

ADVANCED DATA ANALYTICS USING THREE-STAGE INTELLIGENT MODEL PIPELINING FOR CONTAINERIZED MICROSERVICES IN 5G NETWORKS AND BEYOND

Takaya Miyazawa¹, Ved P. Kafle¹, Yusuke Yokota², Yasushi Naruse², Hitoshi Asaeda¹

¹Network Research Institute, National Institute of Information and Communications Technology, Tokyo, Japan, ²CiNet, Advanced ICT Research Institute, National Institute of Information and Communications Technology, Kobe, Hyogo, Japan

NOTE: Corresponding author: Takaya Miyazawa, takaya@nict.go.jp

Abstract – The recent rapid advancement of cloud-native networking infrastructure has leveraged the resource virtualization technology of containers to realize diverse microservice-based applications in 5G/6G networks and clouds. Containers drastically enhance the efficiency of computational resource allocation and utilization as compared to the related virtualization technology of Virtual Machines (VMs). The networking environment leveraging both VM and container mixed virtualization technologies makes the most use of them to realize a computational platform whose resources can be dynamically adjusted to a fine granularity. To continuously meet the required levels of quality of services in 5G/6G networks and clouds in that platform, an agile and autonomous data analytics system in the control plane is essential for the accurate prediction of server workloads and dynamic allocation of enough amount of computational resource. In this paper, we introduce a framework, which complies with Recommendation ITU-T Y.3177, for autonomous computational resource control and management. The framework consists of an advanced data analytics system and a resource control system. We propose an architecture for the advanced data analytics system consisting of learning and prediction components. The learning component includes a three-stage intelligent model pipelining with three cascaded machine learning models, nonlinear regression, clustering, and multiple regression. These models determine the fluctuation trends in CPU utilization, classify services with similarities in the trends, and predict the peak CPU utilization of each containerized microservice. We evaluate the proposed models through experiments and numerical analysis. The results prove that the models support agile data analytics, which can complete data processing in the time granularity of seconds and achieve higher prediction accuracy of CPU utilization.

Keywords – Data analytics, intelligent model pipelining, machine learning, microservice, resource management

1. INTRODUCTION

Network Function Virtualization (NFV) technologies have already been essential for fifth generation (5G) mobile networks and will also be essential for beyond 5G or 6G networks and clouds [1]–[3], to accommodate a large number of diverse application services in common networking and computing infrastructures. NFV enables us to implement any Network Functions (NFs) of traditional network equipment through software on network virtualization infrastructures consisting of general-purpose servers at reduced CAPEX/OPEX because multiple NFs can be installed and operated from a common physical machine and/or Virtual Machine (VM). Virtual functions constituting a monolithic application deployed on VMs are called Virtual Network Functions (VNFs), while the ones deployed in containers in the cloud are called container network or Cloud-Native Functions (CNFs), which correspond to microservices [4].

VM and container technologies have their own advantages. For example, VM technologies virtualize physical servers at the hardware level. This feature makes it easier to meet the diverse Quality of Service (QoS) requirements of 5G/6G networks and clouds such as guaranteed communication capacity and low latency, and also to secure virtual servers against unauthorized access. Container technologies are more effective for applications that re-

quire communication between multiple microservices, as they virtualize processes on a common OS. Microservice architecture is a particular way of designing an application service as a suite of independently deployable microservices. Different microservices can be deployed in different containers. Microservices have several advantages such as being lightweight, having high agility in the activation and deactivation of containers, excellent portability, and high scalability.

Because both virtualization technologies have different advantages, they can be mixed in future server virtualization environments to optimally meet the application service requirements. For instance, an application that requires a high quality of service and security can be deployed in a separate VM, which is not shared with other applications. Similarly, multiple applications consisting of a large number of microservices can be deployed in various containers on the same VM. Containers are expected to enhance service provisioning and management by adopting microservice-based application architectures for networks and clouds [5].

In 5G/6G networks and clouds, the rapid spread of cloud-native environments accommodating diverse application services will make the operation of servers more complex due to the drastic increase in the number of controlled and managed elements and the diverse require-

ments of personalized services [6]. Moreover, because the amount of computational resource such as CPU, memory, and storage are limited in each server, we need to efficiently use the resources to meet the QoS requirements, while responding to the fluctuations of workload and/or the occurrence of failures [7]. For this purpose, we require advanced data (e.g., resource usage) analytics technologies and autonomous resource adjustment technologies. Many prior studies have investigated various approaches for the control and management of cloud-native environments and microservices [6]–[21]. Reactive approaches to dynamically adjust computational resources can be straightforward, but they may not be able to agilely respond to rapid fluctuations in resource usage; thus, increasing the occurrence of resource shortage. Furthermore, the trends of fluctuations vary depending on the characteristics (e.g., physical specifications and management tools) of server virtualization infrastructures running microservices and these trends may be difficult for humans to predict due to their high complexity. Hence, technologies for the agile demand prediction of computational resources by advanced data analytics utilizing Machine Learning (ML) techniques will be essential, and their prediction accuracies need to be high. The application of ML techniques to data analytics contributes to satisfying various levels of QoS requirements in 5G/6G network and cloud systems based on their capability to enhance computational resource control and management by autonomously suppressing resource shortage with high efficiency of resource utilization, as discussed in [22]–[27]. Such enhancement can also be achieved for the resource control and management of microservices (such as CNFs) by exploiting the advantages of ML techniques, such as reducing the time required to find an appropriate solution for computational resource adjustments compared to existing complex optimization approaches (as discussed in [26]), and enabling proactive resource control through training and prediction processes. In this paper, we first introduce a framework of computational resource control and management for CNFs to provide microservice-based application services. The framework is compliant with Recommendation ITU-T Y.3177 [28]. Next, we propose the architecture for an advanced data analytics system included in the control and management framework, and a three-stage intelligent model pipelining that cascades the following three ML models in the learning component of the data analytics system: 1) nonlinear regression, 2) clustering, and 3) multiple regression. To the best of our knowledge, there is no prior study on a functional architecture and detailed modeling to realize dynamic, pipelined deployment of several ML models for advanced data analytics of many CNFs (i.e., microservices) constituting a microservice-based application to reflect in autonomous and agile computational resource adjustments. To confirm the effectiveness of the proposed models, we have performed experimental mea-

surements and numerical analyses of CPU utilization of each microservice. The results show that our ML models achieve agile data analytics in the time granularity of seconds and reduce the prediction errors of CPU resource utilization. In summary, our main contributions through this study are clarified as follows:

1. In the autonomous resource control and management framework for containerized microservices that is compliant with Recommendation ITU-T Y.3177, we propose an architecture of a data analytics system consisting of learning and prediction components to analyze many CNFs constituting microservice-based applications in parallel. The system is capable of accurately predicting the CPU utilization in response to fluctuations in resource usage for each microservice.
2. For the learning component, we propose a three-stage intelligent model pipelining consisting of a cascade of three ML models, each of which is explained in points 3, 4, and 5 below.
3. We propose a nonlinear regression analysis model applying a Least Squares Support Vector Regression (LS-SVR), to quantitatively evaluate the Mean Square Errors (MSEs). It obtains the trends of fluctuations in CPU utilization in an agile manner in the time granularity of seconds for each microservice with a visually understandable and generalized nonlinear function. The main benefit of using LS-SVR as a nonparametric ML model is that it does not require the estimation of a nonlinear function in advance of the regression process, which reduces the number of manual operations.
4. We propose an agile clustering model by applying a k-means++ algorithm. This clustering method classifies the microservices constituting each application based on their similarities, which simplifies the operations of many microservices by defining a small number of analytic models and resource allocation policies for all microservices. The clustering results can be obtained with a time granularity of seconds.
5. We propose a multiple regression analysis model applying the Least Absolute Shrinkage and Selection Operator (LASSO) followed by the Least Squares Method (LSM) to predict the maximum value of CPU utilization in a future specific period for each microservice. We quantitatively evaluated MSEs, the amount of overestimation, and the underestimation of peak CPU utilization. The results show that it reduces the MSEs and enables us to predict the future peak utilization better than other existing methods, which are adopting the average of a set of maximum values, adopting the largest one in a set of maximum values, LSM only, and LASSO only, in the time granularity of seconds.

The remainder of this paper is organized as follows. We introduce related work in Section 2. In Section 3, we discuss a framework for autonomous computational resource control and management for many containerized microservices. We present an architecture of the data analytics system to analyze CPU utilization of many microservices constituting each application in parallel in Section 4, and the proposed three-stage intelligent model pipelining in Section 5. We evaluate our proposed models quantitatively using numerical analysis to measured CPU utilization of three applications in Section 6. Finally, in Section 7, we conclude this paper and provide an overview of our future work.

2. RELATED WORK

A previous study [29] introduced a framework for computational resource control management for VNFs deployed on VMs. The authors proposed an architecture for a data analytics system that includes a two-stage intelligent model pipelining with two cascaded ML models, namely nonlinear regression and multiple regression. Nonlinear regression was used to assess the trends of fluctuations in CPU utilization in the time granularity of seconds for each VNF with a visually understandable and generalized function model. Multiple regression was used to predict the maximum value of CPU utilization of the VNF to allocate sufficient CPU resources in advance such that CPU saturation causing degradation of QoS could be avoided. We measured the CPU utilization of an in-network cache function [30],[31] as a VNF, quantitatively evaluated our ML models for data analytics on CPU utilization, and proved that our models are effective in enhancing the agility of data analytics and reducing the prediction errors of peak CPU utilization. However, the framework and architecture specified in [29] lack mechanisms for handling CNFs (i.e., microservices), so they are unsuitable for microservice-based applications in 5G/6G networks and clouds. The main differences between the proposal in this paper and the system proposed in [29] are that (i) the data analytics system proposed in this paper analyzes many CNFs (i.e., microservices) in parallel, rather than analyzing a single VNF, and (ii) it adds a clustering model as a second stage of pipelining (i.e., after nonlinear regression and before multiple regression) to classify multiple microservices constituting each application.

Data analytics for autoscaling of microservices are key challenges to be addressed in order to realize future cloud-native environments. Kubernetes has already equipped a threshold-based horizontal autoscaling function for containers [5], and the authors in [8] proposed a controller-agnostic algorithm based on the autoscaling of Kubernetes. However, Kubernetes dynamically increases or decreases the number of containers allocated to each microservice using a reactive approach that monitors computational resource usage and compares it to a predefined threshold. Therefore, it does not utilize intel-

ligent data analytics for autoscaling. The authors in [9] proposed combining both horizontal and vertical scaling approaches and showed its complementary benefits compared to Kubernetes's horizontal autoscaling. However, they have only adopted a reactive approach as in [5]–[8], which does not utilize intelligent data analytics.

The authors of [10] proposed a microservice-based architecture for industrial data analytics. Their goal was to predict and control the paths of moving robots in several tens of milliseconds, which differs from workload prediction for the proactive autoscaling of each microservice on server virtualization infrastructures in the cloud. Additionally, their method monitors the number of unprocessed messages and scales up subscribing microservices using a threshold-based (i.e., reactive) approach, but they did not specify any algorithm or mathematical modeling for path prediction and scaling. The authors in [11] proposed a model-driven method using mixed-integer nonlinear programming and a genetic algorithm. However, it is generally known that the two algorithms will take a long time to obtain a solution because of the strict optimization, which is not suitable for agile data analytics responding to time-varying and rapid fluctuations in computational resource usage.

Reactive approaches make it difficult to deal with rapid fluctuations in CPU utilization; and thus, a proactive approach utilizing intelligent data analytics is required for the autoscaling of microservices. The authors in [7] proposed a predictive autoscaling of microservices for cloud-native environments in mobile networks. However, they only used classical methods such as ARIMA and linear regression and did not assume the utilization of more advanced technologies such as ML techniques. The proposed autoscaling was verified using emulation data (i.e., not real data), which may have a simple probabilistic distribution of data generation. In such a case, it might be enough to apply such classical methods, but we need to utilize advanced technologies such as ML techniques to autonomously analyze complicated fluctuations in CPU utilization of every microservice. The authors in [6] proposed a framework of AI-based resource-aware orchestration, which aims to construct autonomous cloud-native environments with closed-loop automation utilizing ML techniques, the ZSM concept, and so on. They also introduced a simulator to simulate the deployment of cloud-native clusters. However, they only proposed the framework, and thus did not sufficiently discuss detailed algorithms and models for advanced data analytics for microservices.

The authors in [12] proposed the application of a reinforcement learning technique for horizontal scaling and container deployment scheduling. Likewise, the authors in [13] proposed applying reinforcement learning to learn a threshold for autoscaling in consideration of resource demands and quality of services. However, when using the reinforcement learning technique, it is imperative to execute iterations (i.e., trials and errors) many times so that the learning can converge. This means that it takes

too much time to complete each reinforcement learning task, and thus is not suitable for agile data analytics responding to rapid fluctuations in computational resource usage.

