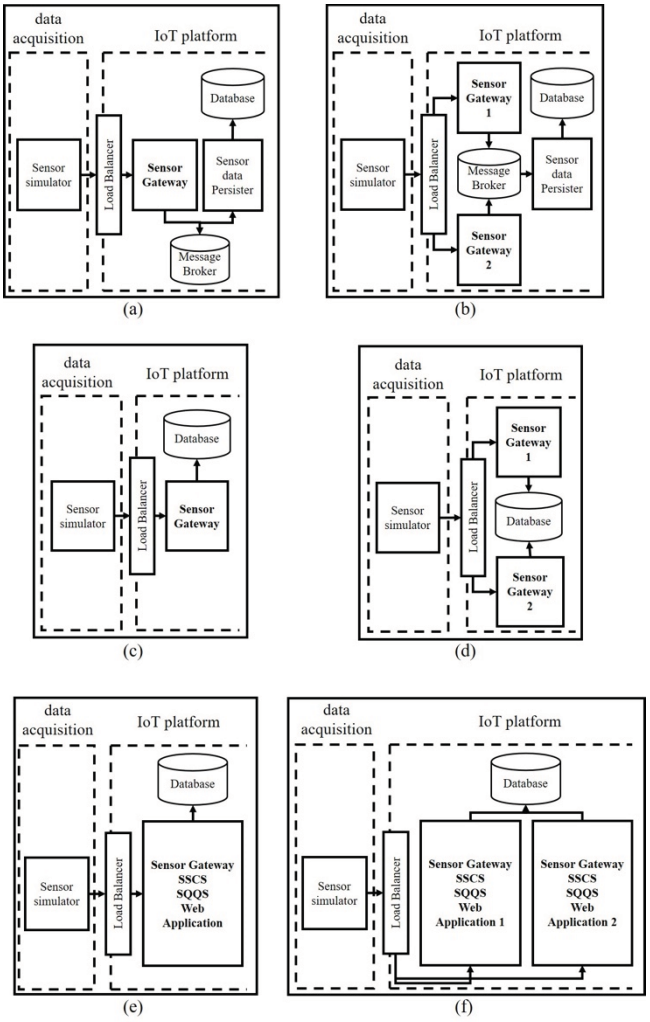

Fig. 6 The Architecture comparison for evaluation (a) 1McsED(proposed), (b) 2McsED (proposed), (c) 1McsDB [2], (d) 2McsDB [2], (e) 1MnlDB [3], [4] , (f) 2MnlDB [3], [4]

A. Throughput and Response Time Evaluation

The comparison of the evaluated architectures is shown on Fig. 6. The number of requests used for testing is 20,000 requests using Apache Jmeter. The proposed architecture for sleep quality monitoring system using microservices and event-driven (McsED) will be compared with microservices without event-driven/directly to the database (McsDB) [2], and monolith architecture (MnlDB) [3], [4] with one and two ser vice replicas. Hence, there are three architectures with two types of replica number that yield six types of tested architectures.

