Hindawi
Scientific Programming
Volume 2019, Article ID 5075412, 19 pages
https://doi.org/10.1155/2019/5075412

Research Article

Model-Based Extraction of Knowledge about the Effect of Cloud Application Context on Application Service Cost and Quality of Service

Ivana Stupar[1] **and Darko Huljenić**[1,2]

[1]*ETK Research Unit, Ericsson Nikola Tesla d.d., 10002 Zagreb, Croatia*
[2]*Faculty of Electrical Engineering and Computing, 10000 Zagreb, Croatia*

Correspondence should be addressed to Ivana Stupar; ivana.stupar@ericsson.com

Received 4 June 2019; Accepted 12 August 2019; Published 2 September 2019

Academic Editor: Fabrizio Riguzzi

With the increased usage of cloud computing in production environments, both for scientific workflows and industrial applications, the focus of application providers shifts towards service cost optimisation. One of the ways to achieve minimised service execution cost is to optimise the placement of the service in the resource pool of the cloud data centres. An increasing number of research approaches is focusing on using machine learning algorithms to deal with dynamic cloud workloads by allocating resources to services in an adaptive way. Many of such solutions are intended for cloud infrastructure providers and deal only with specific types of cloud services. In this paper, we present a model-based approach aimed at the providers of applications hosted in the cloud, which is applicable in early phases of the service lifecycle and can be used for any cloud application service. Using several machine learning methods, we create models to predict cloud service cost and response times of two cloud applications. We also explore how to extract knowledge about the effect that the cloud application context has on both service cost and quality of service so that the gained knowledge can be used in the service placement decision process. The experimental results demonstrate the ability of providing relevant information about the impact of cloud application context parameters on service cost and quality of service. The results also indicate the relevance of our approach for applications in preproduction phase since application providers can gain useful insights regarding service placement decision without acquiring extensive training datasets.

1. Introduction

In recent years, a significant number of application service providers migrated their workloads to cloud environments [1], offering their services to customers as software as a service (SaaS) solutions. According to the current business analysis projection [2], it is forecasted that 83% of enterprise workloads will be in the cloud by the year 2020. Migration of services to cloud environments enables reduction of service cost by decreasing capital expenditure. However, often it is not simple to choose the best infrastructure provider or determine the right size of a virtual machine instance where a service or its components will be deployed. Cloud computing infrastructure providers are offering computing resources (CPUs, network, memory, and storage resources) on demand and charge them accordingly using various billing models based on the consumption of resources [3]. A recent survey [4] on the challenges related to cloud computing paradigm adoption identified cost management as one of the most prominent issues reported by mature cloud users. An approach that can reduce resource waste and cloud service cost should determine the optimised service placement in a cloud environment, both in terms of infrastructure provider selection and instance right-sizing. Finding such service placement can be a complicated task for a SaaS provider, due to a large number of cloud infrastructure providers and pricing models on the market [5], as well as the potential complexity of the service. Often it is not intuitive which resources are predominantly consumed by the service and in which amount, depending on the dynamic service load.

In the field of machine learning, statistical techniques are used for extracting knowledge from data without using explicit instructions. Due to the growing awareness of the potential value of collected data in various industry areas and advancement of machine learning frameworks, there is an increase of statistical and machine learning application in multiple domains, including optimisation of cloud service placement. Many of such solutions deal with the optimisation of cloud resource allocation from the perspective of the infrastructure providers. The approaches intended for application providers are usually limited to a specific type of application, or they try to optimise service placement on-the-fly once the application has already been deployed in a production environment.

In this work, we examine the possibility of using machine learning techniques to create models for predicting the quality of service (QoS) and service execution cost and evaluate the parameters that affect them the most, based on the statistical models of observed services. In that way, the parameters with the most significant influence can be identified, and knowledge about them can be used to minimise service execution cost and maintain service quality. Also, service models can predict how the specific infrastructure parameters (e.g., instance size and pricing) will affect service performance and cost. This generalised approach can be used by the application service providers for any application in the predeployment process of identifying the best service placement policy so the high service execution cost and potential vendor lock-in can be avoided.

With a significant number of scientific workflows executing in cloud environments, we find such model-based approach driven by application type significant in the context of scientific computing as it is an interdisciplinary field and covers a wide range of software task types. In this paper, we use two service application types to demonstrate the ways that the knowledge about services and their resource utilisation could be used in the process of optimisation of their placement in the cloud. The experimental results demonstrate that it is feasible to use machine learning models and techniques to detect which parameters affect the service execution cost and quality of service, as well as to predict them. Such knowledge can be used by the providers of applications hosted in cloud environments in the decision process of determining optimised cloud service placement to reduce the service cost.

The remainder of the paper is organised as follows. We summarise related work in the field of cloud cost optimisation in Section 2. Section 3 presents the parameters of the cloud application context, which will be considered as features of the prediction models, followed by a description of the use cases analysed in this paper in Section 4. Section 5 describes the data collection method and the mapping of the model features to the measured data. The analysis of service resource usage characteristics can be found in Section 6. Section 7 contains information about methods we applied in the implementation of cost and QoS models, model accuracy metrics, and feature importance estimation. In Section 8, we analyse implemented models and discuss the results, followed by Section 9 in which we conclude the paper and present the following steps in this research.

2. Related Work

As cloud computing paradigm is getting mature, various cloud environments are increasingly used in production environments, both for scientific workflows and industrial applications. Several market analyses [1, 4] confirm that cloud computing is getting more present in industrial settings but with the limited knowledge about cost optimisation of services deployed in a cloud [4].

A body of research is dealing with cost optimisation in cloud environments, including different strategies for achieving the goal of cost minimisation. Some of the most common approaches include optimisation of load-balancing [6] and service scaling algorithms [7], time scheduling of task invocation and performing [8, 9], and the strategy of optimising service placement for lowering its execution cost [10, 11]. Various approaches to optimisation of cloud cost are using techniques from operations research and in particular game theory [12] and metaheuristics [13, 14] with the focus on genetic and other evolutionary algorithms [8, 15]. More recent solutions are focusing on the application of machine learning algorithms to the problem of cost optimisation [11, 13, 16, 17], especially the approaches that aim at allocating resources dynamically in a cloud environment, thus often using machine learning techniques to realise adaptivity. As an example, Zhang et al. [11] presented an architecture of a resource management system for cloud environments based on deep reinforcement learning. The proposed architecture consisted of an intelligent resource manager that continually monitors resource utilisation and application QoS parameters, combines several optimising algorithms, and maps the application to the resources in the second system component—resource polls. As an optimisation algorithm, the authors propose stacked autoencoder Q-network (SAQN) algorithm and evaluate it against another previously developed SmartYARN optimisation algorithm based on reinforcement learning [17].

In addition to achieving optimisation by using different techniques and strategies, the optimisation approaches differ in terms of workloads they focus on, such as scientific workflows [8, 18, 19], gaming [20], medical and health care [16, 21], education [22, 23], and big data [24, 25] workflows. We aim at providing a general approach based on service resource utilisation suitable for any application type that can be deployed as a SaaS service.

Research approaches in the area of cloud cost optimisation are also divided by the perspective of optimisation. As observed through literature review and several surveys [6, 13, 14, 26–28], many resource scheduling solutions and optimisation algorithms are intended for infrastructure service providers, dealing with the optimisation of resource allocation in data centres [29–32]. Fewer solutions are provided for optimisation of services for SaaS providers, who often need a way to evaluate if the placement of the service in a cloud environment is optimal from the perspective of execution cost and quality of service. An example of service

placement optimisation for application providers is presented in [10]. Although this approach is considering the perspective of an application service provider, the solution is developed for the resource market that offers negotiated prices and is not considering application-specific resource demands that might significantly affect the service cost.

Two studies closest to the approach presented in this paper are [33] and [34]. In [33], neural network and linear regression are used to predict CPU utilisation related to an e-commerce benchmark application using time-series data. A similar approach is presented in [34], where a neural network is used to predict the execution time of observed tasks. However, this approach is focused on a specific task of building code in an online repository. The input variables of the presented model contain repository-specific information such as programming language, and a model is created for each of the considered repositories, which makes this solution highly specific for the chosen use-case. The authors of both works do not model the cost of the cloud service execution in the public cloud infrastructure. They also focus on predicting the resource utilisation and do not examine which parameters affect the quality of service or service cost to the greatest extent. In this work, we use interpretability techniques to identify the predictors that have the most significant impact on the model output prediction, which is often not tackled in the available body of the literature.

Approaches that evaluate service placement strategies before the choice of cloud infrastructure provider was made enable further cost optimisation and avoidance of potential vendor lock-in. Guided by this motivation, in this work, we are aiming for an approach that would allow cloud application providers to decide on the best cloud service placement option based on application load, resource utilisation, and other cloud application context parameters in order to minimise service execution cost and maintain an appropriate QoS level. We describe the parameters of the cloud application context in the following section. To demonstrate our approach, we use examples of two cloud application categories identified in [35] and described in Section 4.

3. Cloud Application Context

As we want to observe the effect of various parameters that impact the execution of the cloud application on service cost and QoS, we consider the parameters of the cloud application context (Figure 1).

To create a holistic view of the cloud application context, we define two *planes*. *Application Plane* consists of properties that are specific for application service itself, regardless of where it is deployed. These properties include application *resource usage profile*, which defines the amount of computing resources (*CPU*, *RAM*, *storage*, and *network*) needed for the execution of the application service, or a specific service task that is being analysed, under a defined load. The load is generated by the application users, and it depends on the *number of concurrent users* that send requests to the cloud application, as well as the actions they are performing while using the application service, often defined by the *user type*. As an example, an e-learning platform might have users who upload course materials to the platform and edit lessons webpages and those who consume the course content by streaming video lessons. In the described case, the actions performed by both user types will require different computing resources. Another property usually specified for the application is the *Service Level Agreement* (SLA), which defines the QoS requirements, commonly through a set of *Service Level Objectives* (SLOs), that will guarantee a level of service quality delivered to the application end users. An example of the SLO that could be a part of the application SLA is the response time of the user request, which typically should not exceed a certain time in a defined percentage of total user requests made towards application service.

Each cloud application service is deployed on the cloud infrastructure. *Deployment Plane* of the cloud application context contains features that are related to the infrastructure where the application will be hosted, i.e., its *deployment environment*. The deployment environment chosen by the application provider defines the *amount and properties* of each computing resource type. As an example, cloud infrastructure provider might offer HDD and SSD storage, a different number of virtual CPU cores, etc. The amount of resources might be offered in predefined instances or in the custom amounts requested by the application provider, who is the customer of cloud infrastructure provider. Each deployment environment has one or more (in case of cloud federations) *infrastructure providers*. The infrastructure provider defines *pricing model* according to which the resources will be charged to the users of the cloud infrastructure. Pricing model usually consists of *prices* of the resources, predefined *resource bundles*, the decision on which resources will be *chargeable resources*, and how their consumption will be *measured*. As an example, infrastructure providers often offer free ingress network traffic to attract application providers and decrease their initial cost of the application service migration or deployment to their environment. Some providers offer limited network traffic for free (e.g., up to 10 GB of ingress traffic might not be charged on a monthly basis). The offers of various infrastructure providers also differ in terms of charging granularity, which makes the cloud infrastructure pricing schemes extremely heterogeneous and complex to evaluate in terms of their effect on the cost of the application, considering what might be a good option based on the application load and resource type utilisation.