The authors in [14] proposed a resource-provisioning method using workload clustering in a cloud environment and the authors in [15]–[17] proposed the application of supervised learning techniques for autoscaling with workload forecasting. However, these studies did not investigate the pipelining of several ML models, particularly nonlinear regression and multiple regression, for advanced data analytics to enhance workload prediction and resource adjustments for microservices.

Except for data analytics for autoscaling, the authors in [18][19] proposed service function chaining comprising multiple distributed CNFs, and discussed traffic steering such as load balancing. The authors in [20] proposed a framework for performance monitoring of segment routing for the IPv6 (SRv6) method in cloud-native environments. The authors in [21] proposed a method to enhance the security of cloud-native environments. However, none of these studies focused on the data analytics for autoscaling of microservices.

Federated Learning (FL) has recently received significant attention [32][33]. The FL technique deploys ML models in multiple servers and devices across a network in a distributed manner and trains the ML models on their respective servers and devices without exchanging datasets. Additionally, it does not require the collection of datasets in a centralized server. Although FL deploys ML models in a distributed manner, no method has been defined for pipelining multiple types of ML models to enhance the performance of data analytics.

In ITU-T standardization, ML marketplace and pipeline have been propounded and standardized [28][34], but standardized documents have not recommended any specific analysis models. It is also necessary to interconnect with the framework of the ML marketplace standardized in [28][34], so that any ML function can be deployed on any model in the system.

Online Machine Learning (OML) has recently received significant attention [35][36]. The OML technique executes training and updating the model sequentially by using the new dataset received one after another, and thus, is effective especially in time-series data analytics including the prediction of future data. The OML has several advantages such as a smaller memory usage and a low computation load in training the model because a new dataset is input to the system one by one and old datasets can be removed. However, it has a serious issue of low tolerance to noise data compared to a Batch Learning (BL) because the prediction accuracy can significantly be affected by the noise. Therefore, we require some framework to solve the issue of OML.

3. FRAMEWORK OF COMPUTATIONAL RESOURCE CONTROL AND MANAGEMENT FOR CNFS

We propose a framework of autonomous computational resource control and management for containerized microservices that is compliant with Recommendation ITU-T Y.3177 [28], as shown in Fig. 1. When we run a microservice-based end-user cloud application on a VM/container mixed server virtualization infrastructure, each microservice is deployed on a unique container created in a VM. We only focus on CPU utilization as a computational resource under the condition that every microservice in an application can share and use enough RAM resources. The framework includes an advanced data analytics system and a resource control system, both of which are indispensable to quickly respond to the fluctuations of CPU utilization. The CPU utilization of every microservice is regularly monitored, and the measurement data is sent from the server virtualization infrastructures to the advanced data analytics system. The prediction results and/or external data (e.g., network and service requirements) are fed to the resource control system that makes decisions for resource arbitration and NF migration. The server virtualization infrastructures are reconfigured in accordance with the decision for computational resource adjustments.

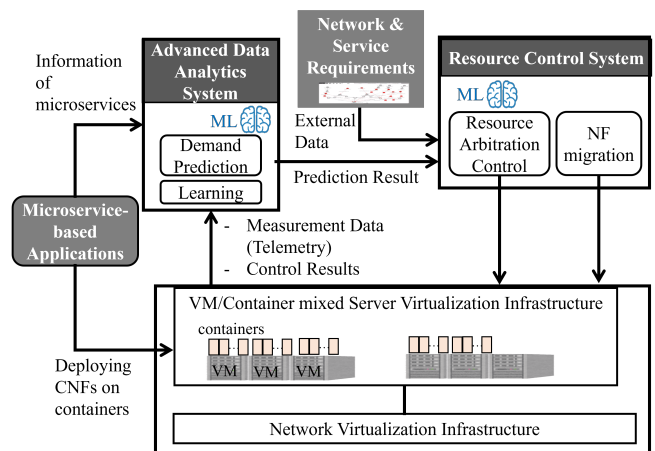


Fig. 1 – Framework of autonomous computational resource control and management for CNFs which is compliant with Recommendation ITU-T Y.3177

3.1 Advanced data analytics system for CNFs

The data analytics system for CNFs consists of the learning and demand prediction components, as is the case with the VNF in [29]. The learning component determines a CPU utilization prediction model for each microservice based on the monitoring data of the microservice's past CPU utilization by means of various ML techniques. Similarly, the demand prediction component predicts the future CPU utilization for each microservice using the current prediction model. The prediction model is updated by the validation and relearning process of the learning

component every time a new dataset is added to the past datasets.

Next, we specify the policy for data analytics as follows:

1. We dynamically select an analytic model suitable for a purpose (e.g., peak workload prediction) considering the characteristics of the data, including the trends of fluctuations in CPU utilization. Note that we cannot create an analytic model that is best for all purposes. Therefore, depending on the purpose, some cases may require ML techniques, while other cases may not require them.
2. We define a prediction model for each microservice individually because the trends of the fluctuations in CPU utilization are different depending on the microservices. This makes it easier to perform fine-grained management, data analytics, and resource control, which are suitable for each microservice.
3. We simplify the operations of microservices as much as possible because there is no doubt that a drastic increase in the number of microservice deployments will make the operations of server virtualization infrastructures more complex. For instance, the classification of microservices depending on their similarities would help simplify the operation of microservices.
4. We prioritize the agility of data analytics to realize the agile resource control mechanism that can respond in a timely way to the rapid fluctuations of CPU utilization (in time granularity of seconds).

Our proposed framework can adopt either learning technique: BL and OML. As we explained in Section 2, the OML is effective in time-series data analytics in the case that the system receives a new dataset one after another. However, the OML technique has a disadvantage of low tolerance to noise data. Besides, the prediction of CPU utilization (at every second or of the peak value) that we focus on in this study needs to utilize past datasets as well as a new dataset so that LASSO/LSM-based multiple regression analysis can be executed and evaluated by a seven-fold cross-validation. Therefore, every time a new dataset is received, our system uses multiple datasets (e.g., 21 latest datasets) including the new dataset for training the model regularly. We adopt this approach, which is a kind of combining of BL and OML into our system.

3.2 Resource control system

The resource control system finds a solution for resource arbitration among CNFs or NF migration between physical servers by using the CPU utilization prediction results received from the advanced data analytics system, and/or other external information such as network and service requirements. Because the main contribution of this paper is an advanced data analytics system, we omit a detailed explanation of the structure and mechanisms of the resource control system and refer readers to [22]–[26].

4. ARCHITECTURE OF ADVANCED DATA ANALYTICS SYSTEM FOR MULTIPLE MICROSERVICES

The two kinds of NFs, VNF and CNF, are different from each other especially in terms of the dependency between NFs. Each VNF usually occupies a VM and is running independent of other VNFs, while multiple CNFs constituting a microservice-based application are closely connected to each other inside a VM. This means that the CPU utilization data of all CNFs in the application needs to be measured, analyzed and predicted simultaneously so that the computational resource control shown Fig. 1 can work properly. This motivates us to design an architecture of a data analytics system which can execute learning and prediction processes of all CNFs constituting an application in parallel.

We propose an architecture for an advanced data analytics system to analyze the many containerized microservices constituting a microservice-based application in parallel, as shown in Fig. 2. As with the architecture in [29], the system consists of (i) the prediction component to predict the CPU utilization of a new dataset; and (ii) the learning component to analyze the past datasets of CPU utilization and tuning parameters of a prediction model. Monitoring data received from server virtualization infrastructures and microservices' name/status information is input to the two components.

The learning component includes the proposed three-stage intelligent model pipelining and its three cascaded ML models: nonlinear regression, microservices clustering, and multiple regression. In each stage of pipelining, the workloads of all microservices can be analyzed in parallel, but individually and independently for each microservice. The model at the first stage analyzes and predicts the trends of fluctuations in CPU utilization, the model at the second stage simplifies the operations of many deployed microservices, and the model at the third stage predicts the peak CPU utilization, and prevents the system from experiencing a shortage of CPU resources due to the autonomous resource control mechanism. The output results of the first and second models are used to determine the second and third models, respectively. Similarly, the output results of the learning component's intelligent model pipelining generate the tuning parameter values of a suitable ML model to be used in the prediction component, as is the case for the architecture discussed in [29].

To predict peak CPU utilization, there may be cases in which the ML model of multiple regression may not be effective for different types of microservices all the time. If it is judged that multiple regression is not necessary in the third stage of pipelining, another option without any ML techniques may be effective for the third stage. Thus, it is possible to use two kinds of models in the third stage of pipelining: a non-ML data analytics model, and a multiple regression model. If a multiple regression analysis is judged to be effective for the cases in which the fluc-

tuations of CPU utilization are relatively large, the output results obtained from the clustering model are forwarded to the multiple regression model; otherwise, they are forwarded to a non-ML data analytics model.

The data analytics system and resource control system are interconnected through a secure connection (e.g., using transport layer security) at all times. The prediction results for the CPU utilization of microservices for a specific period (e.g., every 100 s) are obtained from the prediction component. They are then sent to the resource control system, which replies with an ACK message to the data analytics system. In the next section, we explain each model in detail when describing the learning component.

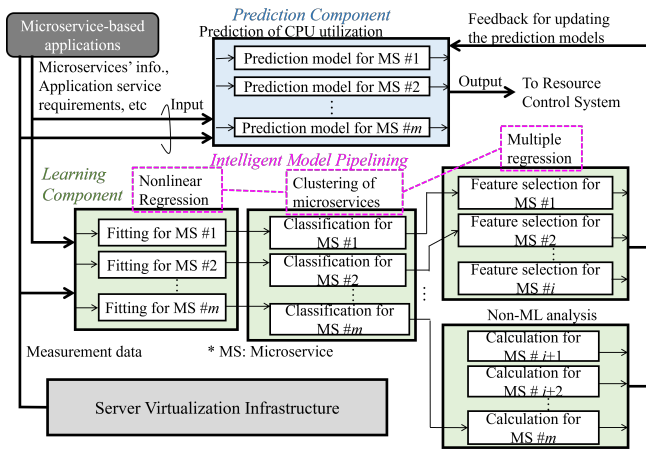


Fig. 2 – Proposed architecture of the advanced data analytics system for microservices

5. THE THREE ML MODELS CASCADED BY THE THREE-STAGE INTELLIGENT MODEL PIPELINING

5.1 First stage: Nonlinear regression model

This model analyzes the trends of the fluctuations in CPU utilization in seconds within a specific period for each microservice. Because the fluctuations of the CPU utilization of several microservices may be too complicated to be expressed by a simple model such as a linear function, a nonlinear regression analysis can be more effective for analyzing the trends of fluctuations in CPU utilization in the first stage of the three-stage intelligent model pipelining [29]. Every time a new dataset is obtained from the monitoring units of containers, the model at the first stage executes a nonlinear regression analysis with the dataset obtained from each container (i.e., microservice). For this model, we propose the application of LS-SVR, as is the case for the two-stage intelligent model pipelining discussed in [29].

5.1.1 LS-SVR

A typical SVR was proposed for the first time in 1996 [37]. It derives a nonlinear regression function by mapping data into a higher-dimensional space by means of a kernel

trick utilizing the characteristics of the kernel function, which achieves high computational efficiency when handling a nonlinear correlation in data. The kernel trick enables the adoption of a nonparametric ML model. The ML model automatically creates a nonlinear function without requiring prior knowledge, such as the nonlinear correlation of data. Therefore, we do not need to select a specific base function by performing manual operations prior to regression analysis. SVR can create a nonlinear function with a relatively high tolerance to noise in data by defining a range to allow differences from real data (i.e., regarding the difference as 0 when it is within a predefined range); thus, it can realize superior nonlinear regression considering both fitting to real data and generalization ability.

LS-SVR was proposed in 2005 [38] as an extension to the typical SVR, and has been applied in many research fields [39]–[41]. Though both typical SVR and LS-SVR have almost the same generalization ability, LS-SVR has several advantages over the typical SVR; such as lower computation cost, flexible selection of kernel functions, simplified maintenance, and lower memory footprint. In particular, a lower computation cost is achieved by defining simple simultaneous linear equations for solving an optimization problem, whereas the typical SVR defines a relatively complicated quadratic programming problem. In this study, we adopt a Radial Basis Function (RBF) kernel [42] for LS-SVR as a representative and the most used kernel method to realize a nonparametric nonlinear regression.

5.1.2 Mathematical model of LS-SVR

Each dataset is expressed as $[(x_1, y_1), \dots, (x_t, y_t), \dots, (x_T, y_T)]$ ($x_t \in R^n$, $y_t \in R$) where T is the maximum number of explanatory variables in a dataset.

