A Study of Microservices System for Precision Medicine with Service Request Scheduling

Shivani Sanjay Kolekar
Chonnam National University
Gwangju 61186, South Korea
shivanikolekar@gmail.com

Sungwoong Yeom
Chonnam National University
Gwangju 61186, South Korea
yeomsw0421@gmail.com

Kyungbaek Kim
Chonnam National University
Gwangju 61186, South Korea
kyungbaekkim@jnu.ac.kr

ABSTRACT

One of the most promising technologies that is raised from the fourth industrial revolution is Digital Twin (DT). Digital twin technology can be used to generate a virtual twin of a hospital to review operational strategies, capacities, and treatment analysis models to identify areas of improvement, predict future challenges, and optimize medicine. The digital twin provides feedback aiming to improve the system and can present the advantage to be quicker, cheaper, and less constraining compared to mechanistic approaches. Applications of artificial intelligence have a growing interest in medical field and the use of digital twin in supporting the precision medicine is now emerging. We build a Microservices System for Precision Medicine using digital twin where services such as survival analysis, rapid response system etc. are included. The quality of service degrades as requests from large number of users take place simultaneously causing request overload. To mitigate the issue of request overload we propose the application priority-based request scheduling algorithm to the microservices system.

KEYWORDS

Digital Twin, Microservices, Distributed Systems, Big Data

1 INTRODUCTION

The concept of Digital Twin (DT) originally appeared in manufacturing literature at the beginning of 2010s, referring to a digital representation of an asset (e.g., physical objects, processes, devices) containing the model of its data, its functionalities and communication interfaces [13]. In recent years, DT has been applied to multiple different areas of industry such as smart cities and manufacturing settings (automobile industry etc.). In recent years, digital twins have become an important concept in Industry 4.0 [14], and into the process of designing and developing industrial assets, involving different research fields—from the Internet of Things (IoT) to Simulation and Artificial Intelligence. A digital twin of an industrial asset collects in real-time data provided by the asset's sensors to build a digital counterpart. Using this counterpart, it is possible carrying out simulations about the current and future state of the asset, reasoning about potential condition before they occur, or continuously collecting and analyzing data on the ongoing state of the asset to prevent unwanted situations. This can be exploited for operation optimization and for the maintenance of physical assets, systems, and manufacturing processes [1]. The Digital Twin can tackle the challenge of seamless integration between IoT and data analytics through the creation of a connected physical and virtual twin (Digital Twin). A Digital Twin environment allows for rapid analysis and real-time decisions made through accurate analytics [2].

In healthcare, DT paradigm is being explored for various purposes. Personalized medicine, surgical procedures simulation, testing and redesign drugs, resource optimization (Leveraging historical and real-time data of hospital operations and surrounding environment for e.g., Covid-19 cases, car accidents etc.) and precision medicine. Precision medicine is an emerging approach for disease treatment and prevention that takes into account individual variability in genes, environment, and lifestyle for each person. For example, AI (Artificial Intelligence) techniques are being used in precision cardiovascular medicine to understand genotypes and phenotypes in existing diseases, improve the quality of patient care, enable cost-effectiveness, and reduce readmission and mortality rates [15].

We build a docker based microservices system for precision medicine (MSPM) services such as rapid response system and survival analysis of lung cancer patients using digital twin technology. The element 3 of maturity of digital twin is applied in this system where Dynamic or operational data can be obtained and displayed in real (or near real) time through one-directional flow from the physical to the digital asset. This data can be analyzed to inform and predict the behavior of the built asset, and facilitate decision making, with the output or results fed back and updated into the organization's existing systems. The MSPM provides services which include rapid response system for abnormality prediction, lung cancer survival analysis (Deep-Surv) and binary survival classification. These artificial intelligence services are significantly helpful for the critical decision making in case of emergency medical care and oncology. This also means the system should value time constraints meticulously. As our system is deployable to the large medical staff, request traffic for the services is bound to

happen. In this context, to meet QoS (Quality of Service) expectations in the web server overload conditions is crucial.
In order to solve the above-mentioned problem, we propose Microservices System for Precision Medicine with Service Request Scheduling. We consider priority scheduling algorithm concept for the services based on the demand of the requests.