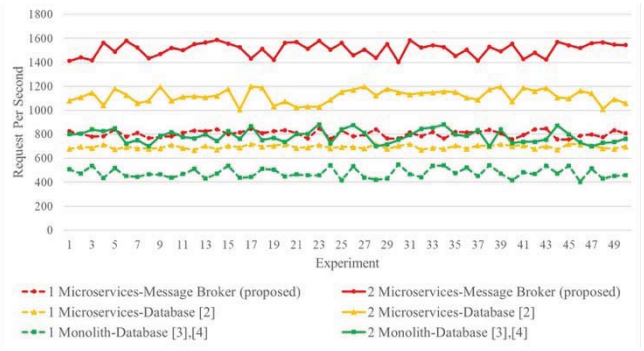

Fig. 7 Throughput Comparison Result

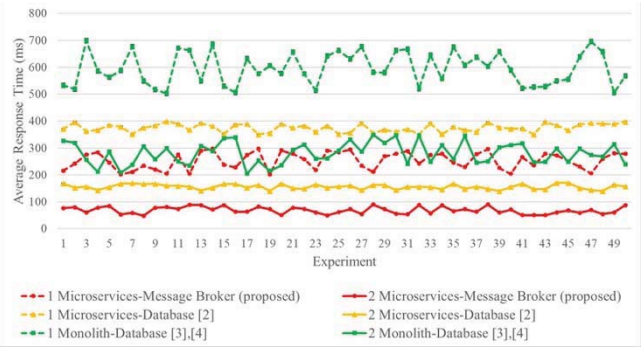

Fig. 8 Response Time Comparison Result

The results of the throughput and response time comparisons are presented on Table 1, Fig. 7, and Fig. 8. For one and two replicas, the proposed system has the best performance. A replica of the proposed architecture can process 803 sensor data with a response time of 253.34 ms, while two replicas can process 1508 transactions per second with a response time of 68.8 ms.

The implementation of microservices and event-driven architecture increases the throughput by 92.59% and decreases the response time by 75.48% for two replicas compared to the monolith architecture connected directly to the database. When compared to microservices architecture that is connected directly to the database, the use of the message broker increases the throughput by 34.76% and decreases the response time by 55.85% for two replicas. The proposed architecture had better performance in throughput and response because the gateway sensor services are separate from other functions so that it is more focused on doing its tasks and there is no need to make transactions to the database. The usage of the message broker makes the process of storing data asynchronous, which is handled by the sensor data persister service. The microservices which do not use event-driven had lower performance because the sensor gateway needs to serve two functions, namely receiving data and storing it on the database. Meanwhile, the monolith architecture had the worst performance because it needs to serve all sleep quality monitoring functions in one large application.

B. Memory Allocation Evaluation

The resource memory/RAM usage per architecture is shown on Fig. 9 The average memory consumption per instance replication for up to seven replicas of sensor-gateway replication. Equation (1) is used to calculate v , which is the

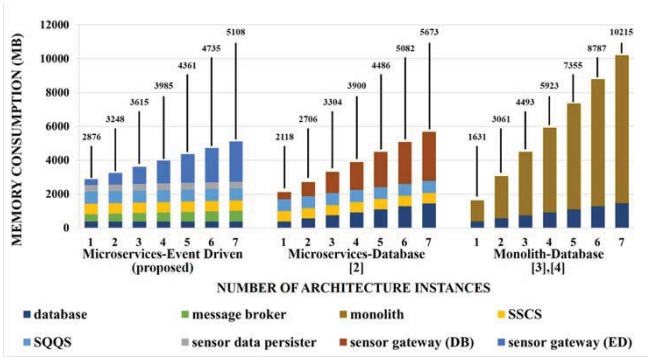

Fig. 9 The average memory consumption per instance replication

Table 2 The Memory Consumption for 7 Replicas (N=7)

Architecture	v (MB/replica)
McsED (proposed)	372
McsDB [2]	592.5
MnlDB [3], [4]	1,430.67

rate of the increase in memory consumption per instance replication, where:

$$v = \frac{1}{N}\sum_{i=i}^{N} x_{i+1} - x_i \quad (1)$$

For one instance, monolith architecture (MnlDB) has the best/most efficient memory usage. Whereas the proposed architecture, McsED, architectures have the highest memory consumption at 2,898 MB.

For two to four replicas, the lowest memory usage was found on McsDB. However, if the replica is further increased, the proposed architecture had the lowest memory consumption per added replica (v) at 372 MB in average, compared to the microservices-database (McsDB) architecture of 592.5 MB and monolith architecture (MnlDB) of 1,430.67 MB. The rate of increase in memory consumption per instance replication for seven replicas is summarized on Table 2.

The low memory usage of proposed architecture corresponds to the principle of microservices architecture which enables the replication on the required services, in this case, the sensor-gateway. In the MnlDB architecture, sensor-gateway replication cannot be done separately because monolith architecture is a single entity application. Implementing an event-driven architecture reduces the additional use of RAM resources to connect to the database. In the McsDB architecture, the addition of a gateway sensor replica (DB) increases database memory usage because the gateway sensor is connected directly to the database so that it requires additional memory allocation.