The nonlinear regression function of LS-SVR is expressed as:

$$f(x) = \sum_{t=1}^T \alpha_t K(x, x_t) + b_s, \quad (1)$$

where $\alpha_t (\in R)$ is the Lagrange multiplier of the t th explanatory variable, and b_s is a bias term. Both of these are obtained by solving an optimization problem formulated in detail in [38]. $K(x, x_t)$ is the RBF kernel, which is expressed as:

$$K(x, x_t) = \exp(-g\|x - x_t\|^2), \quad (2)$$

where g is a kernel-related parameter that determines the form (degree) of the function expressed as $g = 1 / 2\sigma^2$, and σ^2 is a kernel function parameter. A regularization constant to avoid overfitting, γ , is included in the optimization problem formulated in [37]. In general, we can change the nonlinear regression function by adjusting the values of the two hyperparameters g and γ [37][38]. In this study, we set $\gamma = 100$ statically. As the value of g increases, the nonlinear function becomes more complex (i.e., a higher degree).

5.2 Second stage: Clustering model

In the clustering model, all microservices used to provide a network or cloud application service are classified into several classes according to their patterns of resource usage by means of an unsupervised learning clustering technique. In future server virtualization infrastructures, a large number of microservices will be deployed on containers created on VMs and physical servers, and the trends of fluctuations in their CPU utilization will be diverse. Hence, in the second stage of the intelligent model pipelining, it is effective to classify the microservices based on their similarities in CPU resource usage so that the operations of microservices, including the dynamic selection of learning and prediction models, can be simplified.

When a network or cloud application is deployed for the first time, human operators may need to define the clustering model (i.e., two or more dimensional spaces) based on the data analytics results of the first ML model. In this way, every time a new dataset is obtained, the clustering model can be automatically re-executed. We propose the application of the k -means++ method as a clustering technique.

5.2.1 k -means++ method

The standard k -means method was first used in 1967 [43]. First, it roughly divides the data into several clusters. Subsequently, the clustering results are adjusted by the mean values of clusters several times so that all data can be appropriately classified into several clusters based on the similarities between the data. The number of clusters is expressed as k , which can be set to any number. However, the standard k -means method tends to experience unstable clustering results, which may change depending on the initial parameter values. To address this issue, a k -means++ method was proposed in 2007 [44]. The method executes the initialization so that the locations of multiple cluster centers cannot be close to each other. It suppresses the dependency on the initial values of the cluster centers by means of a weighted probability distribution.

5.2.2 Algorithm of k -means++ method

We denote $D(m)$ to be the shortest distance from a data point to the closest center that has already been chosen.

Step 1: Choose a point from all data points randomly, and regard it as cluster center c_1 .

Step 2: Choose a new cluster center c_m randomly with the following probability:

$$P(m) = \frac{D(m)^2}{\sum_{m=1}^M D(m)^2}, \quad (3)$$

where $m \in \{1, 2, \dots, M\}$. The chosen data point should be one, except for the data points that have already been chosen. M is the total number of data points classified by

the k -means++ method, which corresponds to the number of microservices ($M = 12$ for robotshop, $M = 15$ for sockshop, and $M = 21$ for eshop) in this study.

Step 3: Repeat step 2 until k cluster centers are chosen.

Step 4: Execute the standard k -means algorithm by taking the k cluster centers chosen in the above steps as the initial values.

5.3 Third stage: Multiple regression model

In the multiple regression model, the past data of CPU utilization is analyzed to predict the maximum value of CPU utilization in a future specific time. Even when the fluctuations in CPU utilization tend to be relatively large, the peak CPU utilization of a microservice can be accurately predicted to achieve a reduction in the cases of CPU resource shortages and promote high-efficiency CPU utilization. Therefore, in the third stage of intelligent model pipelining, it is effective to deploy a multiple regression analysis model by appropriately defining the input and output of data; in this study, we define the input as CPU utilization in a specific period, and the output as the maximum value of CPU utilization in a specific future time. We apply the LASSO followed by an LSM, which we call LASSO/LSM, as a multiple regression model.

5.3.1 LASSO

LASSO is a type of regression analysis model used as a sparse modeling technique in machine learning [45]–[48]. We formulate the model utilizing LASSO to predict the maximum value of CPU utilization, especially for microservices where the fluctuations in CPU utilization are relatively large.

Previous studies have shown that there exists a sparsity of time-series data on CPU utilization [24][25][29]. Thus, to predict the peak CPU utilization, only a small number of explanatory variables in the data may be important for predicting the objective variable of data. The explanatory variables are a set of CPU utilization values in a specific period, while the objective variable (i.e., output) is the maximum value of CPU utilization in a future specific time. If the data is sparse, a sparse modeling method such as LASSO is suitable because it is a simple mathematical model and reduces prediction errors by mitigating overfitting.

5.3.2 Mathematical model of LASSO

We define E as the Least Mean Square Error (LMSE) in the prediction of the objective variable (output of multiple regression analysis), which can be formulated as:

$$E = \sum_{n=1}^N (Y_n - y_n)^2 + \lambda \sum_{i=1}^v \|w(i)\|, \quad (4)$$

where v is the explanatory variables (input of multiple regression analysis), Y_n is the actual observed value of the

objective variable, λ is a regularization parameter, and y_n is the predicted value of the objective variable, which is expressed as:

$$y_n = \sum_{i=1}^v w(i)x_n(i) + b_l, \quad (5)$$

where $x_n(i)$ and $w(i)$ are the actual value and weight of the i th explanatory variable in the n th dataset ($n \in (1, 2, \dots, N)$), respectively. b_l is the intercept coefficient, and v is the number of explanatory variables. LASSO sets up the values of $w(i)$, b_l , and λ such that the value of E is minimum. The second term in Eq. 4 is a regularization term. If $\lambda = 0$, this term does not affect the prediction error, and the model is the same as that of LSM. In this way, LASSO strives to avoid overfitting to real data by adding a regularization term to the LSM [45]–[48].

5.3.3 LASSO/LSM

The application of LSM after selecting a small number of explanatory variables using LASSO can further reduce prediction errors [24][29][49]. We need to create a regression model with a smaller number of explanatory variables, only the ones important for prediction, using LASSO. As a result of LASSO, the weights of some explanatory variables are set to non-zero, and the weights of all other explanatory variables are set to zero. We adopted a seven-fold cross-validation to create a generalized regression model, as in our previous studies [24][29]. The seven types of cross-validation results may be different from one another depending on the data. We define important explanatory variables as the variables that are chosen (i.e., whose weights are non-zero) four or more times in the seven-fold cross-validation of LASSO [24][29]. Subsequently, to determine a prediction model, the LSM-based multiple regression is executed using only the important explanatory variables.

5.4 Non-ML analysis model

If the fluctuations in CPU utilization of a microservice are relatively high in the time scale of seconds, the multiple regression analysis may be effective to predict the peak CPU utilization though it depends on the regression results. Otherwise, we may not need to apply any ML technique to predict peak CPU utilization. In other words, for microservices where the CPU utilization rates do not fluctuate from the beginning, we only need to allocate the right amount of CPU resource to each microservice by means of past data. Additionally, when multiple regression does not reduce the prediction errors of peak CPU utilization, an ML technique may not be necessary. In these cases, we may apply a method without any ML technique, which we call non-ML method such as (i) adopting the Average of a set of Maximum Values (AMV), and (ii) adopting the Largest one in a set of Maximum Values (LMV), as is the case for the non-ML methods discussed in [29].

The estimated value of the objective variable (i.e., the predicted value of the maximum CPU utilization) of the n th dataset in AMV, y_n , is expressed as:

$$y_n = \sum_{i=1}^{n-1} \frac{Y_i}{n-1}, \quad (6)$$

where Y_i is the actual observed value of objective variable of i th dataset.

Similarly, the predicted value of the objective variable of the n th dataset in LMV, y_n , is expressed as:

$$y_n = \max(Y_1, Y_2, \dots, Y_{n-1}), \quad (7)$$

where Y_i ($i \in (1, 2, \dots, n-1)$) is the actual observed value of the objective variable of the i th dataset.

5.5 Consideration on determining a prediction model for a new dataset

The learning results obtained from the three-stage intelligent model pipelining led to the selection and configuration parameters of a prediction model for future CPU utilization. Because the learning results include both a nonlinear regression function and an output value of multiple regression analysis, we need to decide how to reflect both results in creating the prediction model for dynamic resource allocation to each CNF (microservice). One approach is as follows: We regard the difference between the output value of multiple regression and the peak value on CPU utilization of the nonlinear regression function as the resource margin. Then, we determine the prediction model as a function where the amount of CPU with the value of margin is added to all values on the predicted nonlinear regression function.

However, in this study, we focus on the learning component of the advanced data analytics system for CNFs constituting each microservice-based application, including the three-stage intelligent model pipelining and each ML model. Hence, creating an optimal prediction model for dynamic resource adjustments based on multiple types of ML analysis results is beyond the scope of this study.

5.6 Possible examples of using our methods

The proposed data analytics system consisting of three-stage intelligent model pipelining is effective for simple NF operations and the proactive autoscaling of VM/container resources because it yields low prediction errors for computational resource usage. Therefore, it can be installed in server virtualization infrastructures such as Amazon Web Services (AWS) and the Google Cloud Platform (GCP). It can also be integrated with other state-of-the-art NFV operation architectures and tools such as open-source MANO [4].

6. EVALUATIONS OF PROPOSED METHODS

Fig. 3 illustrates the experimental system for the measurement and data analytics of CPU utilization of mi-

crosservices. The experimental system was configured on high-spec server machines in a testbed network system in our laboratory. In the VM/container mixed server virtualization infrastructure, the host OS and guest OS are Ubuntu 18.04LTS. The physical servers have 3.30 GHz 10-core/20-thread CPUs and 64 GB of memory. As shown in Fig. 3, we implemented four VMs using Oracle VM VirtualBox (version 6.1) to install and operate microservice-based applications on VM 1, VM 2, and VM 3, and an NF manager in VM 4. To the guest OS for each VM, we allocated one CPU, 2 GB of memory, and 25 GB of storage. The NF manager (VM 4) can remotely create several Docker containers on VMs 1, 2, and 3 by operating Docker Community Edition (CE) and Docker Compose when running three microservice-based applications, each of which is executed on a VM separated from other applications. The data analytics engine (which is operated in a separate server machine with the specification of 3.10 GHz 2-core/2-thread CPU and 64 GB memory) is directly connected to the server virtualization infrastructure via an SSH connection to obtain the measurement data and analyze to predict the CPU utilization of VM 1, VM 2, and VM 3.

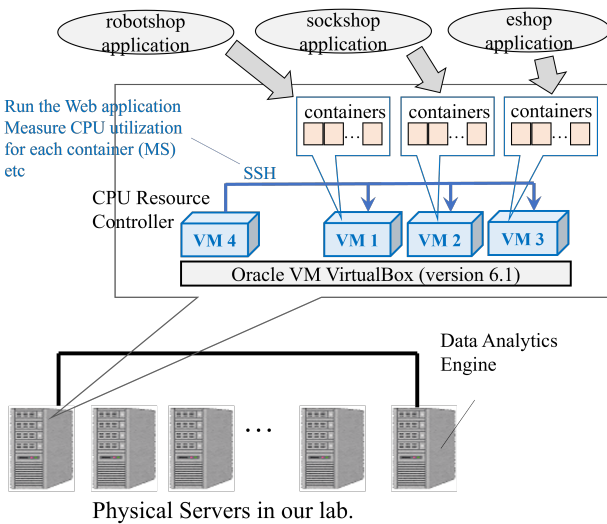


Fig. 3 – Experimental system

To verify our proposed methods, we used three web applications, namely (a) Stan’s robot shop web application (hereafter called robotshop) [50], the (b) Sock shop web application (hereafter called sockshop) [51], and (c) eShop on container web applications (hereafter called eshop) [52], as samples of microservice-based cloud applications. We deployed robotshop on VM 1, sockshop on VM 2, and eshop on VM 3. Applications (a), (b), and (c) consisted of 12, 15, and 21 microservices, respectively. Therefore, we targeted 48 microservices in the measurements and data analytics performed in this study. Application (a) is provided by *robotshop* – *, where * represents the following 12 microservices: *shipping*, *rabbitmq*, *cart*, *payment*, *catalogue*, *user*, *ratings*, *redis*, *web*, *mysql*, *dispatch*, and

mongodb. Application (b) is provided by *sockshop* – *, where * represents the following 15 microservices: *carts*, *queuemaster*, *orders*, *shipping*, *rabbitmq*, *cataloguedb*, *frontend*, *userdb*, *usersim*, *cartsdb*, *catalogue*, *edgerouter*, *ordersdb*, *payment*, and *user*. Application (c) is provided by *eshop* – *, where * represents the following 21 microservices: *rabbitmq*, *sqldata*, *catalogueapi*, *identityapi*, *orderingapi*, *seq*, *basketapi*, *webstatus*, *orderingbackgroundtasks*, *webhooksapi*, *mobileshoppingagg*, *webspa*, *orderingsignalrhub*, *webshoppingagg*, *webmvc*, *paymentapi*, *basketdata*, *mobileshoppingapi*, *nosqldata*, *webhooksclient*, and *webshoppingapi*.