2 RELATED WORK

A digital twin is a digital representation of a physical asset reproducing its data model, its behavior, and its communication with other physical assets. Digital twins act as a digital replica for the physical object or process they represent, providing nearly real-time monitoring and evaluation without being in close proximity [1]. Although most of their concrete applications can be found mainly in the industrial context, healthcare represents another relevant area where digital twins can have a disruptive impact [1,13]. It is possible to develop personalized treatments using precision medicine with digital twin technology more rapidly, for physicians to make more precise clinical decisions, for patients with chronic conditions to receive customized treatment options that extend and improve their lives, and for resource-starved hospitals to maximize staffing, workflows, and capacity more effectively. Digital twins are also being used by healthcare and life sciences organizations around the world to fulfill the promise of personalized medicine. This includes allowing physicians to leverage digital care-backed clinical decision support solutions and potentially thousands of variables to intelligently model the best course of treatment at the point of care. In this context we study the healthcare related artificial intelligence literature works.

Ying et al. [3] proposes a cloud-based framework for the elderly healthcare services using digital twin. In this work, the author builds cloud environment for monitoring, diagnosing, and predicting aspects of the health of individuals using, for example, wearable medical devices, toward the goal of personal health management, especially for the elderly. This system has service request interface which provides interactive support for service provision, service request and platform operation. But the paper author does not consider the service request overload situation making it one of the limitations.

Jorge et al. [5] proposes using digital twin to enable the vision of precision cardiology. Authors focus on providing AI support to individual treatment and prevention of cardiovascular disease based on accurate predictions.

Shivani et al. [7] proposes web based microservice framework for survival analysis of lung cancer patient using digital twin. They propose creating a microservice based system for lung cancer survival classification. Although they propose a web service system, this paper does not consider request scheduling on the system server to provide quality of service. Angelo et al. [1] proposes integration of Agents and Digital Twins in Healthcare. Here, he attempts integration of digital twins with agents and Multi-Agent Systems (MAS) technologies. According to the paper, the technology can be used for trauma medicine and emergency care. The work is in preliminary stage therefore they lack in following the QoS in optimal request handling context.

With the growth of World Wide Web applications, Web servers have existed lots of hard issues such as poor quality of service (QoS) to client requests. Web services QoS is only just emerging as an important theme in service-oriented computing, and it is being studied in multiple directions. The notion of QoS in Web services is much broader and refers to a wider range of non-functional service characteristics. It describes how well a service is provided (e.g., reliable, available, secure, consistent, etc.). We consider QoS in the context of optimized request handling and service provisioning. Generally, Web servers may be busy to service hundreds of requests at the same time, in this case has a large of requests to be delayed or to be lost in Web servers [10]. In generally, the common Web servers handle incoming client requests based on the first-come first-served mechanism that is called FCFS. All requests correctly received are eventually handled regardless of the type of them. Such as the Apache Web servers that are one of the most widely used web servers [10].

To study the request scheduling mechanisms, we survey following literature. Karamcheti et al. [8] authors consider QoS expectations of the users and propose a server-side request scheduling mechanism that addresses these problems. Here, Reward-Driven Request Prioritization (RDRP) algorithm gives higher execution priority to client web sessions that are likely to bring more service profit (or any other application-specific reward). The method is based on reward driven reinforcement learning technology. David Olshefsk [16] presented an online server-based mechanism that is called the Client Response Time Estimated by the Server. His mechanism allows Web servers to estimate mean client perceived response time via measured client. Lee et al. [17] considered a Web server that can provide differentiated services to clients with different quality of service (QoS) requirements. His method can provide N classes of proportional delay differentiated services (PDDS) to heterogeneous clients. With PDDS, Web servers can provide consistent performance spacing over a wide range of system loading and this simplifies many pricing issues. The variation of contexts in which a Web service could be used and the resulting variation in Quality of Service (QoS) expectations makes a clear case for further research to extend Web services management platforms with more sophisticated control mechanisms to cater for multiple service offerings [6]. To implement and improve the QoS of the web-based system for precision medicine we propose Microservices System for Precision Medicine with Service Request Scheduling.

We build the microservice system which uses different containers to store the precision medicine based medical deep learning models and provide access to the medical staff. The MSPM consists of main elements which are web server, container server, docker containers and different medical services. The digital twin, in this case, is meant to be useful for realizing more effective care interventions, helping Physicians and other intersecting care technologies in understanding the survival rate of the patient. In this context, the main research question that we consider in this paper is about constructive simulation of DT with Microservice based precision medicine services system while considering QoS in context of optimal request scheduling and management.