In our analysis, we aim to examine the effect of the described cloud application context properties and to define the approach that offers a generalised method of modelling any application service deployable in cloud environments, predicting its execution cost and QoS as well as gaining knowledge of the most significant contributors to the cost and QoS. We also want to examine if the appropriate level of prediction accuracy can be achieved using the selected properties as the model features.

In Section 5, we describe how the proposed cloud application context properties are mapped on the measured data that are going to be used as the training data for implementation of the application service cost and QoS models (Section 7).

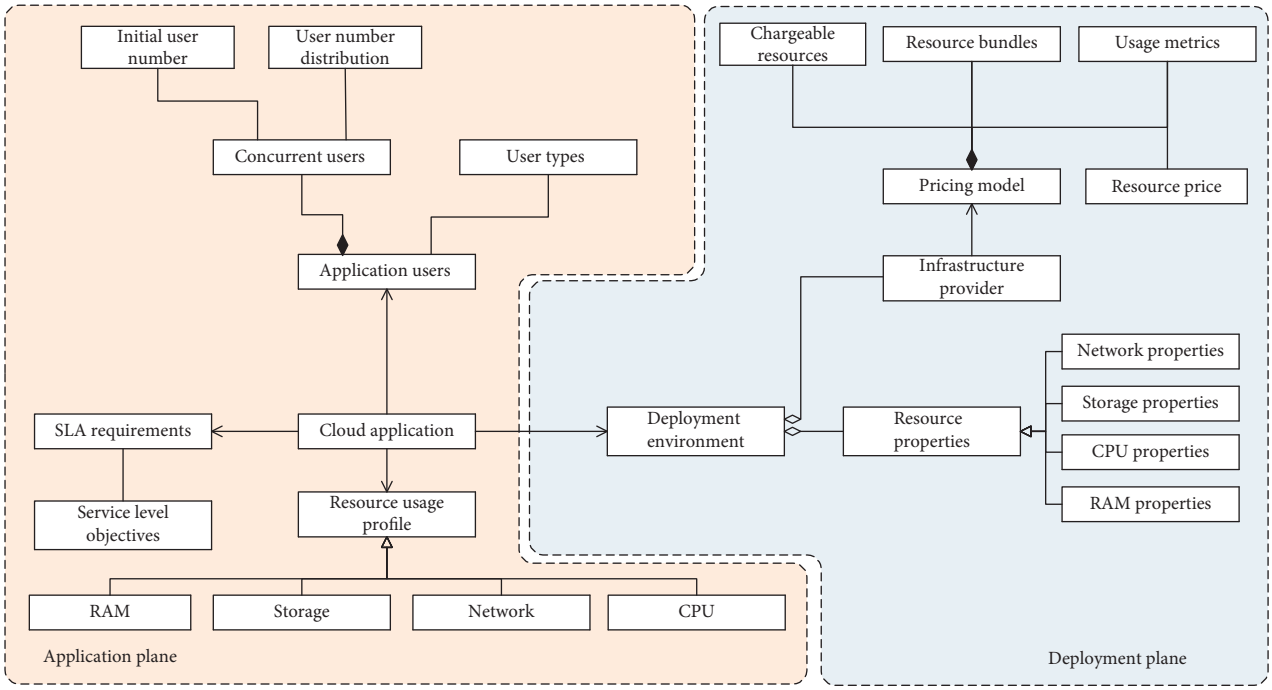

Figure 1: Properties of the cloud application context related to the application (application plane) and cloud infrastructure (deployment plane).

4. Use Cases

As various application types demand a different amount of resources for execution of user requests, in this work, we consider two services of different application types to observe the differences in the resource utilisation during their execution and to examine which parameters will affect the cost or QoS of each service to the highest degree.

As the first cloud application use case, we chose a medical record system (MRS). In our analysis, we observe the scenario of an MRS user performing a query for retrieving a medical record of a specific patient. For the purpose of obtaining the data, we chose an open-source implementation of the MRS system [36]. The MRS service has three main components—user interface, application logic, and a database storing the medical records. The implementation of our measurement scenario consists of a series of three user requests (Figure 2). The first request is used for accessing the user interface of the MRS service, followed by providing user credentials and finally sending the request for retrieving the medical record of a defined patient from the medical record database. For our measurements, user requests were generated using load generator deployed on machines outside of the testbed cloud environment. The response time was measured as the time needed for the end user to perform the described sequence of requests and retrieve the patient's medical record.

As we want to examine the effect of service resource utilisation and infrastructure parameters on the service execution cost and QoS, we define the Service Level Objective (SLO) for the specification of the acceptable application QoS level for the MRS service. We identify an SLO violation as a number of user requests with response time longer than 5 seconds. The number of such requests should not exceed 95% of total user requests.

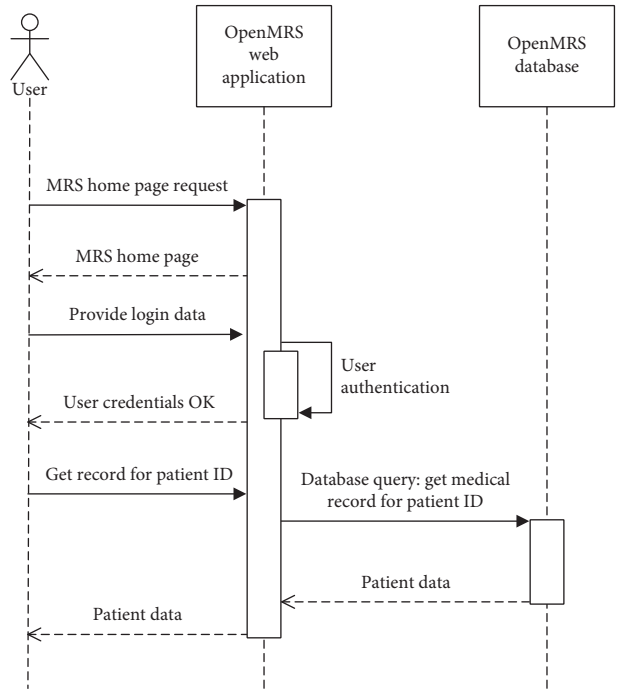

Figure 2: UML sequence diagram of the OpenMRS use-case scenario.

As the second cloud application use case, we chose a video streaming service deployed in the cloud environment. We used Nimble Streamer video streaming server [37] and a scenario of video-on-demand access with users outside of

cloud environment who generate the requests for short video content (Figure 3). Video streaming is implemented using HTTP protocol, transmitting streams of video data which are divided into chunks. The response time in this context is the time needed for transmission of the entire video file to the end user. Considering the deployment of the service in the cloud environment, the video server was deployed as a single component on an Ubuntu virtual machine. The client side consists of the end users outside of the cloud environment who generate the requests for the video content.

The observed QoS metric used for an SLO definition of a video streaming service is, similar to the MRS service, a response time of the user request. For the analysis purposes in the next sections, we observe if there is more than 95% of response time exceeding 10 seconds and define it as an SLO violation.

5. Data Collection

To create a service model based on resource utilisation, we needed to collect the dataset that would allow us to observe the relation between service load, request response time as the chosen QoS metric, and average resources utilisation at the certain number of concurrent user requests, and with a certain amount of resources allocated to a service. For that purpose, we set up a cloud-based measurement environment where we have deployed both use-case services and generated load using a load generator tool which also collected performance metrics, including request response times.

As a measurement environment, we used a private cloud infrastructure deployed using OpenStack [38] platform (Figure 4). The infrastructure consisted of three servers, one used as a controller node and two as compute nodes (i.e., virtual machine hosts) with specifications available in Table 1.

Both use case services were deployed on the measurement environment using several instance sizes (Table 2) through which different resource quantities were allocated to the services so we can observe the effect of instance-sizing on the QoS and cost.

The MRS service web application, providing the application logic and user interface, and the MRS database were deployed on two separate instances. The video server was deployed on a single virtual machine instance. User requests for both use cases (Figures 2 and 3) were generated using JMeter load testing tool [39] that was installed on PCs outside the cloud environment (Figure 4).

We measured the consumption of resources defined by the proposed cloud application context attributes (Figure 1), which are listed in Table 3. Service resource usage data and parameters relevant for the quality of service, such as request response times, were collected each minute during the measurement periods of 15 minutes. Each 15-minute sample was gathered under constant load. The measurements were performed with the number of concurrent users increasing from 1 user up to 150, with the step of 10 concurrent users for each following sample. We recorded values of measured resources consumption for each sample.

After conducting performance measurements and collecting resource utilisation data, we calculated the cost of

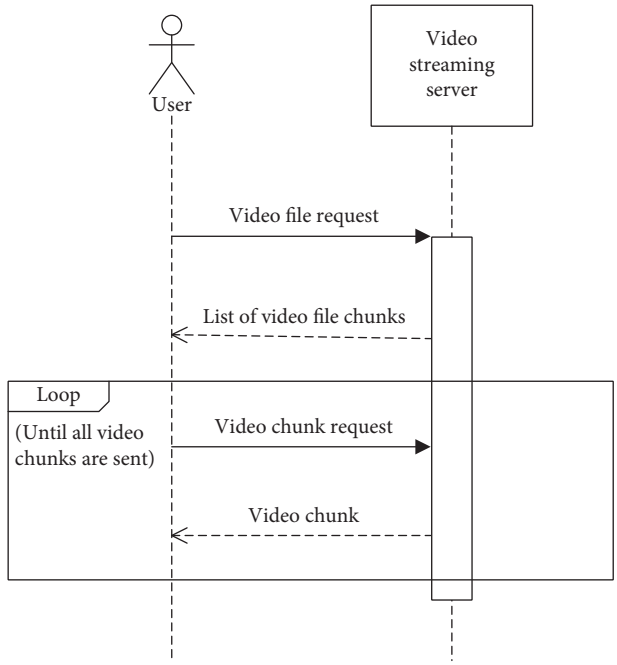

Figure 3: UML sequence diagram of the video streaming use-case scenario.