We monitored CPU utilization over a period of 100 s from the start time of the three web applications. The data of CPU utilization is not simulated data, but real data output by running each microservice. In the monitoring, to prepare the dataset, we measured CPU utilizations of each of the 48 microservices every second over a period of 100 s. This means that, in our dataset, the maximum value of v was 100. This is because the period of 100 s from the start time of application shows much larger fluctuation in CPU utilization compared to the later period after 100 s. We obtained 21 datasets (i.e., 21 trials) in this work: $N = 21$, each of which had data of CPU utilizations in the period of 100 s.

Fig. 4 show examples of CPU utilization data used as the dataset for the first trial ($n = 1$) in each of three applications. The CPU utilization rate of each microservice is expressed as {In-use amount of CPU} divided by {Allocated amount of CPU for the microservice}. Similarly, the amount of CPU resource allocated to a microservice is expressed as {Total amount of CPU allocated for each application} divided by {Number of containers (microservices)}. In this study, we set the {Total amount of CPU allocated for each application} as the {Total amount of CPU of the physical server} $\times 0.3$. From the figure of robotshop data, we can observe that the CPU utilization of the two microservices (i.e., *shipping* and *rabbitmq*) are relatively high. Meanwhile, several microservices experience the relatively high CPU utilization in some time, and other microservices do not really experience the high CPU utilization for the entire 100 s. Thus, the trends of the fluctuations in CPU utilization are different between microservices, but we could classify the trends of fluctuations in CPU utilization into several classes depending on their similarities. Similarly, in other applications (i.e., sockshop and eshop), we can observe that the trends of the fluctuations in CPU utilization are different between microservices. It should be noted that the measured data of fluctuation in CPU utilization of each microservice did not show drastic changes in different trials. However, even if the data does show drastic changes in different trials, our proposed method utilizing ML techniques is effective at reducing the prediction errors of CPU utilization. This is because the performances of existing reactive approaches are affected by abnormal noise data more than methods utilizing ML techniques such as LS-SVR.

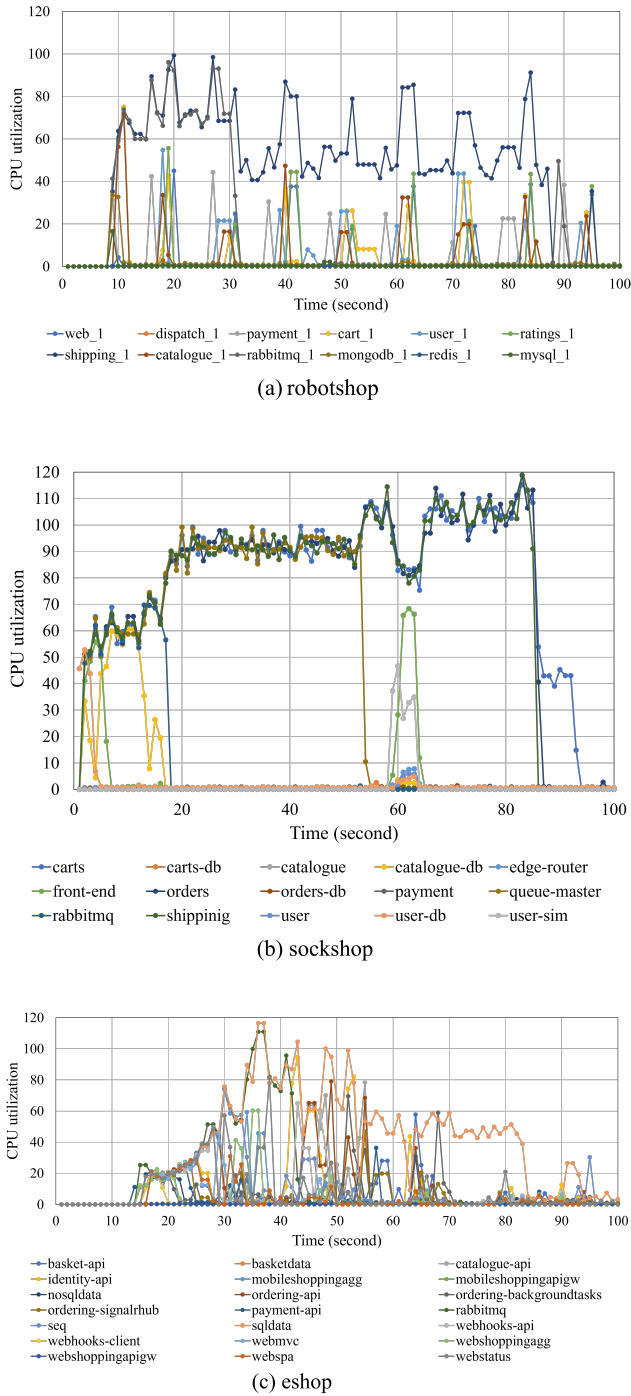


Fig. 4 – Examples of real data on CPU utilization

It should also be noted that our proposed methods are also effective at improving the agility of data analytics and reducing the prediction errors of CPU utilization for a larger scale of infrastructure (e.g., larger number of VMs and physical servers). This is because data analytics are executed individually and independently for each application (i.e., each VM) in each physical server, and the trends of fluctuations in the CPU utilization of each microservice (i.e., each container) do not change unless CPU saturation occurs. Similarly, the NF manager and three applications can also be deployed on separate physical servers

based on their independence, but in this experiment, we deployed them on the same physical server because the advantages of the proposed methods (i.e., agility of data analytics and lower prediction errors) would not change, as discussed above.

6.1 Nonlinear regression analysis

We implemented our proposed nonlinear regression analysis using LS-SVR using MATLAB (version R2021a). In all evaluations, we selected two microservices for each of the three applications as representative services to present the corresponding analysis and evaluation results. These microservices tend to exhibit higher CPU utilization than the other microservices. Specifically, we selected *robotshop* – *shipping* and *robotshop* – *rabbitmq* for robotshop, *sockshop* – *carts* and *sockshop* – *queuemaster* for sockshop, and *eshop* – *rabbitmq* and *eshop* – *sqldata* for eshop.

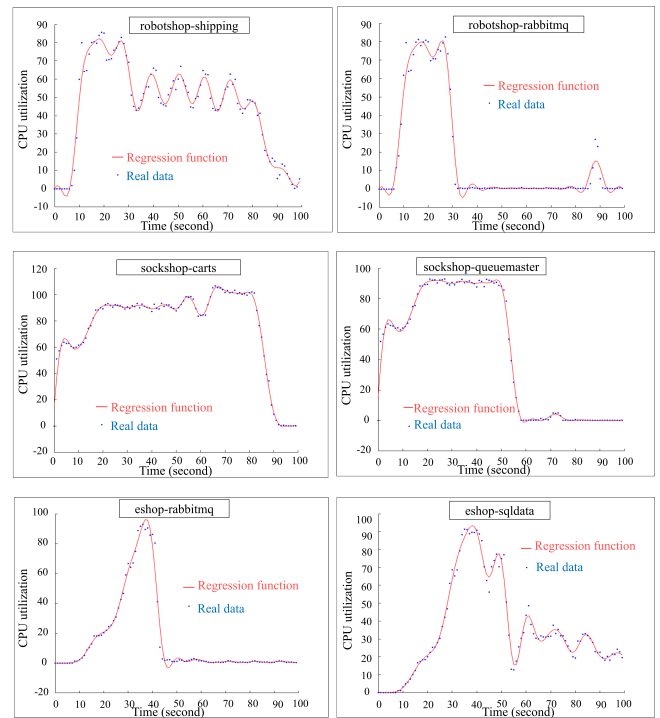


Fig. 5 – Example nonlinear regression results

Fig. 5 show examples of nonlinear regression results obtained from the CPU utilization datasets of six microservices in the three applications mentioned in the above. In each graph, the blue dots represent the real data, while the red line is the nonlinear regression function. Generally, a nonlinear regression analysis can be executed on one dataset, and thus, the real data in each graph in Figures 7-12 are the average of 20 datasets (from the 1st to 20th trial). We set $g = 0.02$ as the RBF kernel parameter. We can observe that LS-SVR automatically derives a nonlinear regression function for each microservice, which we can visually and easily understand the trend of fluctuations in CPU utilization over 100 s.

In addition, each function seems to have a relatively low degree of function which is expected not to cause overfitting on new datasets. In addition, we confirm that LS-SVR can obtain a nonlinear regression function of approximately 2 s per dataset, which contributes to enhancing the agility of CPU utilization analysis.

To evaluate the prediction accuracy of methods, there are several approaches such as MSE and Mean Absolute Percentage Error (MAPE). In this study, we use MSE because the penalty (i.e., degradation degree of evaluation value) increases exponentially as the prediction errors become larger, and thus, it enables us to see the degradation of performances more clearly compared to MAPE. Besides, MAPE has two disadvantages: (1) When the correct answer value is 0, it cannot execute a division process and the calculation error occurs. (2) When the correct answer value is less than 1 and very close to 0, the evaluation value tends to become very large.

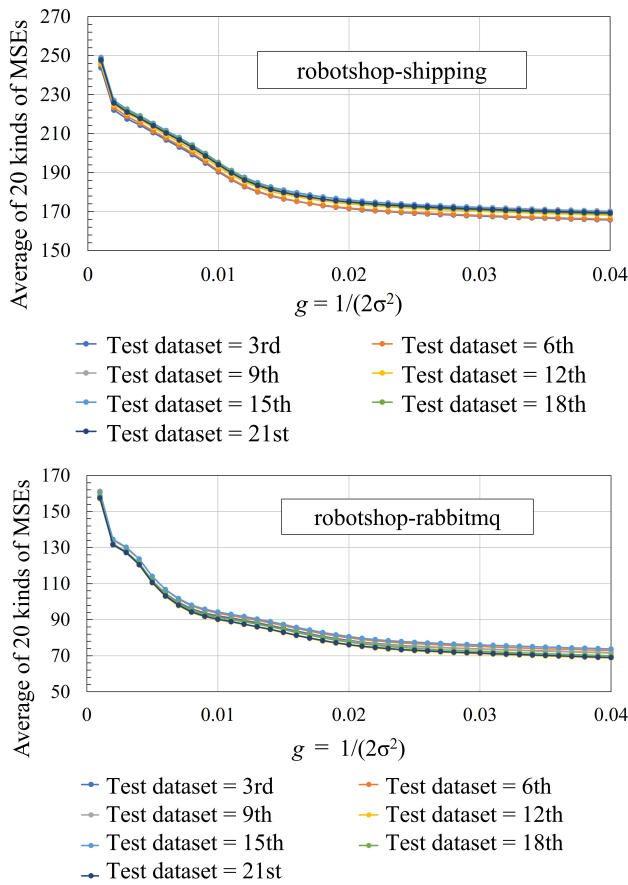


Fig. 6 – Average MSEs versus value of g ranging from 0.001 to 0.04 for the two microservices in the learning phase (robotshop)

Figures 6-8 show the average MSEs versus the value of g ranging from 0.001 to 0.04 for (a) *robotshop – shipping* and *robotshop – rabbitmq*, (b) *sockshop – carts* and *sockshop – queuemaster*, and (c) *eshop – rabbitmq* and *eshop – sqldata* in the learning phase. We set the value of γ to 100 as the regularization constant. We evaluated the proposed method by applying LS-SVR by means of

leave-one-out cross-validation, and selected 20 training datasets for the learning phase and one test dataset for the prediction phase. In the learning phase, we performed seven cases of leave-one-out validation in which the test datasets were 3rd, 6th, 9th, 12th, 15th, 18th, and 21st, respectively. It is expected that similar trends of MSE curves are also obtained in the other cases of leave-one-out validation because the rough trends of the fluctuations in CPU utilization are similar to each other.