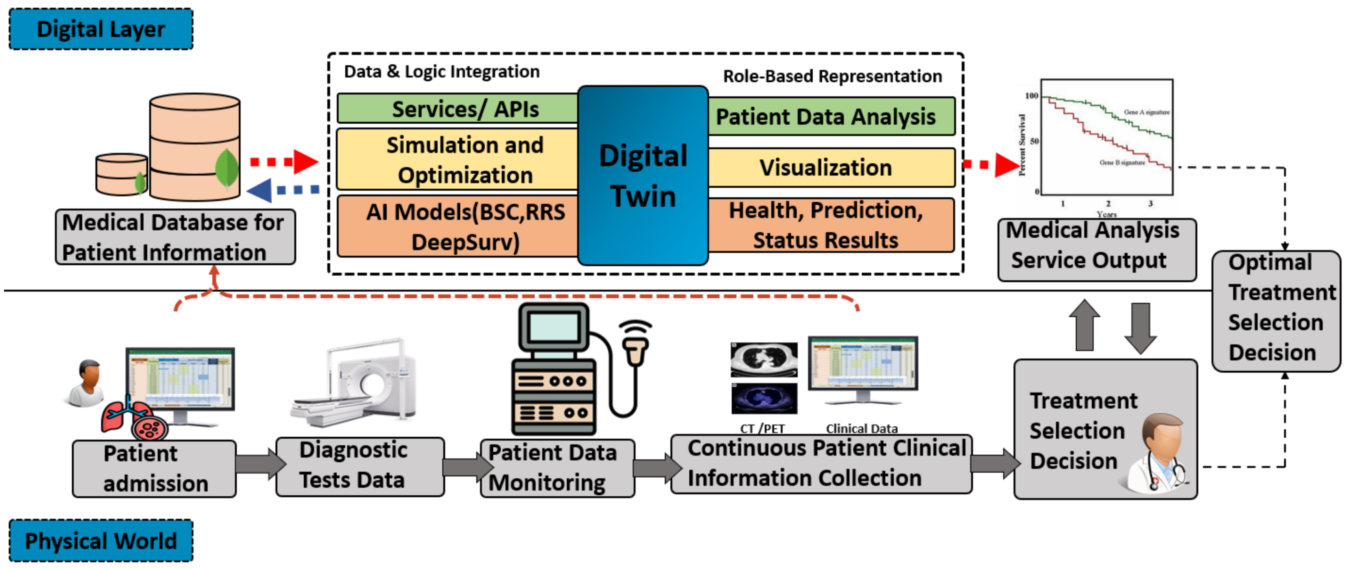
Digital Layer
Data & Logic Integration
Role-Based Representation
Services/ APIs
Simulation and Optimization
AI Models(BSC,RRS DeepSurv)
Digital Twin
Patient Data Analysis
Visualization
Health, Prediction, Status Results
Medical Database for Patient Information
Medical Analysis Service Output
Optimal Treatment Selection Decision
Patient admission
Diagnostic Tests Data
Patient Data Monitoring
CT /PET
Clinical Data
Continuous Patient Clinical Information Collection
Treatment Selection Decision
Physical World

Figure 1. Overview of precision medicine services provisioning using digital twin

3 MEDICAL MICROSERVICES SYSTEM WITH SERVICE REQUEST SCHEDULING

In order to build an integrated platform for artificial intelligence models created for oncological and emergency medicine decision support, we propose creating a microservices based precision system medicine service system. One of the main challenges this type of faces is large database handling along with request handling when it has huge number of users such as medical staff of the healthcare organization.

3.1 Microservices System for Precision Medicine

To build Microservices System for Precision Medicine (MSPM) we consider use of docker to create the container based microservices. Microservices also known as the microservice architecture is an architectural style that structures an application as a collection of services that are highly maintainable, loosely coupled and independently deployable on multiple machines. The purpose of using docker containers for the system is flexibility in deployment. This helps to deploy the different precision medicine services in independent docker containers and can be used a microservices running independent of each other. We divide MSPM in 5 main components – MSPM User interface, web server, medical database, MSPM containers and container server.