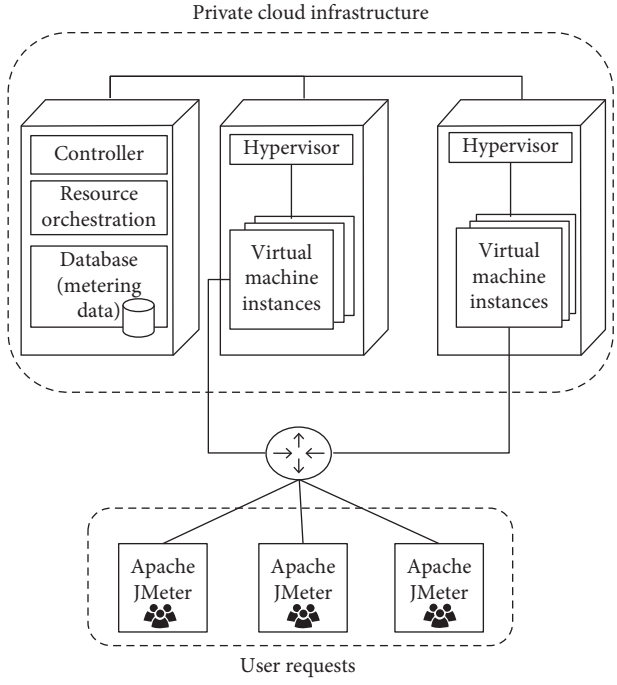

Figure 4: Architecture of the measurement environment.

running both services for 24 hours using publicly available price calculators of seven cloud infrastructure providers [40–46] and a method of calculation described in our previous work [47].

Table 3 brings a comparison of measured parameters and properties of the cloud application context (Figure 1). For this analysis, we do not consider different user types. We focus on user requests for single application task described in scenarios in

Table 1: Specifications of hardware used for a measurement cloud environment.

Node role	System	CPU	Memory	Storage
Controller	HP ProLiant BL460c Gen8	32 × 2.00 GHz	16 × 16.0 GB 256.0 GB total	2 drives 0.6 TB total
Compute	HP ProLiant BL460c Gen8	32 × 2.00 GHz	16 × 16.0 GB 256.0 GB total	2 drives 0.6 TB total
Compute	HP ProLiant DL 380 G6	16 × 2.93 GHz	8 × 8.0 GB 64.0 GB total	1 drive 1.9 TB total

Table 2: Resources quantity specified by instance types used for measurements.

Instance type	RAM (MB)	CPU (virtual cores)	Storage (GB)
Small	2048	1	20
Medium	4096	2	40
Large	8192	4	80

Table 3: Mapping of measured data and proposed cloud application context.

Parameter ID	Parameter domain	Parameter name	Unit of measurement	Related cloud application context attribute (Figure 1)
1	Application	Average CPU utilisation	%	App. resource usage profile: CPU
2	Application	Number of virtual CPU cores	Integer	App. resource usage profile: CPU
3	Application	Network: average incoming byte rate	B/s	App. resource usage profile: network
4	Application	Network: ingress traffic	GB	App. resource usage profile: network
5	Application	Network: average outgoing byte rate	B/s	App. resource usage profile: network
6	Application	Network: egress traffic	GB	App. resource usage profile: network
7	Application	Average RAM used	MB	App. resource usage profile: RAM
8	Application	Average disk read bytes rate	B/s	App. resource usage profile: storage
9	Application	Average disk write bytes rate	B/s	App. resource usage profile: storage
10	Application	Used storage	GB	App. resource usage profile: storage
11	Application	Request response time	ms	SLA requirements
12	Application	Number of parallel users	Integer	Service users: concurrent users
13	Deployment	Instance vCPU cores	Integer	Resource properties: CPU
14	Deployment	Instance RAM	MB	Resource properties: RAM
15	Deployment	Instance storage	GB	Resource properties: storage
16	Deployment	Provider_n	Not applicable	Infrastructure provider
17	Deployment	Price	USD	Pricing model and resource price

the previous section. In the case of more complex services, however, various user types should be included for a comprehensive picture of user-based aspects that might affect the load. In this work, we observe the number of concurrent users, a parameter that affects the application load directly. Similarly, we calculate the cost solely based on infrastructure providers' pricing offers available through the information and calculators available publicly on the web, hence not including a broad spectrum of different pricing models or pricing bundles available to customers upon agreement with the infrastructure service provider.

6. Service Resource Usage Analysis

To explore the data and prepare datasets for model training, as well as to be able to interpret the results better, we examine relationships between observed parameters described in the previous section. We also analyse the differences between the two selected use-case applications in terms of resource utilisation.

Figures 5 and 6 present the average resource utilisation values for MRS and video streaming service, respectively. Each diagram contains information about service resource consumption depending on the service load, i.e., number of concurrent users, and instance size the service was deployed on (Table 2). To demonstrate the spread of the data from the mean values, we present the standard deviation for the response time and CPU utilisation values. As observed, a request response time for both services displays linear growth in the investigated load range. MRS service response times demonstrate similar values in case of medium and large instances, but they get significantly longer when the service is deployed on a small instance. Such observation is useful in terms of service placement since a very similar QoS level is achieved for different image sizes, regardless of the higher CPU utilisation of the medium instance. As QoS parameters, we observe the response time and the number of SLO violations of the request response time, which are significantly higher when MRS service is deployed on a small instance. The difference between

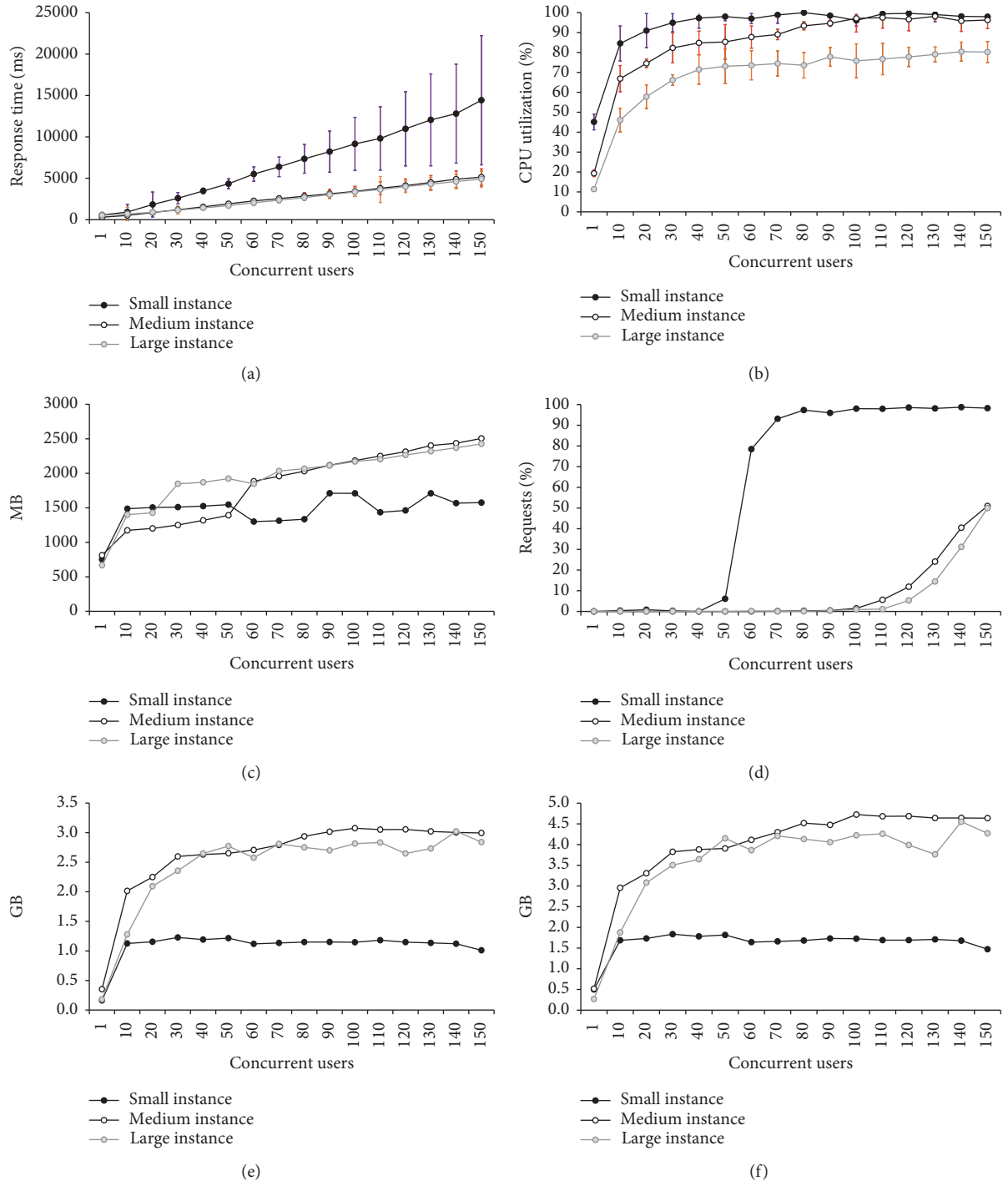

Figure 5: Continued.