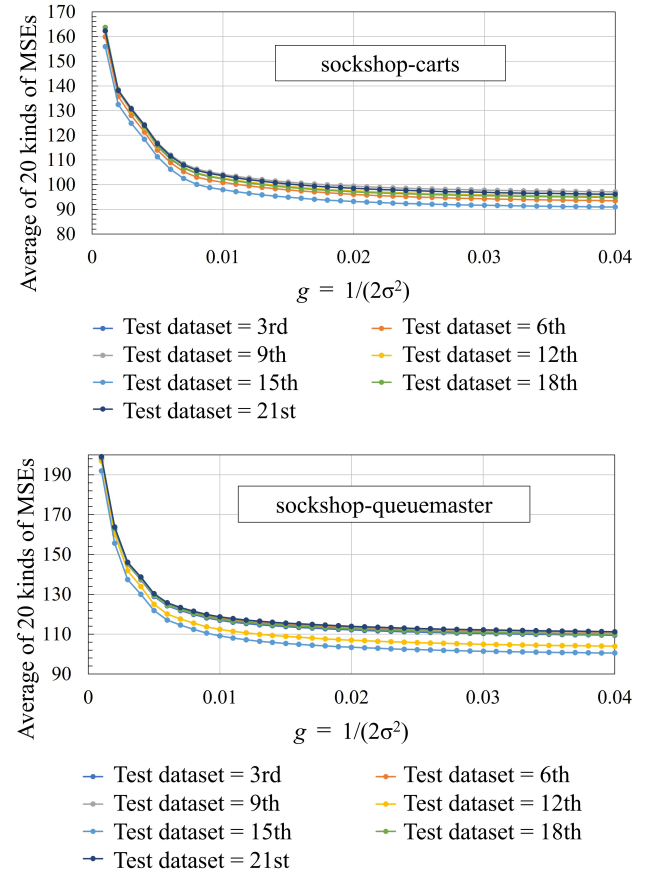


Fig. 7 – Average MSEs versus value of g ranging from 0.001 to 0.04 for the two microservices in the learning phase (sockshop)

We obtained a nonlinear regression function derived from the average of 20 datasets. We derived the MSE of each real data of 20 datasets and the nonlinear regression function, and derived the average of 20 kinds of MSEs. Generally, as the value of g increases, the degree of the nonlinear regression function increases, and the function fits better with the real data. Therefore, when the value of g is in a very small range (e.g., 0.001–0.01), the MSE increases because of the low degree of the regression function. Contrastingly, larger values of g result in smaller MSEs, but the MSEs converge to almost the same value even if the value of g is increased. This is because the fitting degree for the average of 20 training datasets, and the difference from one test dataset do not really change in the range in which the value of g exceeds a certain value (e.g., more than 0.02). It is ideal for the degree of a nonlinear regression function to be as low as possible because a higher

degree may cause a lower generalization ability for future new datasets, and it may make it more difficult to understand the trends of CPU utilization fluctuations (i.e., non-linear correlation of data) from the perspective of operations in server virtualization infrastructure. Therefore, we required the selection of an optimal value of g to maintain the balance between the low degree of a nonlinear regression function and MSEs. In fact, we confirmed that when the value of g exceeds 0.02, the degree of the regression function becomes too high to understand the trends of fluctuations in CPU utilization. Thus, considering both MSEs and the degree of regression function, we selected $g = 0.02$, for the six microservices highlighted in this study.

It can also be observed that the MSEs of *robotshop – shipping* microservice are larger than those of *robotshop – rabbitmq* microservice. This is because the variation in CPU utilization in a dataset of *shipping* is much larger than that of *rabbitmq*; and thus, the difference between the predicted and real values becomes larger. Similar discussions can be applied for different microservices in different applications.

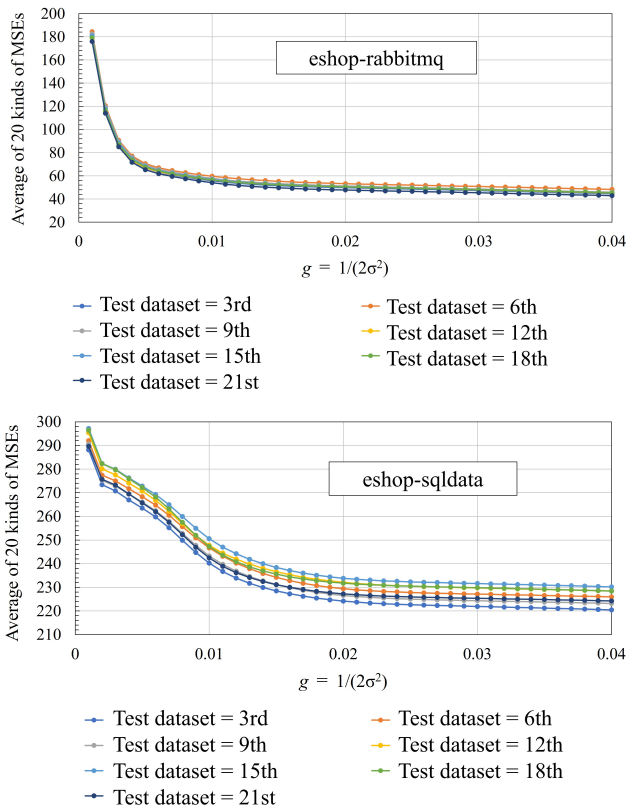


Fig. 8 – Average MSEs versus value of g ranging from 0.001 to 0.04 for the two microservices in the learning phase (eshop)

Consequently, we must maintain a balance between CPU shortage and efficiency of CPU utilization by further upgrading the data analytics system to reflect these analysis results in autonomous resource arbitration between microservices. Figures 9-11 show the MSE versus the trial number (n) of the test dataset in leave-one-out cross-validation (all 21 types) for the six microservices in the

prediction phase. We set $g = 0.02$ and $\gamma = 100$. Because the CPU utilization in each second changes between different datasets, the values of MSEs vary depending on the selection of the test dataset. For *robotshop – shipping*, *robotshop – rabbitmq*, *sockshop – carts*, *sockshop – queuemaster*, *eshop – rabbitmq*, and *eshop – sqldata*, the average prediction errors are approximately 13.5%, 9.0%, 10.1%, 10.8%, 8.1%, and 16.1%, respectively. We required a method to reduce the prediction error as much as possible.

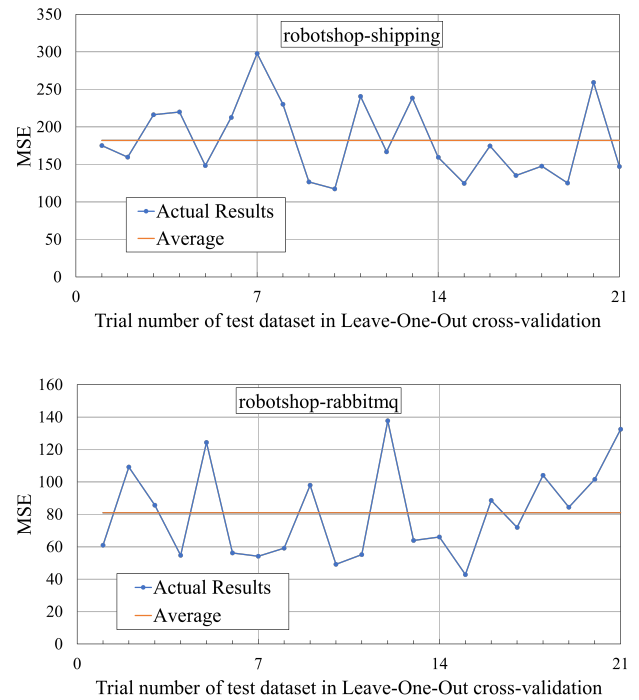


Fig. 9 – MSE versus trial number (n) for the test dataset in leave-one-out cross-validation (all 21 types) for the six microservices in the prediction phase (robotshop)

6.2 Clustering

We implemented our proposed clustering function using the k -means++ algorithm by means of scikit-learn (version 0.19.1). We executed a two-dimensional clustering, where the vertical axis was the average of z measurement trials (i.e., from the first to the z th trial) of maximum CPU utilization in the period from $t = 1$ to $t = 100$, while the horizontal axis was the average of z measurement trials of mean CPU utilization in the period from $t = 1$ to $t = 100$. As the setting of vertical and horizontal axes would depend on the trends of fluctuations in CPU utilization of microservices (i.e., analysis results obtained from nonlinear regression), the above setting is suitable for classifying the microservices of robotshop application in our work.

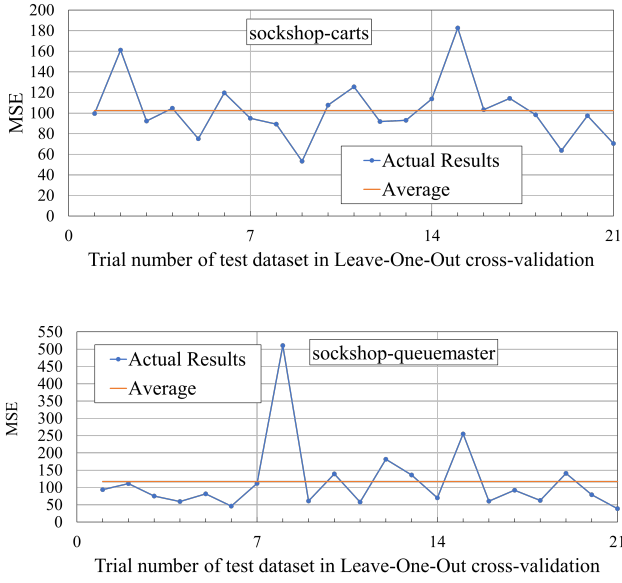


Fig. 10 – MSE versus trial number (n) for the test dataset in leave-one-out cross-validation (all 21 types) for the six microservices in the prediction phase (sockshop)

Fig. 12 shows an example of a captured image of the clustering result for each application where $z = 21$ (i.e., using the average CPU utilization of 21 datasets from the 1st to 21st trial). We set the value of k to 3.

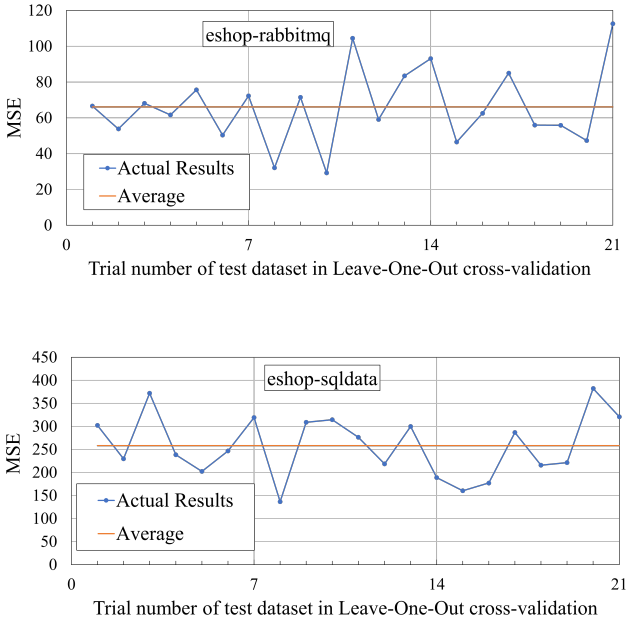


Fig. 11 – MSE versus trial number (n) for the test dataset in leave-one-out cross-validation (all 21 types) for the six microservices in the prediction phase (eshop)

Table 1 presents the analysis results for each of the three applications, where $z = 21$. The 48 microservices are classified into three classes: Class 1 (high CPU utilization), Class 2 (moderate CPU utilization), and Class 3 (low CPU utilization). For the robotshop and sockshop appli-

cations, the clustering results do not change significantly from the first to the 21st trial. For the eshop application, the clustering results fluctuate depending on the value of z . Fig. 13 presents the number of microservices versus z for the eshop application. One can see that when the value of z is less than 11, the clustering results are unstable. This is because the number of datasets used for deriving the average was too small to stabilize in the eshop application. When the value of z is greater than or equal to 11, the clustering results successfully converge.

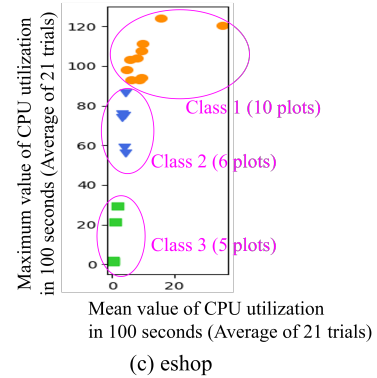
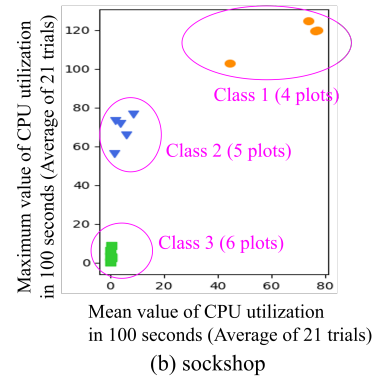
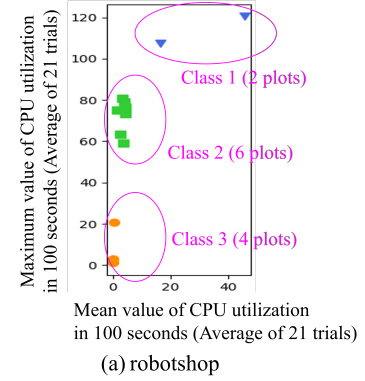


Fig. 12 – Example captured image of clustering, where $z = 21$

We also confirmed that it takes only approximately 16 ms to execute clustering and obtain results for every application. Here, we discuss the management of each class of microservices. Because the trends of fluctuations in CPU utilization in Class 1 are large, the multiple regression model should be selected at the third stage of intelligent model pipelining. For Class 2, the multiple regres-

sion may or may not be selected at the third stage as it depends on the trend of fluctuations in CPU utilization in each microservice of this class. If the multiple regression is judged to be effective for a microservice as a result of multiple regression analysis, the ML model is applied at the third stage; otherwise, the non-ML analysis model is applied. Meanwhile, because Class 3 do not really experience large fluctuations in CPU utilization, a non-ML analysis model is more suitable at the third stage, and therefore, AMV or LMV discussed in Section 5.4 are sufficient to allocate the right amount of CPU resource at the beginning of the deployment. The management function can include other tasks, such as determining a margin value of additional resources to be allocated to the microservices of each class and deciding a policy of container migration from one to another physical server for each class.