3.1.1 MSPM UI. To create a platform for users to utilize the services we create MSPM user interface using Flask and RestfulAPI. We provide patient wise information on the UI with the help of MongoDB database. The UI consists of service options viz. rapid response system, lung cancer survival analysis and binary survival classification for lung cancer patients etc. The medical staff can access the patient pathological tests information with the individual patient ID. We provide open access dicom viewer to visualize radiological scans such as CT, PET and MRI data. The users can study about the patient's condition according to the updated information on the UI. The user can also upload the clinical information of a new patient and perform the service analysis. The output of the service such as lung cancer survival analysis Kaplan Meirs graph, rapid response abnormality prediction for each 8-24 hours can be analyzed in this system UI comfortably.

3.1.2 MSPM Web Server. The main web server is one of the most important components of MSPM as it handles the request-response flow for the users. The webserver uses flask route mapping for request routing. Service request handling is also performed in the web server. Database handling is also performed here. Basically, overall communication requests from user to the database and service handler and sending communication response back to the user is handled by the web server. The user is provided with a unique ID to store the requests per each user. When the user selects the patient ID, we ask user to also select preferred data visualization. If the user opts for visualizing the radiological scans, we send a request to the database for the images. These images are then fetched from the medical database using flask route mapping. Then as a response the images are sent to the user interface through this web server fulfilling the user request.

3.1.3 Medical Database. In order to store the huge datasets of the deep learning models for precision medicine services we opt to use MongoDB database. MongoDB is a source-available cross-platform document-oriented database program. The user requests are identified with the unique ID and this ID is stored in the database along with the request output.
We divide the database according to the data type such as the radiological imaging dataset needs to be stored with the help of GridFS and clinical categorical information can be stored in

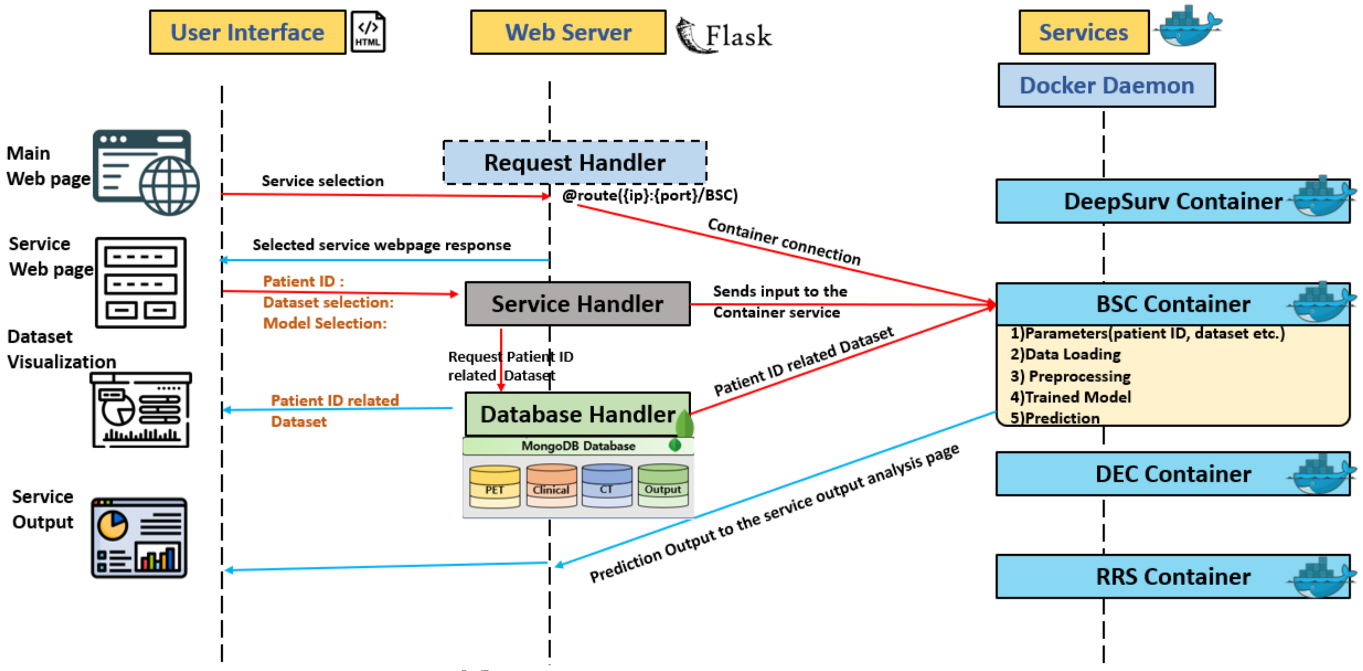

Figure 2 MSPM Request-Response Workflow

service container through container server. This database handles all sorts of data that needs to be stored for this system.