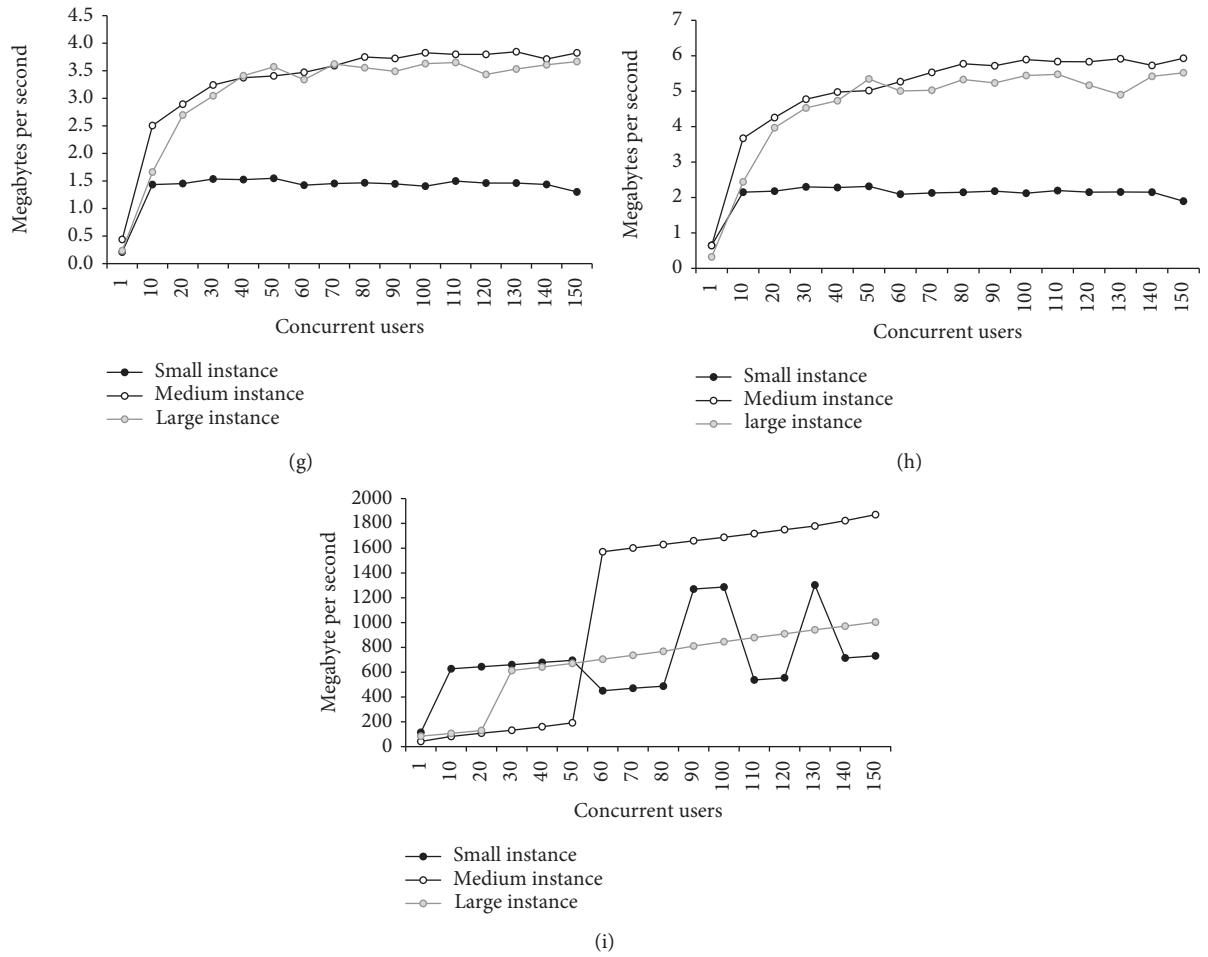

Figure 5: Analysis of resource usage for MRS service: (a) average response time, (b) average CPU utilisation, (c) average RAM used, (d) number of SLO violations, (e) network ingress traffic, (f) network egress traffic, (g) network incoming byte rate, (h) network outgoing byte rate, and (i) average disk write rate.

medium and large instance deployment in terms of percentage of SLO violations is less substantial. For example, if the SLA allows for 5% response time SLO violations, the medium instance will be able to serve up to 110 concurrent users without violating the SLA, and the large instance will be able to serve a maximum of 120 simultaneous users. For comparison, in case of a small instance size, the limit of 5% SLO violations would be reached below 50 concurrent users. The observed number of maximum simultaneous users indicates that the choice of an instance size oriented towards cost minimisation should be based on the expected service load and defined SLA.

As expected and observed for both services, CPU utilisation increases with the allocation of smaller number of vCPUs through instance sizes due to less available processing power. RAM utilisation increases with the load and instance size since it is possible to allocate more memory as the number of requests increases.

Network utilisation of MRS service demonstrates similar values for medium and large instances, with notable less traffic and byte rates for small instance due to its limited user request processing rate.

Other than previously commented differences in CPU and RAM utilisation caused by the amount of resources allocated to service, we also performed an analysis of the resource utilisation for the video streaming service (Figure 6). It demonstrates there is no significant difference in the network utilisation or quality of service delivered to the end users for any observed instance size, which is expected, taken into account the low level of CPU utilisations (maximum 30% for small instance).

As a representative of disk-related metrics, we also display average disk write rate. Since neither of the services is data-intensive in terms of reading or writing data, the increase in the disk write rate is mainly related to the application logging.

When the two services are compared based on resource usage, it can be noticed that the MRS service utilises more CPU and RAM for the same number of concurrent users. The network ingress traffic amount is similar for both services since it is in both cases consisting of simple HTTP requests. As expected, video streaming service has significantly higher egress traffic compared to the MRS service as its egress traffic is generated due to streaming of a video file.

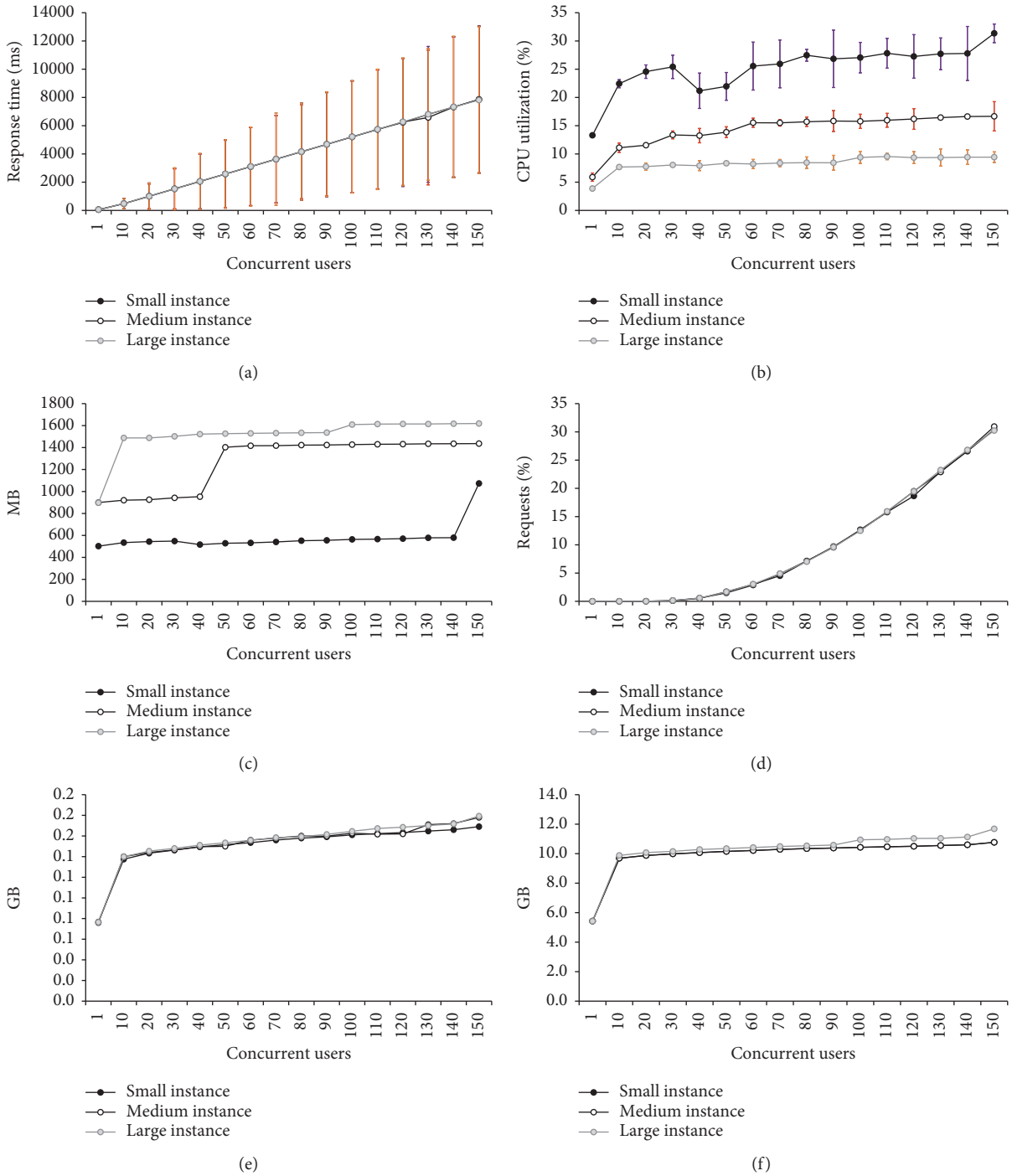

Figure 6: Continued.

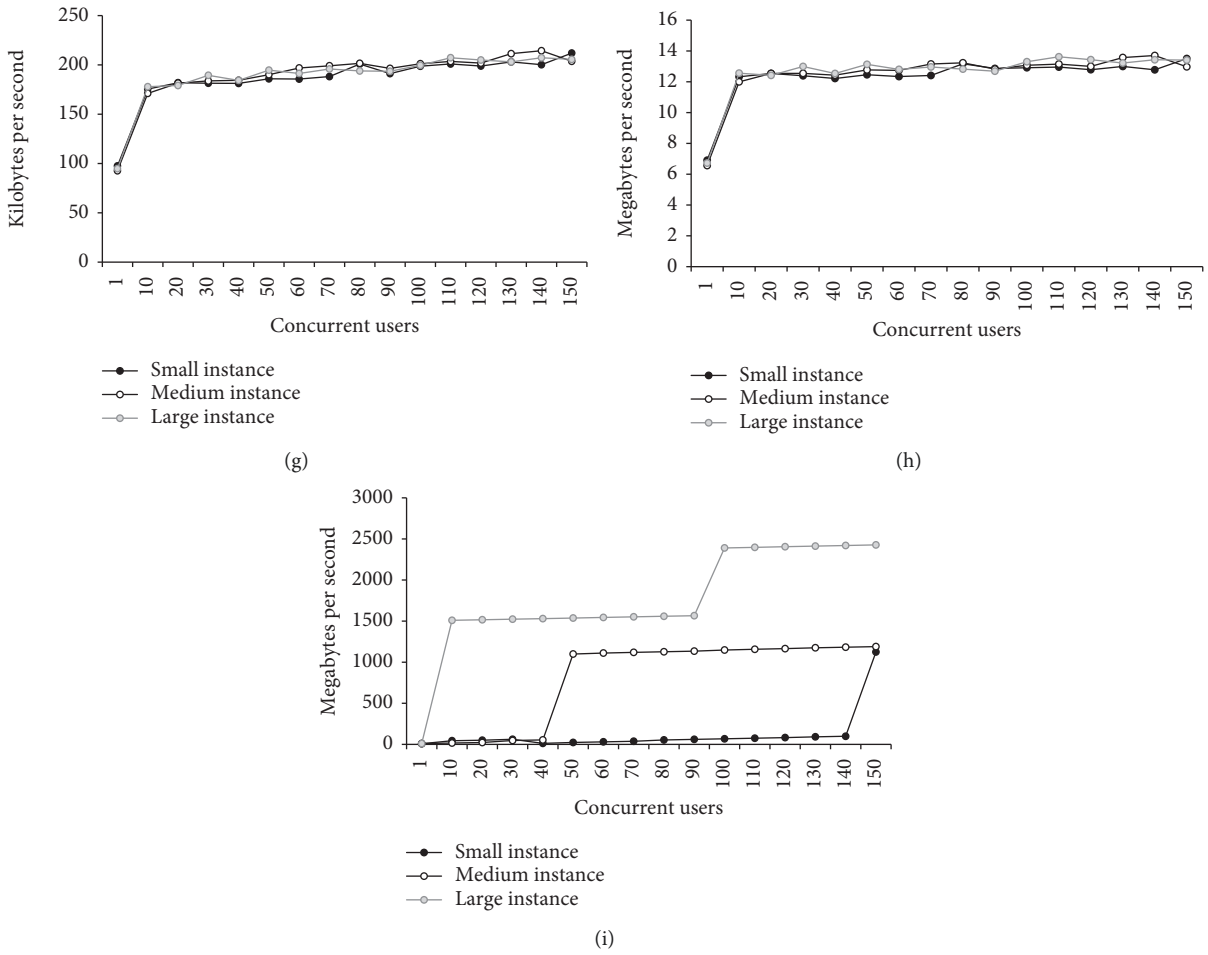

Figure 6: Analysis of resource usage for video streaming service: (a) average response time, (b) average CPU utilisation, (c) average RAM used, (d) number of SLO violations, (e) network ingress traffic, (f) network egress traffic, (g) network incoming byte rate, (h) network outgoing byte rate, and (i) average disk write rate.