Table 1 – Clustering results for the 48 microservices.

Class	Microservices (robotshop)
Class 1	shipping, rabbitmq
Class 2	payment, cart, user, web, ratings, catalogue
Class 3	dispatch, mysql, mongodb, redis

Class	Microservices (sockshop)
Class 1	carts, queuemaster, orders, shipping
Class 2	rabbitmq, cataloguedb, frontend, userdb, usersim
Class 3	cartsdb, catalogue, edgerouter, ordersdb, payment, user

Class	Microservices (eshop)
Class 1	rabbitmq, sqldata, catalogapi, identityapi, orderingapi, seq, basketapi, webstatus, orderingbackgroundtasks, webhooksapi
Class 2	mobileshoppingagg, webspa, orderingsignalrhub, webshoppingagg, webmvc, paymentapi
Class 3	basketdata, mobileshoppingapigw, nosqldata, webhooksclient, webshoppingapigw

In the eshop application, when the value of z is less than 10 (i.e., in unstable scenarios), a dynamic update of the class for each microservice is necessary because fixed resource allocation for Class 2's microservices causes CPU over-allocation if the CPU utilization fluctuates.

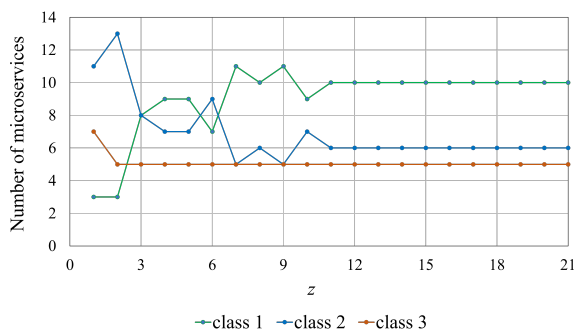


Fig. 13 – Number of microservices versus z in the eshop application

6.3 Multiple regression analysis

As we explained in Section 3, our proposed framework and architecture can adopt the OML as well as the BL for the training and prediction. However, in this evaluation, we only compared our proposed solution using LASSO/LSM to other approaches without ML techniques by means of the fixed 21 datasets in each of 48 microservices.

In all evaluations, similar to the nonlinear regression analysis in the first stage of intelligent model pipelining, we selected two microservices for each of the three applications as representative services to present the corresponding analysis and evaluation results. These microservices tend to exhibit higher CPU utilization than the other microservices. We selected the same six microservices used in the nonlinear regression analysis.

We implemented LASSO and LASSO/LSM using MATLAB (version R2021a). The number of explanatory variables corresponded to the range of time (seconds) for input to the multiple regression model, which is defined as v and used in Eq. 4. The objective variable was the maximum value of CPU utilization in a specific period. The explanatory and objective variables should be appropriately set for each microservice because the trends in the fluctuations of CPU utilization are unique for each microservice. Table 2 presents the targeted range of time for the explanatory and objective variables for each of the six microservices.

Table 2 – Explanatory and objective variables for six microservices

Application	Explanatory variables
robotshop-shipping	$v=20$; ranged from $t=1$ to $t=20$
robotshop-rabbitmq	$v=15$; ranged from $t=1$ to $t=15$
sockshop-carts	$v=20$; ranged from $t=1$ to $t=20$
sockshop-queuemaster	$v=20$; ranged from $t=1$ to $t=20$
eshop-rabbitmq	$v=25$; ranged from $t=1$ to $t=25$
eshop-sqldata	$v=30$; ranged from $t=1$ to $t=30$

Application	Objective variables
robotshop-shipping	In the range from $t=30$ to $t=100$
robotshop-rabbitmq	In the range from $t=20$ to $t=100$
sockshop-carts	In the range from $t=30$ to $t=100$
sockshop-queuemaster	In the range from $t=30$ to $t=100$
eshop-rabbitmq	In the range from $t=30$ to $t=100$
eshop-sqldata	In the range from $t=35$ to $t=100$

We executed seven-fold cross-validation [24] using 21 datasets to evaluate our proposed methods, where 3 test datasets and 18 training datasets were selected for each cross-validation. We compared the MSEs of five methods: AMV, LMV, LSM, LASSO, and LASSO/LSM. We confirmed that it took less than 1 s to execute LASSO by the seven-fold cross-validation using 21 datasets and to obtain the result for each microservice. Table 3 lists the important explanatory variables selected by LASSO for the six microservices.

Table 3 – Explanatory variables selected by LASSO

Application	Selected explanatory variables
robotshop-shipping	$t = 17$
robotshop-rabbitmq	$t = 11$
sockshop-carts	$t = 1, 4, 5, 6, 8, 9, 11, 13, 14, 15, 16, 17, 19, 20$
sockshop-queuemaster	$t = 1, 2, 16, 20$
eshop-rabbitmq	$t = 9, 12, 14, 16, 20, 21, 22, 23, 24$
eshop-sqldata	$t = 11, 14, 16, 17, 18, 20, 21, 22, 23, 25, 26, 27, 30$

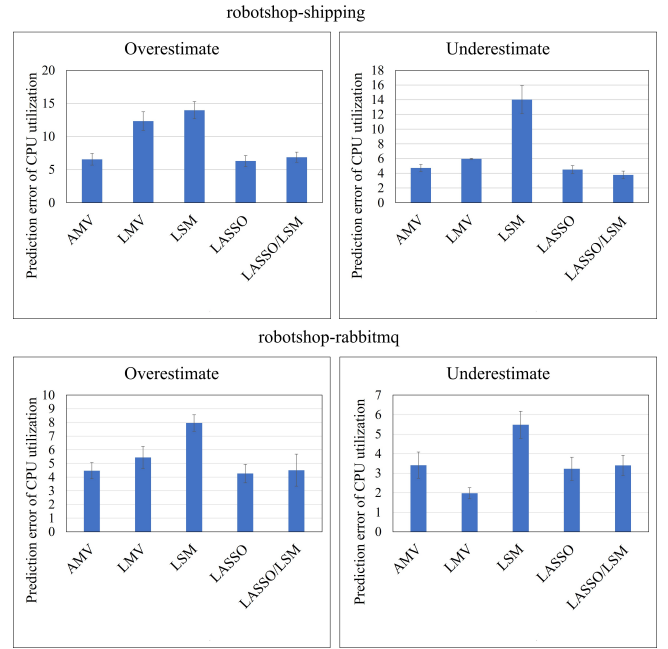
Table 4 lists the MSEs for five methods for the six microservices. According to the seven-fold cross-validation results, the LASSO/LSM model achieved the best performance. This can be attributed to the benefits of LASSO, which suppresses overfitting for real data by simplifying the regression function using a regularization parameter (i.e., retaining only important explanatory variables). Furthermore, the benefits of LSM are enhanced following LASSO-based analysis. In Table 4, one can see that the LASSO/LSM model can reduce the MSEs for the *robotshop – shipping*, *robotshop – rabbitmq*, *sockshop – carts*, *sockshop – queuemaster*, *eshop – rabbitmq*, and *eshop – sqldata* microservices by approximately 14.2% to 86.8%, 20.0% to 80.5%, 80.3% to 91.7%, 20.7% to 90.5%, 49.0% to 92.3%, and 10.8% to 71.9% compared to the other models, respectively.

Table 4 – MSEs of the five methods

MS name	AMV	LMV
robotshop-shipping	51.16416998	173.1909381
robotshop-rabbitmq	34.29609048	57.87053099
sockshop-carts	24.10110476	8.692118778
sockshop-queuemaster	17.24124286	32.08647962
eshop-rabbitmq	327.0167429	474.5988114
eshop-sqldata	51.63883833	163.6811619

MS name	LSM	LASSO
robotshop-shipping	302.8667762	48.09344582
robotshop-rabbitmq	61.29289774	29.84653037
sockshop-carts	11.44237783	10.18134343
sockshop-queuemaster	32.43966569	3.875858928
eshop-rabbitmq	477.6528338	72.15911429
eshop-sqldata	60.65661842	66.82374612

MS name	LASSO+LSM
robotshop-shipping	43.92034519
robotshop-rabbitmq	27.44325738
sockshop-carts	2.000746237
sockshop-queuemaster	3.073510474
eshop-rabbitmq	36.7714837
eshop-sqldata	46.07502043

**Fig. 14** – Amounts of overestimation and underestimation of peak CPU utilization for the two microservices (robotshop)

Figures 14-16 compare the amounts of overestimation and underestimation of the peak CPU utilizations among the five methods for the six microservices. The evaluation results are different for different microservices. For the *robotshop – shipping*, *sockshop – carts*, and *eshop – rabbitmq* microservices, one can see that the LASSO/LSM model can reduce CPU shortages while mitigating CPU resource waste. Additionally, in the *robotshop – rabbitmq* microservice, LASSO achieves the lowest overestimation, whereas LMV achieves the lowest underestimation, although the performance of LASSO is almost the same as that of LASSO/LSM. For the *sockshop – queuemaster* microservice, the LASSO/LSM model achieves the lowest overestimation while LMV achieves the lowest underestimation, but LMV exhibits large overestimation. For the *eshop – sqldata* microservice, the AMV model achieves the lowest overestimation while the LASSO/LSM model achieves the lowest underestimation. As shown in Table 4, the LASSO/LSM model achieves the lowest MSE values for all six microservices. Therefore, when judging comprehensively, the LASSO/LSM model is the best solution for predicting peak CPU utilization.

6.4 Comparison to a threshold-based estimation method

We compared our proposed solution using LASSO/LSM to a threshold-based approach to estimate peak CPU utilization. Generally, a threshold-based estimation method does not utilize any ML techniques and uses a threshold to determine increases or decreases in computational resources, but the policy for setting the threshold can be flexibly defined depending on the scenario and/or objective of data analytics and resource control [5]–[10].

CPU utilization changes with the amount of CPU resource allocated to a microservice. Therefore, in this evaluation, we assumed that the allocated amount of CPU resource was static and that both the threshold-based method and our proposed method estimated the peak CPU utilization without dynamic CPU allocation.

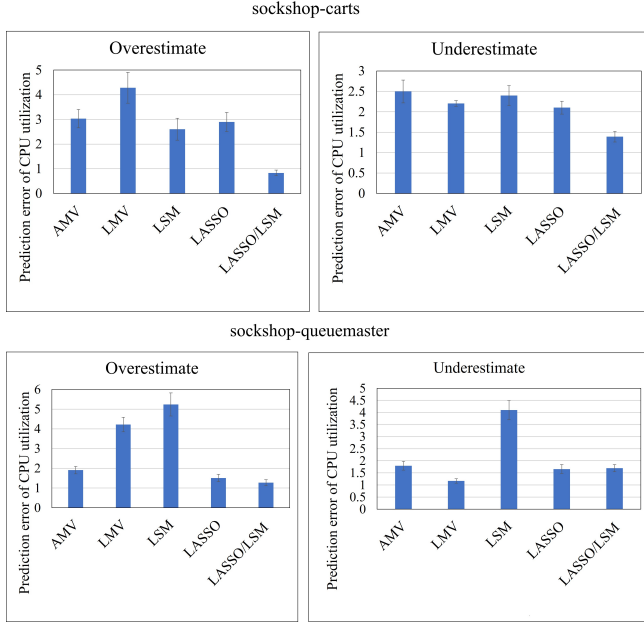


Fig. 15 – Amounts of overestimation and underestimation of peak CPU utilization for the two microservices (sockshop)

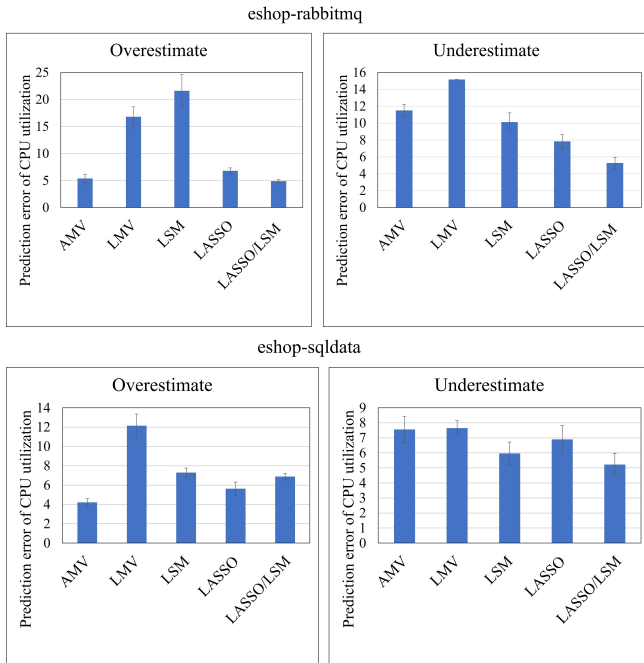


Fig. 16 – Amounts of overestimation and underestimation of peak CPU utilization for the two microservices (eshop)

The threshold-based method estimates CPU utilization in a period of $t + 1$ ($A_n(t+1)$) using the following equation:

$$A_n(t+1) = \theta(x_n(t)) + c, \quad (8)$$

where $x_n(t)$ is the actual CPU utilization at time t in the n th dataset, $\theta(x_n(t))$ is a threshold rounding up the first position of the value of $x_n(t)$, and c is a margin value that is added to the threshold to avoid CPU shortages. In this evaluation, we set two values of c of 10 and 25. The estimation error in the peak CPU utilization in every trial was evaluated. For simplicity, we only targeted the *robotshop—shipping* microservice because it was expected that we could obtain a similar effect for other microservices.