3.1.4 MSPM Containers. The precision medicine services that are provided in the UI menu are all securely stored in the docker containers. A container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings. The docker daemon is The Docker daemon (dockerd) listens for API requests and manages Docker objects such as images, containers, networks, and volumes. A daemon can also communicate with other daemons to manage Docker services.

In order to store the service software in the docker we first create docker image for the deep learning model and tag it with its name for example RRS_Image [18]. This image consists of dockerfile (consists of dependencies of the software) and container server (Software file). This becomes the base image for the container. This means we can run as many containers as possible using the same image and make modifications to them. All the images created are currently independent of each other. The container server handles all the request-responses for the container. In this system, we provide services such as rapid response system (RRS) [18]. It is a method that detects the in-hospital abnormal status of patients based on basic vital signs and blood test values. The service provides abnormal status every 8,16 and 24 hours. Binary survival classification of lung cancer patients (BSC) [19] is also one of the services integrated into the microservices system. The survival is predicted into two categories as event (death) occurred and event not occurred in 5 years of period. This service uses multimodal data including radiological data CT, PET images and clinical categorical information of lung cancer patients. Services also include DEC based survival patients clustering where survival period of patients is predicted based on clustering of lung cancer T, N and M staging [20].

3.1.5 MSPM Container Server. The container server for MSPM handles the service requests from the user. After user selects the service, the container server gets activated. Here, MongoDB database connection is created, and the medical database stored in it is accessed. The container server also performs the preprocessing of the raw data as it is provided directly from the database. After performing preprocessing, the preprocessed data is then processed through the pretrained deep learning model of the service. Here, after certain amount of time of deep learning model processing, the output data is created. This output is then saved in the MongoDB database and routed to the user interface. It becomes very important to compel the database types and accessibility of the pretrained model in case of container server.

3.1.6 MSPM Request-Response Workflow. Further we explain the workflow of these components through figure 2. First, the user interface is deployed on the specific IP address of the service. The user selects the service he needs through the user interface. Then he proceeds with the selection of patient information such as patient ID and dataset type selection for visualization then clicks the submit button. This request is then listened and processed by the request handler at the web server. Request handler immediately checks for the available database of the patient ID and sends it to the UI for the purpose of visualization. Then the request for the service is processed. Here, the main web server sends the patient ID information from user to the container server. The container server fetches