To gain a better understanding of the dependency between the instance size, number of concurrent users, and the response time of a user request, we further examine empirical cumulative distribution function (ECDF) of response times under different load (marked by concurrent users) for both services deployed on small, medium, and large instance sizes (Figures 7–12).

Empirical CDF describes the probability that a random variable, in this case, the response time, will have the value less or equal to x. The ECDF plots of the MRS service (Figures 7–9) indicate significant differences in the response times, depending on the size of the instance and the number of concurrent users. The cumulative distribution functions of response times for observed simultaneous users demonstrate that the response time is growing rapidly with the load in all three examined instance sizes. When comparing Figures 7–9, it can be observed that the response times are increasing more slowly for any concurrent user number as the amount of resources allocated to the service increases with the instance size, which is consistent with Figures 5(a) and 5(d).

The ECDF plots of the video streaming service (Figures 10–12) show denser cumulative distribution

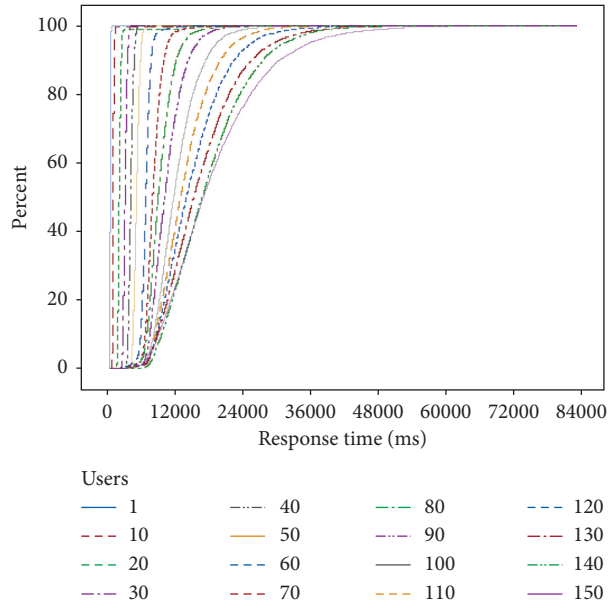

Figure 7: ECDF plot of response time—MRS service, small instance.

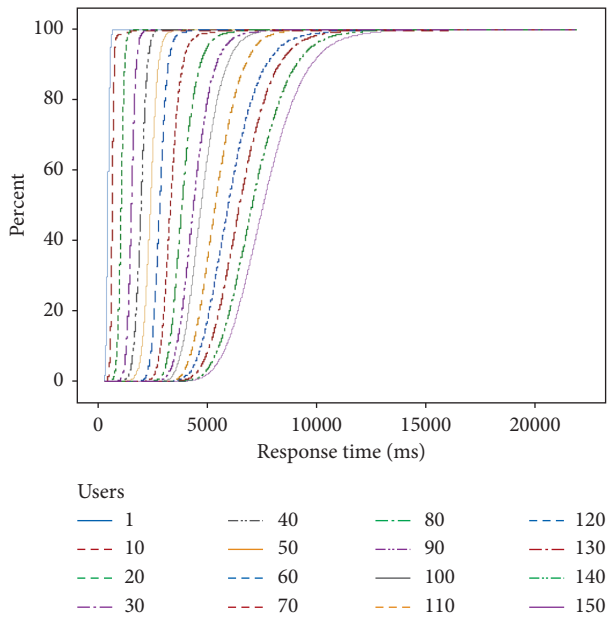

Figure 8: ECDF plot of response time—MRS service, medium instance.

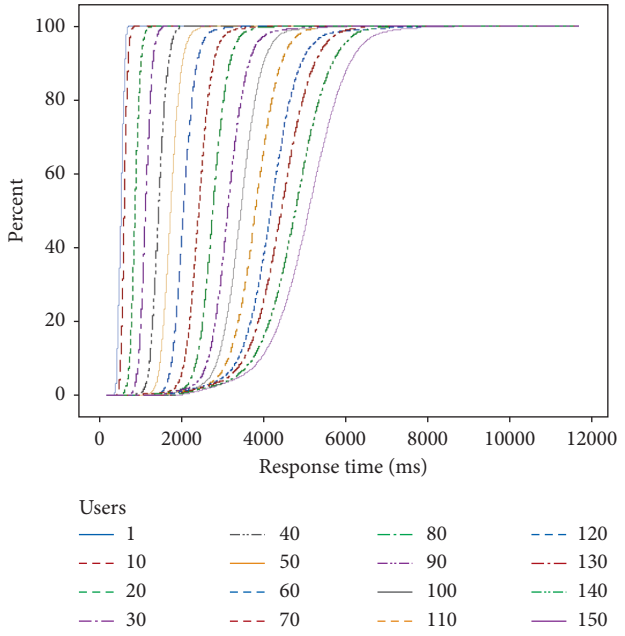

Figure 9: ECDF plot of response time—MRS service, large instance.

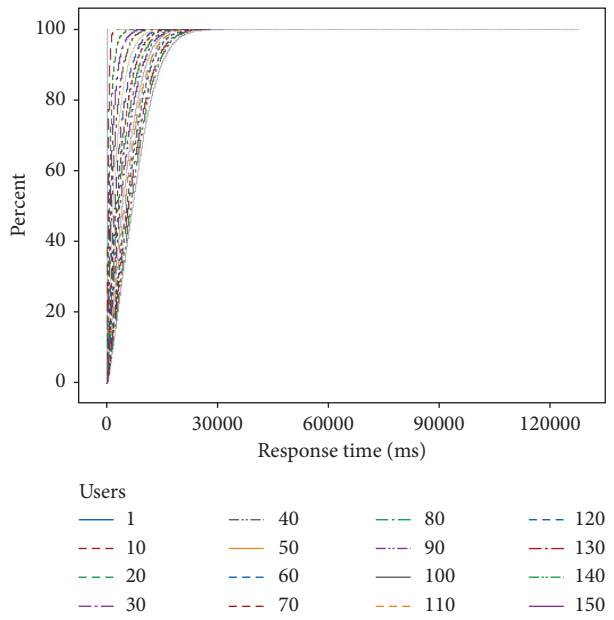

Figure 10: ECDF plot of response time—video service, small instance.

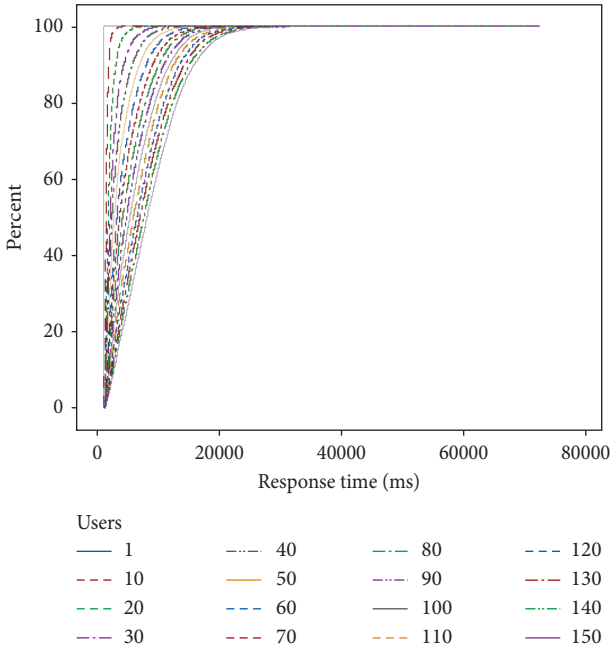

Figure 11: ECDF plot of response time—video service, medium instance.

functions; they are less differentiated and display less increase of response times with load when compared to the MRS service. There are almost no observable differences between ECDFs related to the video streaming service deployed on medium and large instance size, indicating that deploying the service on the large size instance would be overcapacitating for the observed range of concurrent users. Certain difference is visible when comparing the small instance deployment (Figure 10) with medium (Figure 11) and large (Figure 12) instance deployments of video streaming service. The small instance has steeper ECDFs indicating more rapid growth of response time with the higher number of concurrent users (more than 130 parallel users). However, medium and large instances have approximately the same ECDFs, which is also shown in Figure 6. Such observation cannot be made in the case of the MRS service where response time is decreasing as service gains more available resources through different instance sizes. To illustrate, the maximal response time for 150 concurrent users of the MRS service is 83027 ms when deployed on a small instance,

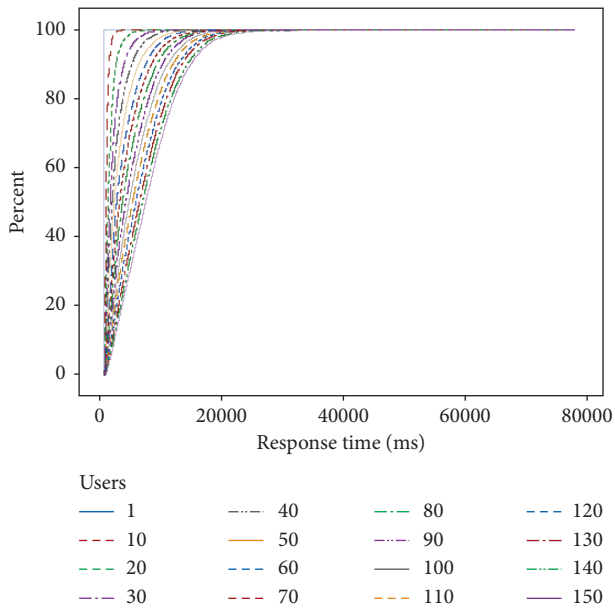

Figure 12: ECDF plot of response time—video service, large instance.

21901 ms for a medium instance, and 11703 ms for a large instance. As expected, differences for the video streaming service are not that considerable—maximal response time for 150 concurrent users reaches 127420 ms for a small instance, 71229 ms for a medium instance, and 79906 ms for a large instance deployment.