Fig. 17 presents the average overestimation of CPU utilization versus the trial number (ranging from 1 to 21). The negative values in the vertical scale indicate underestimation of CPU utilization. One can clearly see that in comparison to the threshold method with $c = 10$, the proposed solution using the LASSO/LSM model can suppress the underestimation of CPU utilization because it reduces the MSE. Additionally, the average MSE in the proposed method is very close to zero. The threshold method with $c = 25$ can mitigate the underestimation of CPU utilization by sacrificing performance in terms of overestimation. The average MSE is also very close to zero, similar to the proposed method. However, one can see that the variance of MSE (when setting the average to zero) in the proposed method is much smaller than that in the threshold method with $m = 25$. This can be attributed to the reduction of MSE in the proposed method utilizing LASSO/LSM. In summary, our proposed solution can reduce the prediction error of peak CPU utilization compared to the existing threshold-based approach without any ML techniques.

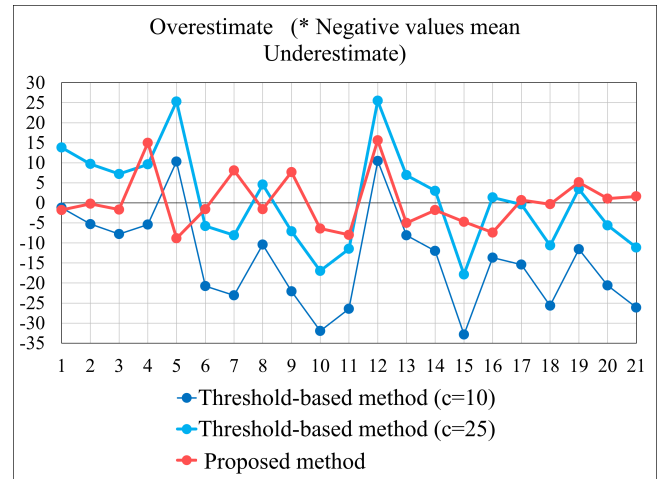


Fig. 17 – Comparison to a threshold-based method on estimating peak CPU utilization

6.5 Discussion of time complexity

In this section, we discuss the time complexity of the proposed intelligent model pipelining with three ML models: nonlinear regression using LS-SVR with an RBF kernel, clustering using k -means++, and multiple regression using LASSO/LSM.

For nonlinear regression, it is well known that the computational (time) complexity of LS-SVR is $O(n^2)$. The time required for regression does not depend on the number of datasets because regression was executed with an average CPU utilization of 100 s for all training datasets. Furthermore, the time complexity does not depend on the number of microservices because regression is executed individually in parallel for each microservice, as shown in Fig. 2. Therefore, n is the number of explanatory variables in each dataset, which was set to 100 in this study. According to the data analytics results, we confirmed that as the number of explanatory variables increases, the computation time increases nonlinearly. For example, if the system must complete nonlinear regression within 1 s and 2 s, the numbers of explanatory variables should be less than 2000 and 4000, respectively.

For clustering, the computational (time) complexity of k -means++ is $O(n \times k)$. Because we set the value of k to three in this study, the complexity is $O(n)$. Clustering depends only on the number of microservices and the n described in $O(n)$ is the number of microservices. Because the number of microservices in every application is very small (i.e., 12, 15, and 21), there is no difference in the time required to obtain clustering results among the three applications. As explained in Section 6.2, the time required for clustering was approximately 16 ms, which is sufficiently short to be ignored in three-stage intelligent model pipelining.

For multiple regression, it is known that the computational (time) complexity of LASSO for extracting important variables is $O(n^3)$ depending on the number of datasets (N) and explanatory variables (v). If the values of both N and v are overly large, the computation time increases significantly. However, in this study, we set $N = 21$ and the value of v in the range of 20 to 30 depending on each microservice. When using such small datasets, the computational time of LASSO is only a few seconds, as discussed in Section 6.3. Additionally, in our proposed LASSO/LSM model, the LSM analysis process following LASSO completed multiple regression in less than 3 s, which is sufficiently fast for agile data analytics.

7. CONCLUSION AND FUTURE WORK

We have introduced a framework of autonomous computational resource control and management for containerized microservices that is compliant with Recommendation ITU-T Y.3177. It consists of server and network virtualization infrastructures, an advanced data analytics system, and a resource control system.

We have proposed an architecture of the advanced data analytics system for microservices, which consists of learning and prediction components to accurately predict CPU utilization in an agile manner in response to fluctuations in resource usage for each microservice.

For the learning component, we proposed a three-stage intelligent model pipelining that cascades the three ML models. The first stage model of the pipelining is for nonlinear regression analysis, which analyzes and predicts

the trend of fluctuations in CPU utilization of each microservice. The second stage model is for the clustering of each application consisting of multiple microservices, which simplifies the operations of microservices, such as a class-based selection of analysis methods with or without ML techniques. The third stage model is for the multiple regression analysis, which predicts the peak CPU utilization of a specific future period in each microservice.

We configured an experimental system for the measurement and data analytics of containerized microservices. We measured the CPU utilization of 48 microservices in three web applications: robotshop, sockshop, and eshop. In the nonlinear regression and multiple regression analyses, we mainly analyzed and evaluated two microservices for each of the three applications, namely *robotshop*—*shipping*, *robotshop*—*rabbitmq*, *sockshop*—*carts*, *sockshop*—*queuemaster*, *eshop*—*rabbitmq*, and *eshop*—*sqldata*. In our clustering analysis, we targeted all 48 microservices in the three applications.

For the nonlinear regression analysis, we propose the application of the LS-SVR with the RBF kernel, and numerically derived the tradeoff between the degree of the regression function and MSEs. We showed that the kernel parameter $g = 0.02$ would be optimal for each microservice—although it also depends on the operation policies of network/cloud service providers.

For clustering, we propose the application of the k -means++ method. We have proven that the clustering results did not change at all in the robotshop and sockshop applications, and converged and became stable when the number of datasets was greater than or equal to 11 in the eshop application.

For the multiple regression analysis, especially for microservices belonging to Class 1 (decided by the clustering), we propose the application of LASSO/LSM. This model can reduce the MSEs for the *robotshop*—*shipping*, *robotshop*—*rabbitmq*, *sockshop*—*carts*, *sockshop*—*queuemaster*, *eshop*—*rabbitmq*, and *eshop*—*sqldata* microservices by approximately 14.2% to 86.8%, 20.0% to 80.5%, 80.3% to 91.7%, 20.7% to 90.5%, 49.0% to 92.3%, and 10.8% to 71.9%, respectively, compared to the other methods. In evaluations of overestimates and underestimates of CPU utilization, we have obtained the results that, judging comprehensively, the LASSO/LSM is the best solution for predicting peak CPU utilization of each microservice in a specific future period.

In future work, we will enhance the advanced data analytics model, which can further reduce the prediction errors, implement the model, and evaluate its performances in a scenario of the OML as well as the BL. We will also investigate methods to create an optimal prediction model based on multiple types of ML analysis results obtained from intelligent model pipelining for sophisticated CPU resource arbitration between microservices. Further, we will implement autonomous interworking between the advanced data analytics system and resource control system, and implement interworking with Kubernetes' horizontal autoscaling. In addition, we will propose and

implement a scheduling method for container migration, data analytics, and verification using other microservice-based applications for networks and clouds utilizing communication network infrastructures (i.e., SDN).

ACKNOWLEDGEMENT

This work was conducted as part of the project titled "Cooperative AI models for Optimal Control of Computational Resources in Multi-Vendor Networks (22K12005)" supported by the Japan Society for the Promotion of Science, Japan. The authors would like to thank Hiroaki Harai, Masahiro Jibiki, Takahiro Hirayama, and Yasunaga Kobari of the National Institute of Information and Communications Technology for their valuable comments on this work.

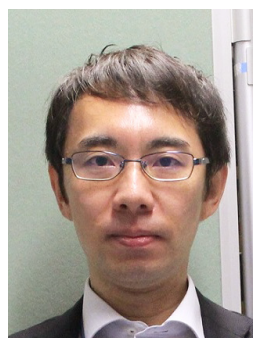
REFERENCES

- [1] 5GMF White Paper: "5G Mobile Communications Systems for 2020 and beyond -Ver.1.1-," Sep. 2017.
- [2] 3GPP White Papers: <https://www.3gpp.org/technologies/presentations-white-papers>.
- [3] 5G-PPP Architecture Working Group White Paper: "View on 5G Architecture -Ver.3-," Feb. 2020.
- [4] ETSI Open Source MANO (OSM): <https://osm.etsi.org/>.
- [5] The Linux Foundation, "Kubernetes": <https://kubernetes.io>.
- [6] A. Boudi, M. Bagaa, P. Poyhonen, T. Taleb, and H. Flinck, "AI-Based Resource Management in Beyond 5G Cloud Native Environment," *IEEE Network*, vol.35, issue 2, pp.128-135, Mar. 2021.
- [7] D. H. Luong, H. T. Thieu, A. Outtagarts, and Y. G. Doudane, "Predictive Autoscaling Orchestration for Cloud-native Telecom Microservices," *Proc. of 2018 IEEE 5G World Forum (5GWF)*, Silicon Valley, CA, USA, 2018.
- [8] A. A. Khaleq and I. Ra, "Agnostic Approach for Microservices Autoscaling in Cloud Applications," *Proc. of IEEE International Conference on Computational Science and Computational Intelligence (CSCI)*, pp.1411-1415, Las Vegas, NV, USA, Dec. 2019.
- [9] J. Wong, A. Kwan, H. A. Jacobsen, and V. Muthusamy, "HyScale: Hybrid and Network Scaling of Dockerized Microservices in Cloud Data Centres," *Proc. of IEEE 39th International Conference on Distributed Computing Systems (ICDCS 2019)*, pp.80-90, Dallas, TX, USA, Dec. 2019.
- [10] H. Dinh-Tuan, F. Beierle, and S. R. Garzon, "MAIA: A Microservices-based Architecture for Industrial Data Analytics," *Proc. of IEEE International Conference on Industrial Cyber-Physical Systems (ICPS 2019)*, pp.23-30, Taipei, Taiwan, Province of China, May 2019.
- [11] A. U. Gias, G. Casale, and M. Woodside, "ATOM: Model-Driven Autoscaling for Microservices," *Proc. of IEEE 39th International Conference on Distributed Computing Systems (ICDCS 2019)*, pp.1994-2004, Dallas, TX, USA, Dec. 2019.
- [12] H. Sami, A. Mourad, H. Otrouk, and J. Bentahar, "Fscaler: Automatic Resource Scaling of Containers in Fog Clusters Using Reinforcement Learning," *Proc. of International Wireless Communications and Mobile Computing (IWCMC2020)*, pp.1824-1829, Limassol, Cyprus, July 2020.
- [13] A. A. Khaleq and I. Ra, "Intelligent Autoscaling of Microservices in the Cloud for Real-Time Applications," *IEEE Access*, vol.9, pp.35464-35476, Feb. 2021.
- [14] A. Shahidinejad, M. Ghobaei-Aran, and M. Masdari, "Resource Provisioning Using Workload Clustering in Cloud Computing Environment: A Hybrid Approach," *Springer Cluster Computing*, 24(5), pp.319-342, Mar. 2021.
- [15] M. Abdullah, W. Iqbal, A. Mahmood, F. Bukhari, and A. Erradi, "Predictive Autoscaling of Microservices Hosted in Fog Microdata Center," *IEEE Systems Journal*, vol.15, no.1, pp.1275-1286, Mar. 2021.
- [16] N. M. Magdelaine and T. Ahmed, "Proactive Autoscaling for Cloud-Native Applications using Machine Learning," *Proc. of IEEE Global Communications Conference (GLOCOM 2020)*, Taipei, Taiwan, Province of China, Dec. 2020.
- [17] M. Abdullah, W. Iqbal, J. L. Berral, J. Polo, and D. Carrera, "Burst-Aware Predictive Autoscaling for Containerized Microservices," *IEEE Transactions on Services Computing*, pp.1-13, May 2020.
- [18] A. Bouridah, I. Fajjari, N. Aitsaadi, and H. Belhadef, "Optimized Scalable SFC Traffic Steering Scheme for Cloud Native based Applications," *Proc. of IEEE 18th Annual Consumer Communications and Networking Conference (CCNC 2021)*, Las Vegas, NV, USA, Jan. 2021.
- [19] S. D. L. Shah, M. A. Gregory, and S. Li, "Cloud-Native Network Slicing Using Software Defined Networking Based Multi-Access Edge Computing: A Survey," *IEEE Access*, vol.9, pp.10903-10924, Jan. 2021.
- [20] P. Loreti, A. Mayer, P. Lungaroni, F. Lombardo, C. Scarpitta, G. Sidoretti, L. Bracciale, M. Ferrari, S. Salsano, A. Abdelsalam, R. Gandhi, and C. Filsfil, "SRv6-PM: A Cloud-Native Architecture for Performance Monitoring of SRv6 Networks," *IEEE Transactions on Network and Service Management*, vol.18, no.1, pp.611-626, Mar. 2021.