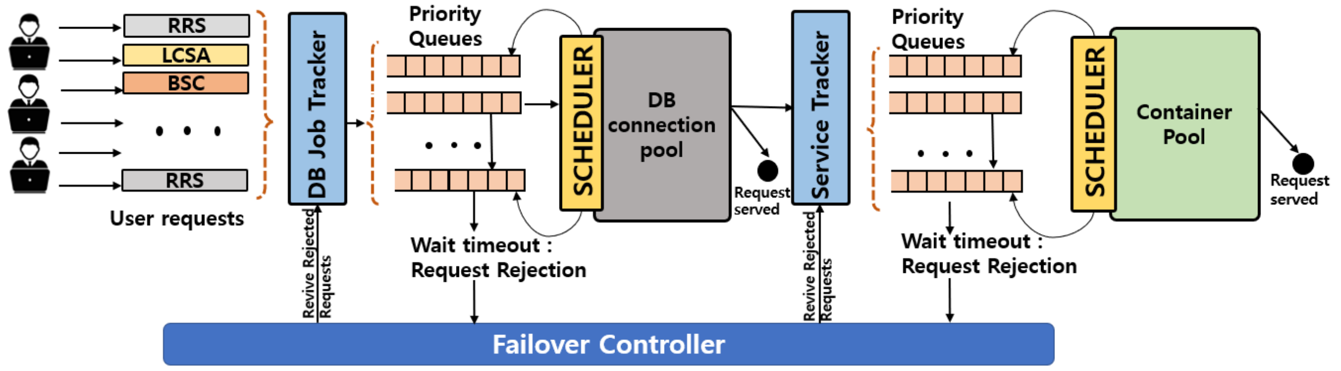

Figure 3. Web Server Request Scheduling Architecture

the raw data related to the patient and selected service, preprocesses it and runs through the pretrained model to get the output. The output is achieved according to the specific time amount of the software model. The output is then saved to the database with the unique user request ID. The service output is then visualized in the user interface making it the final step in the request-response workflow of MSPM. In order to fulfil this request-response workflow, it is essential to keep running the containers in the background so that they are able to listen to the requests. One or more users can select one or more services at a time. The service containers for each service are able to run in the background parallelly.

3.2 MSPM with Service Request Scheduling

In the medical field, artificial intelligence is being used many times. This also includes AI services based on precision medicine. As main user group for this type of services most likely the medical experts. The services we provide through the MSPM system are all time critical services. Which creates a necessity of providing the output of each service to each user on time while keeping the optimum quality of service. This task becomes difficult when there are large number of users at a time requesting the services. This becomes even more complicated when all the requests are dependent on the processing time of the model.

In order to solve this problem, we propose a web server request scheduling architecture for MSPM. We use the basic priority queue algorithm by assigning priority to each of the services for the execution. We assign the priority according to the following rules.

3.2.1 Service Priority Rules

1) Importance of Service. We decide which medical AI service is more important than the other according to the demand and the chances of being prioritized for the execution.

2) Response Time of Service. According to the response time of the service we prioritize the service. Here, lower the service time, increase in the priority value rule is followed.

3) Requests demand. As the number of requests for a specific service increase, we prioritize the service. Higher the demand higher the priority consideration.

According to the figure 3, the requests are processed. The DB job tracker tracks the service requests for database and according to the priority whereas, the service tracker keeps track of service selection requests and follows the priority queue. In case of time out while waiting in the queue the request gets failed. But these failed requests can later be accepted by failover controller. Fail over controller adds the failed requests back to the queue to be processed.

The requests from user then valued for priority queue using weighted priority approach. In priority queueing [2], separate queues are used for packets of different classes of traffic (different priorities). Packets for transmission (over the shared communication channel) are always selected starting from the (nonempty) queues of highest priority. Consequently, packets from lower priority queues are selected only if all higher priority queues are empty. This can block the lower priority classes of traffic for extended periods of time if the traffic is intense.

Weighted priority scheduling limits the number of consecutive packets of the same class that can be transmitted over the channel; when the scheduler reaches this limit, it switches to the next nonempty priority queue and follows the same rule. These limits are called weights and are denoted ω_1 . With k classes of traffic, if there are enough packets in all classes, the scheduler selects w1 packets of class 1, then ω_2 packets of class 2, ..., then wk packets of class k, and again ω_k packets of class 1, and so on. Consequently, in such a situation (i.e., for sufficient supply of packets in all classes), the channel is shared by the packets of all priority classes, and the proportions are:

$$u_i = \frac{\omega_i \,/\, s_i}{\sum_{j=1,\cdots k} \omega_j \,/\, s_j}, i = 1,\dot{2}, \dots, k$$

Where, s_i, i =1..., k is the transmission rate for packets of class i. If the transmission rates are the same for packets of all classes (as is assumed for simplicity in the illustrating examples), the proportions are:

$$u_i = \frac{\omega_i}{\sum_{j=1,\cdots k} \omega_j}, i = 1,\dot{2}, \dots, k$$

4 CONCLUSIONS

In this paper, we have proposed microservices system for precision medicine with service request scheduling. This system includes microservices system for deep learning models using docker container and the request-response module where the application of the user requests handling with the help of weighted priority queue scheduling is carried out. The proposed system handles large number of requests from multiple users of medical organization. However, when number of requests are fluctuated on the large scale, the system is not able to consider scalability for scheduling of the services. In future, we plan to solve this limitation by developing a scalable solution and testing with different scheduling algorithms.