7. Prediction and Feature Importance

To gain a better understanding of service execution cost and QoS, our goal was to create cost and response time models for the two described use cases, in order to accurately predict the model outputs and use such models as a base for extracting the knowledge about model features that significantly affect cost and QoS. In this section, we describe methods used to implement models and analyse the feature importance. We also describe metrics used for evaluation of model prediction accuracy.

Our choice of algorithms for model implementation was based on the size of the collected datasets, predictive nature of the model, and the goal of enabling cloud application providers an easy approach for evaluation of the best infrastructure provider and service placement options that would require obtaining only smaller experimental datasets. Since we wanted to explore the possibilities of predictive models, we performed analysis using regression techniques, including linear regression, regularisation regression techniques, and neural networks.

7.1. Linear Regression. Linear regression [48] is used to determine the equation of a line that best fits the observed data and models the relationship between input and output variables. A most commonly used form of linear regression is multiple linear regression which uses more independent variables and one continuous dependent variable.

The relation between model input variables and output is defined by (1) where $\hat{y}$ is predicted output value, β_0 is a model intercept, and x_i is a model input variable with its coefficient β_i:

$$\hat{y} = \beta_0 + \beta_{1x_1} + \ldots + \beta_{ix_i}. \quad (1)$$

The regression coefficients β_i are determined by minimising the residual sum of squares denoted by RSS and given by

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2, \quad (2)$$

where y_i is the i^{th} observation and $\hat{y}_i$ is the predicted value of i^{th} observation calculated according to (1).

7.2. Least Absolute Shrinkage and Selection Operator (Lasso). Least absolute shrinkage and selection operator (Lasso) regression [49] uses regularisation technique that can deal with multicollinearity in the data by penalising the absolute size of the coefficients in the regression model. This method will generalise the model and hence avoid overfitting that might otherwise occur due to complex relationships between variables. The Lasso method uses the absolute value of the regression coefficient as a penalty term in the loss function, according to the following equation:

$$L_{\text{lasso}} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{m} \left|\hat{\beta}_j\right|, \quad (3)$$

where λ is the regularisation penalty parameter, $\hat{\beta}$ is the vector of regression coefficients, m is the number of input variables, y_i is the i^{th} observation out of n observations, and $\hat{y}_i$ is the predicted value of i^{th} observation.

7.3. Least Angle Regression (LARS). The Least Angle Regression (LARS) [49] provides a method for piecewise construction of the linear path to efficiently solve the Lasso regression described previously. The LARS method is based on forward stepwise regression, meaning that it will add standardised predictors one by one as the model is being constructed. In each step of the LARS method, the best variable for inclusion is selected based on its absolute correlation with the residual, and the penalisation of coefficients is, similar to Lasso method, performed using the regularisation penalty parameter λ. The entire method applied to solve the Lasso problem can be found in [50].

7.4. Multivariate Adaptive Regression Splines (MARS). Multivariate Adaptive Regression Splines (MARS) [51] is a nonparametric adaptive regression method commonly used for problems with a large number of inputs and potential nonlinearities in the data. MARS implements the strategy of forward model building similar to stepwise linear regression with the difference of using base functions instead of using input variables x_i directly. The model is defined in the following form:

$$\widehat{y} = \beta_0 + \sum_{i=1}^{m} \beta_i h_i(X), \tag{4}$$

where $\widehat{y}$ is the predicted output value, β_0 is the model intercept, h_i is a basis function, and β_i is the coefficient of the basis function h_i. Similar to the linear regression, β_i coefficients are estimated by minimising the residual sum of squares (2), and the best subset of basis functions h_i included in the model is selected based on generalised cross validation [51].

7.5. Neural Network. Artificial Neural Network (ANN) [48] can model the dependent variable of the regression model as a nonlinear function using linear combinations of model input variables. ANNs consist of several layers—an input layer, one or more hidden layers, and an output layer. The connections of neurons in different layers have weights that are determined during the ANN training process.

The training is performed by feeding observations to the network input layer. The initialisation method is applied to assign the initial values of the weights, followed by calculation of activations at each network layer and eventually generating predicted output. After obtaining the predicted output value, backpropagation is used to update the network weights in order to minimise the loss function. The parameter that determines the rate of the weights update is called the learning rate. After the weights have been updated, another iteration, called an epoch, of calculating activations and updating network parameters can be started, until a specific termination condition has been reached, e.g., a small enough prediction error.

7.6. Model Evaluation Metrics. To evaluate model accuracy, as well as to compare modes implemented using different machine learning methods, we use several metrics: mean absolute error (MAE) [52], mean absolute percentage error (MAPE) [52], root mean square error (RMSE) [53], and coefficient of determination (R^2) [48].

Mean absolute error (MAE) is a measure of average absolute difference between the predicted and observed values of a variable defined by (5), where n is the number of observations in the dataset, m is the measured (observed) value, and p is the predicted value:

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |m - p|. \tag{5}$$

Mean absolute percentage error (MAPE) is defined according to (6), where n is the number of observations in the dataset, m is the measured (observed) value, and p is the predicted value. Lower values of the MAPE metric indicate better model accuracy:

$$\text{MAPE} = \left(\frac{1}{n} \sum_{i=1}^{n} \frac{|m - p|}{|m|} \right) \cdot 100\%. \tag{6}$$

Root mean square error is defined by the following formula:

$$\text{RMSE} = \sqrt{\frac{\sum_{i=1}^{n} (p - m)^2}{n}}, \tag{7}$$

where n is the number of observations in the dataset, m is the measured (observed) value, and p is the predicted value. Due to its definition, RMSE values will always be positive, with lower RMSE indicating a better model fit.

Coefficient of determination, also known as R^2, is an accuracy statistic for assessing the percentage of dependent variable variance explained by the model predictors. Used for comparison of models, the higher value of R^2 is usually an indicator of a better result. R^2 is defined by the following formula:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (p - m)^2}{\sum_{i=1}^{n} (p - \overline{m})}, \tag{8}$$

where n is the number of observations in the dataset, m is the measured (observed) value, $\overline{m}$ is the mean of the observed values, and p is the predicted value. In case of multiple linear regression, adjusted form of R^2 is used to penalise the addition of input variables that do not contribute to the variance of the predicted variable. The adjusted form of the R^2 is calculated according to the following formula:

$$R^2_{\text{adj}} = 1 - \frac{(1 - R^2)(n - 1)}{n - k - 1}, \tag{9}$$

where n is the number of observations, k is the number of independent variables used in the model, and R^2 is the coefficient of determination value calculated according to (8).

7.7. Permutation Importance. Recent approaches in machine learning research include the ability to interpret and explain the results of sophisticated but often complex machine learning and deep learning algorithms. To inspect the importance of model features, we used the permutation importance method. Permutation importance was introduced by Breiman in [54]. In this work, we use permutation importance analysis on the regularised regression models.

Permutation importance is calculated after the model has been fitted. The algorithm uses permutations of single feature values in the validation data and measures the effect that the permutations had on the accuracy of predictions. The concept behind the permutation method is based on the fact that the feature with the biggest impact on the predicted output data, i.e., the one that the model depends on extensively for predictions, will cause the most significant accuracy decline when permuted randomly. The method output value is the increase in prediction error in the case of single predictor values permuted compared to the prediction error with all variables in their intact state. Hence, as the permutation importance measures how much each feature contributes to the model, we examine the features with the highest positive permutation importance values that indicate the most significant effect on the variable predicted by the model.

8. Model Implementation and Result Analysis

In this section, we describe the models' implementation and evaluate the accuracy of models using the evaluation metrics specified in the previous section. After the report on the accuracy of the models, we present feature importance analysis based on permutation importance technique.

8.1. Model Implementation. After examining the data, we implement models for predicting the dependent variables of our interest. As we want to explore the effects of model features on service cost and QoS and predict them, we create two models for each service, using the cost and the request end-to-end response time as the predicted values. We apply a set of machine learning algorithms on both use-case datasets to observe which algorithm will provide the best result and how much the results differ between the two use cases, as well as between dependent variables. We also analyse which features have the most impact on model output. As potential model features, we consider input variables presented in Table 3, from which highly correlated features were removed from the feature set. All input variables were standardised and normalised as a part of the data preparation procedure. After feature selection was performed, the following predictors were selected for both models: *network egress (GB)*, *average RAM usage (MB)*, *average CPU utilisation (%)*, and *used storage (GB)*. In addition to the common set of predictors, in the case of QoS models, the number of parallel *users* was used as a predictor as well. The cost models had information about the infrastructure provider as an additional model input variable.

To create models, we used machine learning algorithms described in the previous section implemented in the Python-based environment (source code available at https://github.com/cloudSPO/cloudappcontext). The data collection process, described in Section 5, resulted in 336 samples of cost calculations and response time measurements. For all the models in this paper, we use 70% of the obtained data for training, and the remainder is used for validation of the model. As a baseline, we started with the simplest option of linear regression with the goal of creating cost and QoS models for both use cases. We used an open-source machine learning Python-based framework scikit-learn [55] for implementation of linear regression, Lasso, and LARS models, in addition with py-earth library [56] for MARS models. Since the development of regression models using the scikit-learn library is well documented and straight forward, we briefly report on essential parameters used in the models. The regularisation penalty parameter (λ) values used in the Lasso and LARS models, which demonstrated minimal model loss (Figure 13), are listed in Table 4. In the MARS model, we used two as the maximum degree of terms generated by the model forward pass and 1.0 as the penalty parameter used for the generalised cross validation.

In addition to regression models, we also deployed a neural network model for each of the observed dependent variables to examine if it could be possible to obtain better prediction results. For the neural network implementation,

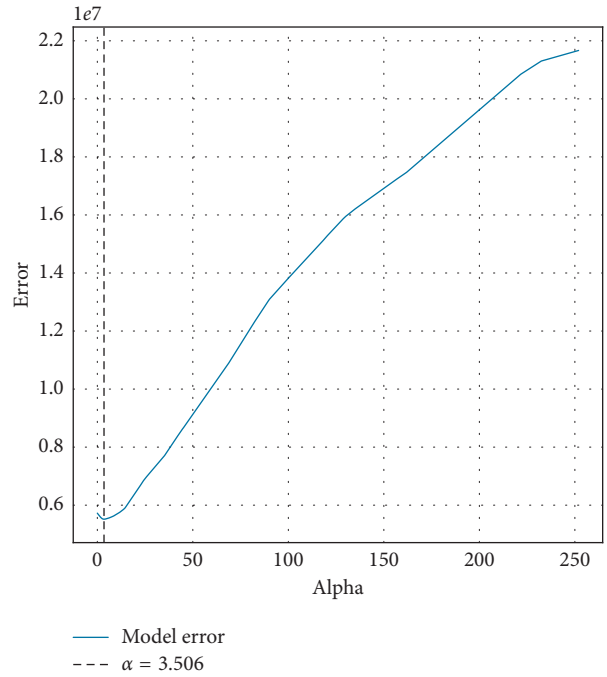

Figure 13: Model error over regularisation parameter alpha used in LARS model of MRS service cost.