- [21] W. Qiang, W. Chunming, Y. Xincheng, and C. Qi-umei, "Intrinsic Security and Self-Adaptive Cooperative Protection Enabling Cloud Native Network Slicing," *IEEE Transactions on Network and Service Management*, vol.18, no.2, pp.1287-1304, Jun. 2021.
- [22] T. Miyazawa, M. Jibiki, V. P. Kafle, and H. Harai, "Autonomic Resource Arbitration and Service-continuable Network Function Migration along Service Function Chains," *Proc. of IEEE/IFIP Network Operations and Management Symposium (NOMS 2018)*, Apr. 2018.
- [23] V. P. Kale, Y. Fukushima, P. Martinez-Julia, T. Miyazawa, and H. Harai, "Adaptive Virtual Network Slices for Diverse IoT Services," *IEEE Communications Standards Magazine*, vol.2, issue 4, pp.33-41, Dec. 2018.
- [24] T. Miyazawa, H. Harai, Y. Yokota, and Y. Naruse, "Sparse Regression Model to Predict a Server Load for Dynamic Adjustments of Server Resources," *Proc. of IEEE 22nd Conference on Innovation in Clouds, Internet and Networks and Workshops (ICIN 2019)*, pp.249-256, Paris, France, Feb. 2019.
- [25] T. Hirayama, T. Miyazawa, M. Jibiki, and V. P. Kafle, "Sparse Regression Model-based Relearning Architecture for Shortening Learning Time in Traffic Prediction," *IEICE Transactions on Information and Systems*, vol.E104-D, no.5, pp.606-616, May 2021.
- [26] T. Hirayama, T. Miyazawa, M. Jibiki, and V. P. Kafle, "Service Function Migration Scheduling based on Encoder-Decoder Recurrent Neural Network," *Proc. of IEEE International Conference on Network Softwarization (NetSoft 2019)*, pp.193-197, Paris, France, June 2019.
- [27] Y. Sup55: Recommendation ITU-T Y.3170-series, "Machine learning in future networks including IMT-2020: use cases," Oct. 2019.
- [28] Recommendation ITU-T Y.3177, "Architectural framework for artificial intelligence-based network automation for resource and fault management in future networks including IMT-2020," Feb. 2021.
- [29] T. Miyazawa, V. P. Kafle, and H. Asaeda, "Data Analytics Using Two-Stage Intelligent Model Pipelining for Virtual Network Functions," *Proc. of IEEE International Conference on Cloud Networking (CloudNet 2021)*, pp. 72-80, Nov. 2021.
- [30] H. Asaeda, A. Ooka, K. Matsuzono, and R. Li, "Cefore: Software Platform Enabling Content-Centric Networking and Beyond," *IEICE Transactions on Communications*, vol. E102-B, no. 9, pp. 1792-1803, Sep. 2019.
- [31] "Cefore", Available at: <https://cefore.net/>.
- [32] J. Konecny, H. B. McMahan, and D. Ramage, "Federated Optimization: Distributed Optimization Beyond the Datacenter," arXiv:1511.03575v1 [cs.LG], Nov. 2015.
- [33] K. M. J. Rahman, F. Ahmed, N. Akhter, M. Hasan, R. Amin, K. E. Aziz, A. K. M. M. Islam, M. S. H. Mukta, and A. K. M. N. Islam, "Challenges, Applications and Design Aspects of Federated Learning: A Survey," *IEEE Access*, vol.9, pp. 124682-124700, Sep. 2021.
- [34] Recommendation ITU-T Y.3179, "Architectural framework for ML model serving in future networks including IMT-2020," Apr. 2021.
- [35] S. C. H. Hoi, D. Sahoo, J. Lu, and P. Zhao, "Online Learning: A Comprehensive Survey," SMU Technical Report 1, arXiv:1802.02871, pp.1-100, 2018.
- [36] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, Y. Singer, "Online Passive Aggressive Algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 551-585, 2006.
- [37] H. Drucker, C. Burges, L. Kaufman, A. Smola, and V. Vapnik, "Support Vector Regression Machines," in *Advances in Neural Information Processing Systems* 9, NIPS 1996, pp.155-161, MIT Press, Dec. 1996.
- [38] H. Wang and D. Hu, "Comparison of SVM and LS-SVM for Regression," *Proc. of International Conference on Neural Networks and Brain (ICNNB)*, Beijing, China, pp.279-283, Oct. 2005.
- [39] R. S. Wibowo, B. AW, K. Anam, M. Abdillah, A. Soeprijanto, O. Penangsang, and N. Yorino, "Control of TCSC and SVC Using Least Square Support Vector Regression (LS-SVR) to Improve Voltage Stability," *Proc. of IEEE International Conference on Information Technology and Electrical Engineering (ICITEE)*, Yogyakarta, Indonesia, Oct. 2013.
- [40] X. Li, L. Yu, L. Tang, and W. Dai, "Coupling Firefly Algorithm and Least Squares Support Vector Regression for Crude Oil Price Forecasting," *Proc. of IEEE 6th International Conference on Business Intelligence and Financial Engineering (BIFE 2013)*, pp.80-83, Hangzhou, China, Nov. 2013.
- [41] R. Andrae, "Least-squares Support Vector Regression for GSP-Phot," *Gaia Data processing and Analysis Consortium (DPAC)*, issue 1, pp.1-13, Feb. 2016.
- [42] J. P. Vert, K. Tsuda, and B. Scholkopf, "A Primer on Kernel Methods," *Kernel Methods in Computational Biology*, pp.1-42, July 2004.
- [43] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," *Proc. of 5th Berkeley Symposium on Mathematical Statistics and Probability*, pp.281-297, Jan. 1967.

- [44] D. Arthur and S. Vassilvitskii, "K-means++: The Advantages of Careful Seeding," *Proc. of the 18th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA 2007)*, pp.1027–1035, New Orleans, LA, USA. Jan. 2007.
- [45] R. Tibshirani, "Regression Shrinkage and Selection via the LASSO," *Journal of the Royal Statistical Society. Series B (Methodological)*, vol.58, issue 1, pp.267–288, 1996.
- [46] T. Hastie, R. Tibshirani, and M. Wainwright, "Statistical Learning with Sparsity: the Lasso and Generalizations," *CRC press*, May 7, 2015.
- [47] Y. Yokota, Y. Igarashi, M. Okada, and Y. Naruse, "Estimation of the Reaction Times in Tasks of Varying Difficulty from the Phase Coherence of the Auditory Steady-State Response Using the Least Absolute Shrinkage and Selection Operator Analysis," *Proc. of International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2015)*, Milano, Italy, pp. 6662–6666, Aug. 2015.
- [48] S. Panicacci, M. Donati, L. Fanucci, I. Bellini, F. profili, and P. Francesconi, "Population Health Management Exploiting Machine Learning Algorithms to Identify High-Risk Patients," *Proc. of IEEE International Symposium on Computer-Based Medical Systems (CBMS 2018)*, Karlstad, Sweden, pp. 298–303, June 2018.
- [49] A. Belloni and V. V. Chernozhukov, "Least Squares After Model Selection in High-Dimensional Sparse Models," *Bernoulli*, vol.19, no. 2, pp.521–547, May 2013.
- [50] Stan's Robot Shop: <https://www.instana.com/blog/stans-robotshop-sample-microservice-application/>
- [51] Sock Shop: <https://microservices-demo.github.io/>
- [52] eShop On Containers: <https://docs.microsoft.com/en-us/dotnet/architecture/cloud-native/introduce-eshoponcontainers-reference-app>

AUTHORS



Takaya Miyazawa received an M.E. degree and a Ph.D. degree in information and computer science from Keio University, Yokohama, Japan, in 2004 and 2006, respectively. From April 2006 to March 2007, he was a visiting researcher at the University of California, Davis, CA, USA. He joined the National Institute of Information and Communications Technology (NICT) as a researcher in 2007, where he is currently a research manager. From April 2019 to July 2020, he worked at the Ministry of Internal Affairs and Communications, Japan as a Deputy Director, and returned to NICT in August 2020. His research interests include the automation of network control, management, and virtualization technologies. He served as Secretary of IEEE Communications Society Asia-Pacific Board from 2016 to 2017. He received the Hiroshi Ando Memorial Young Engineer Award in 2007, the Funai Young Researcher Award in 2010, and Best Paper Award from the ITU Kaleidoscope Academic Conference in 2018. He is a member of the IEEE and the IEICE.



Ved P. Kafle received a B.E. degree in electronics and communications engineering from Punjab Engineering College, Chandigarh, India, an M.S. degree in computer science and engineering from Seoul National University, South Korea, and a Ph.D. degree in informatics from the Graduate University for Advanced Studies, Japan. He joined the National Institute of Information and Communications Technology (NICT), Tokyo, Japan, as a researcher in 2006. He is currently working as a research manager at NICT and concurrently holding a visiting associate professor's position at the University of Electro-Communications, Tokyo, Japan. His recent research interests include 5G and beyond network architectures, Internet of things (IoT), network resource management, network operation automation by AI/machine learning, and standardization. He has been serving as a co-rapporteur of ITU-T Study Group 13 since 2014. He received the ITU Association of Japan's Encouragement Award and Accomplishment Award in 2009 and 2017, respectively. He has also received four Best Paper Awards from the ITU Kaleidoscope Academic Conferences in 2009, 2014, 2018, and 2020. He is a Fellow of ITU-T Study Group 13 and a member of the IEEE and the IEICE.



Yusuke Yokota received an M.E. degree and a Ph.D. degree in electronic and information engineering from the Toyohashi University of Technology, Aichi, Japan, in 2010 and 2013, respectively. He joined the National Institute of Information and Communications Technology (NICT) as a researcher in 2013. His research interests include neural engineering, human interface, and machine learning. He received the Human Interface Society Young Researcher's Award and Presentation Award.



Yasushi Naruse received an M.E. degree and a Ph.D. degree in science from the University of Tokyo, Tokyo, Japan, in 2005 and 2007, respectively. He joined the National Institute of Information and Communications Technology (NICT) as a researcher in 2007, where he is currently the Director of Brain Function Analysis and Imaging Laboratory. His research inter-

ests include neural engineering, neuroergonomics, and machine learning. He received the IEEE Computational Intelligence Society Japan Chapter Young Researcher Award.



Hitoshi Asaeda is a Director of Network Architecture Laboratory, National Institute of Information and Communications Technology (NICT). He received a Ph.D. degree from Keio University. He was previously with IBM Japan, Ltd. and a research engineer specialist at In-

ria, Sophia Antipolis research unit, France. He was a project associate professor at Keio University in 2005–2012. He was a guest editor-in-chief of the special series of IEICE Trans. Commun. in 2016 and a chair of IEICE Technical Committee on ICN from 2017 to 2019. He was a general co-chair of IEEE/ACM IWQoS 2021 and served as a TPC member for premier conferences such as IEEE Infocom and WCNC. He was a program officer for several international projects. He is an advisor of IEEE SIGBDIN and SIGIIE. His research interests include information-centric networking, network coding, high-quality streaming, and distributed computing. He has also been actively working in the IETF standards body. He received the IEICE Communications Society Outstanding Contributions Award in 2019. He is a senior member of the IEEE and the IEICE, and a member of the ACM.