V. Conclusion

In this paper, we have presented the IoT Platform Architecture for sleep quality monitoring system. The IoT platform was designed using microservices and event-driven architecture which splits the whole system into five services: sensor-gateway, sensor-data-persister, sleep stage classification, sleep quality quantification, and a web application gateway. The services use events to communicate to each other and are sent asynchronously between the services and message broker. The evaluation results show that the proposed method can improve the throughput and decrease the response time during data transmission. On the other hand, the rate of increase in memory usage per instance replication for proposed method is lower than the traditional method.

Acknowledgment

This work is supported by the Directorate General of Strengthening for Research and Development, Ministry of Research, Technology, and Higher Education, Republic of Indonesia as a part of Penelitian Terapan Unggulan Perguruan Tinggi Research Grant to Binus University entitled “Portable Sleep Quality Monitoring Prototype and Application based on Cloud Computing Technology and Machine Learning” with contract number: 039/VR.RTT/IV/2019 and contract date: 29 April 2019.

References

[1] N. Surantha, G. P. Kusuma, and S. M. Isa, “Internet of things for sleep quality monitoring system: A survey,” in *Proceedings - 11th 2016 International Conference on Knowledge, Information and Creativity Support Systems, KICSS 2016*, 2017.

[2] M. Kim, M. Asthana, S. Bhargava, K. K. Iyyer, R. Tangadpalliwar, and J. Gao, “Developing an On-Demand Cloud-Based Sensing-as-a-Service System for Internet of Things,” *J. Comput. Networks Commun.*, vol. 2016, pp. 1–17, Aug. 2016.

[3] K. N. Swaroop, K. Chandu, R. Gorrepotu, and S. Deb, “A health monitoring system for vital signs using IoT,” *Internet of Things*, 2019.

[4] S. Mohapatra, S. Mohanty, and S. Mohanty, “Smart Healthcare: An Approach for Ubiquitous Healthcare Management Using IoT,” *Big Data Anal. Intell. Healthc. Manag.*, pp. 175–196, Jan. 2019.

[5] J. Thones, “Microservices,” *IEEE Softw.*, vol. 32, no. 1, pp. 116–116, Jan. 2015.

[6] B. Michelson, “Event-Driven Architecture Overview,” 2006.

[7] O. K. Utomo, N. Surantha, S. M. Isa, and B. Soewito, “Automatic Sleep Stage Classification using Weighted ELM and PSO on Imbalanced Data from Single Lead ECG,” *Procedia Comput. Sci.*, vol. 157, pp. 321–328, 2019.

[8] M. S. Mikowski and J. C. Powell, *Single Page Web Applications JavaScript End-to-end*. 2013.

[9] C. Pautasso, “RESTful Web Services: Principles, Patterns, Emerging Technologies,” in *Web Services Foundations*, New York, NY: Springer New York, 2014, pp. 31–51.

[10] T. Fennia Lesmana, S. Muhamad Isa, and N. Surantha, “Sleep Stage Identification using The Combination of ELM and PSO based on ECG Signal and HRV,” *2018 3rd Int. Conf. Comput. Commun. Syst.*, pp. 258–262, 2018.

[11] A. L. Goldberger *et al.*, “PhysioBank, PhysioToolkit, and PhysioNet,” *Circulation*, vol. 101, no. 23, Jun. 2000.

[12] C. K. Ang, W. Y. Tey, P. L. Kiew, and M. Fauzi, “An artificial intelligent approach using fuzzy logic for sleep quality measurement,” *J. Mech. Eng.*, 2017.

[13] P. Leach, M. Mealling, and R. Salz, “A Universally Unique IDentifier (UUID) URN Namespace,” 2005.

[14] M. M. Arif, W. Shang, and E. Shihab, “Empirical study on the discrepancy between performance testing results from virtual and physical environments,” *Empir. Softw. Eng.*, 2018.