Table 4: Regularisation penalty parameter (λ) values used in Lasso and LARS models.

Model	λ_{Lasso}	λ_{LARS}
MRS cost	10.08	3.506
Video streaming cost	8.687	0.3955
MRS response time	0.021	1.481
Video streaming response time	0.020	0.007

we used the open-source Python-based library Keras [57], which enables the development and evaluation of neural network models. We use the multilayer backpropagation neural network architecture, consisting of an input layer, three hidden layers, and an output with a single neuron used for predicting response time or cost.

The learning rate parameter was set to 0.01. Plots of the model loss values over epochs indicate this value was a good learning rate choice for each of the ANN models. As an example, we show the model loss over the training epoch number for ANN predicting the cost of the MRS service (Figure 14).

As pointed out in Section 2, the authors of [34] base their choice of neural network model parameters on literature review and suggest using hyperbolic tangent activation function (*tanh*) [58] as an activation function together with the Xavier [59] weight initialisation algorithm. Since we address the similar problem of utilisation prediction, we experimented with the suggested parameters in our model. In our experiments, better results were obtained using rectified linear unit (*relu*) activation function [60] and Xavier initialisation function, apart from the ANN predicting the cost of the video streaming service which demonstrated better results using the *tanh* activation

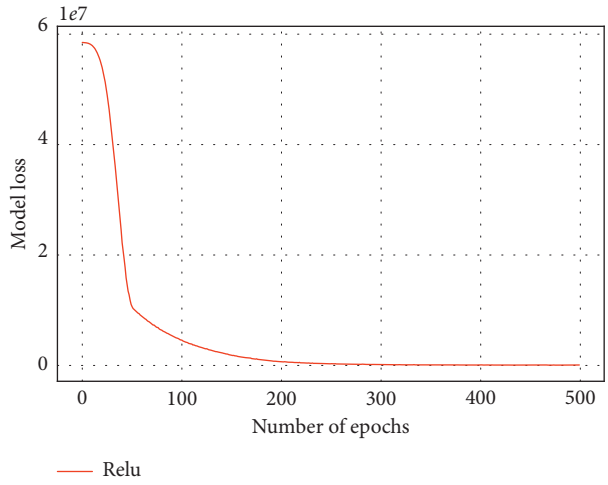

Figure 14: Model loss over number of ANN training epochs—MRS service cost model, *relu* activation function.

function. The comparison of results for ANN models when using different activation functions, with a fixed learning rate (0.01) and the number of training epochs (500), can be seen in Table 5. Due to the stochastic nature of the implemented neural network models, we report the average metric values based on 50 model runs.

As it can be seen from Table 5, ANN models using tanh activation function and Xavier initialisation algorithm performed somewhat worse for the same number of epochs and learning rate compared to ANN models using relu as activation function, except for the cost model of the MRS service. We select the best results for each ANN model and report them in the next section.

8.2. Model Accuracy. In this section, we report the model accuracy of the implemented models and compare them based on the error estimation metrics specified in Section 7.6 section—MAE, MAPE, RMSE, and adjusted R^2. We report values of the error metrics in Table 6.

It can be noticed that the best accuracy is in almost all cases achieved with the MARS models. As an example, we present the prediction error of the response time model for the MRS service implemented using the MARS method in Figure 15. The ANN cost models of both use cases have somewhat lower model accuracy, compared to the QoS models, which can be explained by the different prices and pricing models offered by cloud infrastructure providers that affect service execution cost. Various pricing schemes could make it harder to produce patterns for cost prediction, especially when not using an extensive dataset. Contrary to our expectations and the results presented in [34], the ANN models demonstrated less accurate results compared to the regularisation regression techniques. It is likely that the accuracy of the ANN models would improve with the larger sample size. Although one might argue that we use a small dataset, according to [61], the number of samples used for this analysis is sufficient, taking into account the techniques we used to implement our models and the number of input

Table 5: Comparison of ANN model accuracy using *relu* and *tanh* activation functions.

Activation function	MAE	MAPE	RMSE	Adj. R^2
Response time, MRS				
relu	221.05	6.24	363.62	0.9953
tanh	362.35	8.75	523.66	0.9899
Response time, video streaming				
relu	144.96	7.23	218.15	0.9867
tanh	196.03	9.17	263.05	0.9860
Cost, MRS				
relu	169.51	16.37	248.86	0.9954
tanh	124.91	10.66	192.71	0.9979
Cost, video streaming				
relu	531.55	17.26	719.94	0.9738
tanh	553.18	19.71	782.90	0.9750

Table 6: Comparison of model accuracy.

Metric	Linear regression	Lasso	LARS	MARS	ANN
Response time, MRS					
MAE	243.39	476.62	**181.11**	181.47	221.05
MAPE (%)	9.33	19.82	5.79	**5.61**	6.24
RMSE	368.56	653.35	**263.47**	270.47	363.62
Adj. R^2	0.9942	0.9817	0.9968	**0.9969**	0.9953
Cost, MRS					
MAE	142.98	113.68	**99.00**	106.25	124.91
MAPE (%)	22.82	13.47	**6.33**	15.19	10.66
RMSE	270.99	257.48	212.86	**200.94**	202.71
Adj. R^2	0.9963	0.9967	0.9977	**0.9980**	0.9979
Response time, video streaming					
MAE	266.80	287.26	168.65	**104.87**	221.05
MAPE (%)	10.54	20.50	11.98	**5.09**	6.24
RMSE	495.25	401.45	226.01	**149.86**	363.62
Adj. R^2	0.9588	0.9729	0.9914	**0.9962**	0.9867
Cost, video streaming					
MAE	331.24	234.18	123.59	**72.77**	531.55
MAPE (%)	10.98	5.88	3.84	**2.02**	17.26
RMSE	463.54	333.45	209.41	**186.87**	719.94
Adj. R^2	0.9982	0.9991	0.9996	**0.9997**	0.9750

Best values for each model and metric are marked in bold.

variables. The obtained results demonstrate the ability of the regularised regression techniques to produce the accurate models even with the relatively small sample size which would prove especially useful in the prototyping and early preproduction phases of the cloud service development.

8.3. Permutation Importance. In addition to model accuracy, we observe the importance of features used for predicting cost and response time for both use cases so we can gain knowledge about the factors that affect the service execution cost and QoS to the greatest extent. To assess the importance of observed features, we calculated the permutation importance using Python-based library Eli5 [62]. As a base for the permutation importance method, we chose models with the best accuracy results (Table 6). We report the results for MARS models since they demonstrated the best accuracy for both models of the two use cases. For all models, we used 150

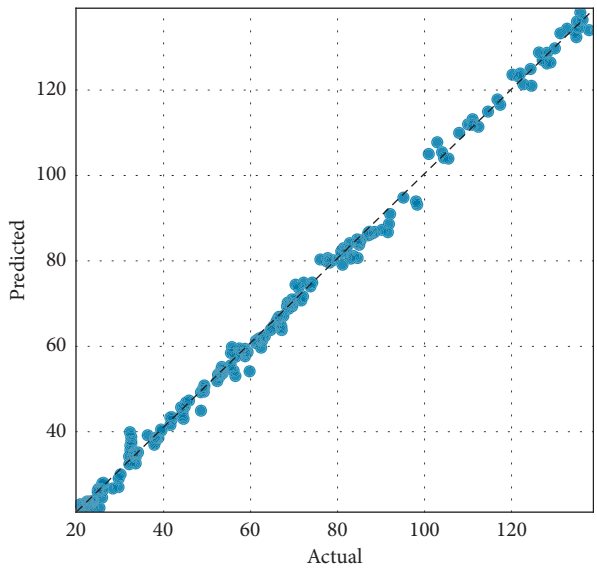

Figure 15: Prediction error of the response time model for the MRS service (method: MARS).

iterations to obtain the permutation importance values for the variables. Weight values indicate the effect that each variable had on the prediction accuracy, i.e., higher positive values indicate the more significant impact of the feature on the predicted variable. Negative values, not present in our results, would suggest that the permutation of the feature values resulted in the increased model accuracy, hence making such features dispensable in their initial intact state. The absence of negative values is a good indicator that the relevant set of features was selected to build our models. We report the top four features for each model. The higher feature weight value rank indicates the higher importance of the feature in comparison to other predictors.

We first examine cost models for both services. The permutation importance ranks of the top four features for the model of video streaming service cost can be seen in Table 7. The feature with the highest importance for video streaming cost model was the amount of egress network traffic, which can be explained by the heavy network load generated by transferring video files. Since the egress traffic is charged on a pay-per-use basis, its amount directly affects the overall service execution cost. The highest ranked feature in case of MRS cost model (Table 8) is also the amount of network egress traffic. However, the weight value of the network egress traffic is lower than the weight of the same feature in the video streaming cost model, indicating less impact on the execution cost amount due to the less generated egress traffic in the MRS service for the same load. The network egress feature is followed by variables marking a provider of the cloud infrastructure, similar to the video streaming service cost model with the infrastructure service providers ordered differently. Such rank might indicate that the prices of the resources relevant for observed task execution offered by the provider with the highest weight value in the feature list resulted in the highest service execution cost. Although the ranks differ somewhat in MRS and video

Table 7: Permutation importance feature rank for the cost model of video streaming service.

Weight	Feature
1.6223 ± 0.4371	NW egress (GB)
0.9892 ± 0.4441	Service provider 1
0.8785 ± 0.4269	Service provider 2
0.6277 ± 0.2374	Service provider 3

Table 8: Permutation importance feature rank for the cost model of MRS service.

Weight	Feature
1.0311 ± 0.5252	NW egress (GB)
0.9498 ± 0.5958	Service provider 2
0.8550 ± 0.5181	Service provider 1
0.6938 ± 0.3571	Service provider 3

streaming service, infrastructure providers 1 and 2 in both cases have similar permutation importance weight values, and infrastructure provider 3's weight value is lower for both services, demonstrating consistency regarding the effect that the prices and pricing models have on the overall application service execution cost.

The rank of the features for both cost models provides directions to the application provider in terms of service placement optimisation with the goal of reducing the service cost. Based on results for two observed services in our analysis, good options for cost reduction, in terms of service placement, would include cloud infrastructure providers that offer lower network egress prices, since this is the feature that had the most impact on the execution cost. Another option might be to optimise the content that the service sends over the network to reduce the overall amount of the generated egress traffic, if possible.