ACKNOWLEDGMENTS

This research was supported by the Bio & Medical Technology Development Program of the National Research Foundation (NRF)& funded by the Korean government (MSIT) (NRF-2019M3E5D1A02067961). This research was supported by the MSIT (Ministry of Science and ICT), Korea, under the Innovative Human Resource Development for Local Intellectualization support program(IITP-2022-RS-2022-00156287) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation).

REFERENCES

[1] Croatti, Angelo, et al. "On the integration of agents and digital twins in healthcare." Journal of Medical Systems 44.9 (2020): 1-8.

[2] Fuller, A., Fan, Z., Day, C. and Barlow, C., 2020. Digital twin: Enabling technologies, challenges, and open research. IEEE access, 8, pp.108952-108971.

[3] Liu, Ying, et al. "A novel cloud-based framework for the elderly healthcare services using digital twin." *IEEE access* 7 (2019): 49088-49101.

[4] Lareyre, Fabien, et al. "Using digital twins for precision medicine in vascular surgery." Annals of vascular surgery 67 (2020): e577-e578.

[5] Corral-Acero, Jorge, et al. "The 'Digital Twin'to enable the vision of precision cardiology." European heart journal 41.48 (2020): 4556-4564.

[6] Madni, Azad M., Carla C. Madni, and Scott D. Lucero. "Leveraging digital twin technology in model-based systems engineering." Systems 7.1 (2019): 7.

[7] Kolekar, S. S., Yeom, S., Choi, C., & Kim, K. (2021). Web based Microservice Framework for Survival Analysis of Lung Cancer Patient using Digital Twin. In Proceedings of the Korea Information Processing Society Conference (pp. 537-540). Korea Information Processing Society.

[8] Totok, Alexander, and Vijay Karamcheti. "Improving performance of internet services through reward-driven request prioritization." 200614th IEEE International Workshop on Quality of Service. IEEE, 2006.

[9] Dilanka, Gayan, et al. "A novel request handler algorithm for multi-access edge computing platforms in 5g." 2022 IEEE 19th Annual Consumer Communications & Networking Conference (CCNC). IEEE, 2022.

[10] Huang, Guimin, and Ya Zhou. "A request handling mechanism with the shortest processing-time first in web servers." 2008 International Conference on Computer Science and Software Engineering. Vol. 3. IEEE, 2008.

[11] Zhou, Hao, et al. "Overload control for scaling wechat microservices." Proceedings of the ACM Symposium on Cloud Computing. 2018.

[12] Erradi, Abdelkarim, Srinivas Padmanabhuni, and Niranjan Varadharajan. "Differential QoS support in web services management." 2006 IEEE International Conference on Web Services (ICWS'06). IEEE, 2006.

[13] Grieves, M., and Vickers, J., Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems, pp. 85–113. Cham: Springer International Publishing, 2017.

[14] . Schwab, K., The Fourth Industrial Revolution Hardcover. Crown Business, 2017.

[15] Davenport, Thomas, and Ravi Kalakota. "The potential for artificial intelligence in healthcare." Future healthcare journal 6.2 (2019): 94.

[16] David Olshefsk; Jason Nieh; Dakshi Agrawa; "Using Certes to Infer Client Response Time at the Web Server", ACM Transactions on Computer Systems (TOCS), Volume 22 , Issue 1, February 2004, pp. 49-93.

[17] Sam C. M. Lee; John C. S. Lui; David K. Y. Yau; "A Proportional-Delay DiffServ-Enabled Web Server: Admission Control and Dynamic Adaptation", IEEE Transactions on Parallel and Distributed Systems, Volume 15 , Issue 5, May 2004, pp.385-400

[18] Nguyen, Trong-Nghia, Thanh-Hung Vo, Bo-Gun Kho, Guee-Sang Lee, Hyung-Jeong Yang, and Soo-Hyung Kim. "Deep Interpretable Learning for a Rapid Response System." In *Proceedings of the Korea Information Processing Society Conference*, pp. 805-807. Korea Information Processing Society, 2021.

[19] Choi, Chul-woong, et al. "Review of Lung Cancer Survival Analysis with Multimodal Data." Proceedings of the Korea Information Processing Society Conference. Korea Information Processing Society, 2020.

[20] Choi, Chulwoong, and Kyungbaek Kim. "Accessing the Clustering of TNM Stages on Survival Analysis of Lung Cancer Patient." Smart Media Journal 9.4 (2020): 126-133.