In addition to cost, we observe what features affect the response time the most. The permutation importance results for QoS model of video streaming (Table 9) and MRS service (Table 10) demonstrate the same feature that has the most impact on the prediction of the response time—the number of parallel users. This is an expected result since the number of parallel user requests affects the resource utilisation and the ability to process request efficiently. Since we are interested in identifying a feature impactful enough to response time in terms of service placement, we additionally observe features that ranked highest after the number of concurrent users. In the case of video streaming service, the feature ranked second is network egress traffic, indicating that the streaming chunks of video data had an impact on the response times due to the limited bandwidth. Other features seem to have little or no effect on the response time of the video streaming service. After the number of concurrent user requests, the MRS service response times depended mainly on the used RAM, meaning that the processing of the user requests had a more significant impact on the MRS service response times than the bandwidth consumption, as compared with the video streaming service. This observation can be used by the application provider to identify the resources crucial for achieving adequate service QoS level.

Table 9: Permutation importance feature rank for QoS model of video streaming service.

Weight	Feature
2.2935 ± 0.7169	Users
0.0299 ± 0.0147	NW egress (GB)
0.0001 ± 0.0001	Avg. RAM used (MB)
0 ± 0.0000	Disk read bytes

Table 10: Permutation importance feature rank for QoS model of MRS service.

Weight	Feature
2.0382 ± 0.3191	Users
0.8644 ± 0.1578	Avg. RAM used (MB)
0.4996 ± 0.1399	Disk read bytes
0.4075 ± 0.1038	NW egress (GB)

Considering the feature importance rank for the video streaming service, the application provider should, when choosing the service placement, prioritise network bandwidth and network ports offering greater throughput to avoid bottleneck for achieving adequate service quality. The application owner of the MRS service should prioritise the placement of the service using the instances that provide enough RAM to process user requests efficiently.

9. Conclusion

Cloud computing enables cloud application providers, i.e., SaaS providers, to host their applications in the cloud environments and to be charged for the used computing resources in a pay-per-use manner. Renting public cloud infrastructure will undoubtedly cut down capital investments, but it can still be a very complex task for the SaaS providers to determine what is the optimal service placement and how to choose the cloud infrastructure provider to minimise the service cost while maintaining the appropriate QoS levels.

In this paper, we propose an approach that allows SaaS providers to predict service execution cost and observed QoS parameters for a cloud application and to extract knowledge about the features of the cloud application context that affect the cost and QoS to the greatest extent. For our experiments, we used two applications as cloud service use cases—a medical record system and a video streaming service. We propose a set of features used to implement service models. The features we use are chosen to be applicable for any application that can be deployed as a SaaS service, unlike many solutions available in the literature that implement service models using features specific for a particular type of application. In that way, the proposed approach can be applied to a broad spectrum of applications hosted in cloud environments.

In our approach, we use various machine learning methods fit for the size of our acquired dataset—linear regression, regularisation regressions, and neural network. To evaluate models, we use several error metrics for the purpose of comparing the implemented models. The results demonstrate the ability of regression models to accurately predict the cost and QoS parameters of applications deployed in the cloud. The proposed method can be applied even when an extensive dataset is not available for the analysis, compared with many solutions available in the literature that require service already deployed in a production environment. We consider this especially relevant for services that are still in the early operational phase when there is no possibility to acquire large datasets, but the decision on the production environment and cloud infrastructure providers has not yet been made. We further demonstrate the ability to extract the knowledge about the main contributors to the observed predicted variables, using the permutation importance method that can be applied even on the models implemented using sophisticated techniques such as neural networks. This property of the permutation importance method enables the applicability of our approach to a wide range of models, allowing the analysis of the effect of the cloud application context on application's cost and QoS even in the case of complex cloud services.

To summarise, the results presented in this paper demonstrate that it is possible to achieve high model accuracy by using generalised features related to cloud application and infrastructure, which can be applied to any application service deployed in a cloud environment. Also, we show that the permutation importance method can be used to extract knowledge about the impact of cloud application context parameters to the service cost and QoS. Lastly, we demonstrate that the proposed approach can be performed without acquiring large datasets, which makes it beneficial for usage in the preproduction phase of the service lifecycle.

The main aim of our future work is to utilise the knowledge about cloud application context parameters that have the most significant impact on service cost and QoS in the optimisation of the cloud application service placement. Information on the effect of predictors can be used as an input for an algorithm dealing with decision making on the cost-minimising service placement in cloud environments, which will at the same time fulfil the requirements on the service quality. The application owners can obtain this knowledge before deploying service in its production environment, and the model-based approach enables prediction of various service placement outcomes. We believe that such approach can be valuable to the cloud application service providers, especially in the context of scientific applications that include a broad spectrum of various application types.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

This study was performed as a part of the employment of the corresponding author at the Research Unit of Ericsson Nikola Tesla d.d.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

[1] K. Buckley, *By 2019, 60% of IT Workloads Will Run in the Cloud*, 451 Research, Boston, MA, USA, 2018, https://451research.com/blog/1910-by-2019,-60-of-it-workloads-will-run-in-the-cloud.

[2] L. Columbus, *83% of Enterprise Workloads Will Be in The Cloud By 2020*, Forbes Media LCC, New York, NY, USA, 2019, http://www.forbes.com/sites/louiscolumbus/2018/01/07/83-of-enterprise-workloads-will-be-in-the-cloud-by-2020/.

[3] P. M. Mell and T. Grance, *The NIST Definition of Cloud Computing*, NIST Special Publication National Institute of Standards and Technology, Gaithersburg, MD, USA, 2011, https://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-145.pdf.

[4] RightScale 2019 State of the Cloud Report from Flexera, 2019, https://www.rightscale.com/lp/state-of-the-cloud.

[5] O. Rogers and W. Fellows, *The Cloud Pricing Codex–2013*, 451 Research LLC, Boston, MA, USA, 2013.

[6] A. Kaur, B. Kaur, and D. Singh, "Optimization techniques for resource provisioning and load balancing in cloud environment: a review," *International Journal of Information Engineering and Electronic Business*, vol. 9, no. 1, pp. 28–35, 2017.

[7] A.-F. Antonescu and T. Braun, "Simulation of SLA-based VM-scaling algorithms for cloud-distributed applications," *Future Generation Computer Systems*, vol. 54, pp. 260–273, 2016.

[8] J. Yu and R. Buyya, "Scheduling scientific workflow applications with deadline and budget constraints using genetic algorithms," *Scientific Programming*, vol. 14, no. 3-4, pp. 217–230, 2006.

[9] L. Qi, J. Yu, and Z. Zhou, "An invocation cost optimization method for web services in cloud environment," *Scientific Programming*, vol. 2017, Article ID 4358536, 9 pages, 2017.

[10] W. Li, S. Petter, T. Johan, and E. Erik, "Cost-optimal cloud service placement under dynamic pricing schemes," in *Proceedings of the 2013 IEEE/ACM 6th International Conference on Utility and Cloud Computing*, IEEE Computer Society, Dresden, Germany, December 2013.

[11] Y. Zhang, J. Yao, and H. Guan, "Intelligent cloud resource management with deep reinforcement learning," *IEEE Cloud Computing*, vol. 4, no. 6, pp. 60–69, 2017.

[12] G. Skourletopoulos, C. X. Mavromoustakis, G. Mastorakis, J. N. Sahalos, J. M. Batalla, and C. Dobre, "A game theoretic formulation of the technical debt management problem in cloud systems," in *Proceedings of the 2017 14th International Conference on Telecommunications (ConTEL)*, pp. 7–12, IEEE, Zagreb, Croatia, June 2017.

[13] S. Memeti, S. Pllana, A. Binotto, J. Kołodziej, and I. Brandic, "Using meta-heuristics and machine learning for software optimization of parallel computing systems: a systematic literature review," *Computing*, vol. 101, no. 8, pp. 893–936, 2018.

[14] S. H. H. Madni, M. S. A. Latiff, Y. Coulibaly, and S. M. Abdulhamid, "An appraisal of meta-heuristic resource allocation techniques for IaaS cloud," *Indian Journal of Science and Technology*, vol. 9, p. 4, 2016.

[15] Z. Zhu, G. Zhang, M. Li, and X. Liu, "Evolutionary multi-objective workflow scheduling in cloud," *IEEE Transactions on Parallel and Distributed Systems*, vol. 27, no. 5, pp. 1344–1357, 2016.

[16] A. Abdelaziz, M. Elhoseny, A. S. Salama, and A. M. Riad, "A machine learning model for improving healthcare services on cloud computing environment," *Measurement*, vol. 119, pp. 117–128, 2018.

[17] Y. Xu, J. Yao, H.-A. Jacobsen, and H. Guan, "Cost-efficient negotiation over multiple resources with reinforcement learning," in *Proceedings of the 2017 IEEE/ACM 25th International Symposium on Quality of Service (IWQoS)*, pp. 1–6, IEEE, Barcelona, Spain, June 2017.

[18] Z. Li, J. Ge, H. Hu, W. Song, H. Hu, and B. Luo, "Cost and energy aware scheduling algorithm for scientific workflows with deadline constraint in clouds," *IEEE Transactions on Services Computing*, vol. 11, no. 4, pp. 713–726, 2018.

[19] M. A. Rodriguez and R. Buyya, "Budget-driven scheduling of scientific workflows in IaaS clouds with fine-grained billing periods," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 12, no. 2, pp. 1–22, 2017.

[20] B. Ahmad, S. McClean, D. Charles, and G. Parr, "Analysis of energy saving technique in CloudSim using gaming workload," *Cloud Computing*, vol. 2018, p. 143, 2018.

[21] R. Pakdel and J. Herbert, "Scalable cloud-based analysis framework for medical big-data," in *Proceedings of the 2016 IEEE 40th Annual Computer Software and Applications Conference (COMPSAC)*, vol. 2, IEEE, Atlanta, GA, USA, June 2016.

[22] Á. Salánki, G. Kincses, L. Gönczy, and I. Kocsis, "Data analysis-based capacity planning of VCL clouds," *International Journal of Cloud Computing*, vol. 6, no. 4, pp. 370–383, 2017.

[23] F. Koch, M. D. Assunção, C. Cardonha, and M. A. S. Netto, "Optimising resource costs of cloud computing for education," *Future Generation Computer Systems*, vol. 55, pp. 473–479, 2016.

[24] A. N. Toosi, R. O. Sinnott, and R. Buyya, "Resource provisioning for data-intensive applications with deadline constraints on hybrid clouds using Aneka," *Future Generation Computer Systems*, vol. 79, pp. 765–775, 2018.

[25] C. B. Hauser, J. Domaschka, and S. Wesner, "Predictability of resource intensive big data and HPC jobs in cloud data centres," in *Proceedings of the 2018 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*, pp. 358–365, IEEE, Lisbon, Portugal, July 2018.

[26] E. N. Alkhanak, S. P. Lee, and S. U. R. Khan, "Cost-aware challenges for workflow scheduling approaches in cloud computing environments: taxonomy and opportunities," *Future Generation Computer Systems*, vol. 50, pp. 3–21, 2015.

[27] R. Ranjan, B. Benatallah, S. Dustdar, and M. P. Papazoglou, "Cloud resource orchestration programming: overview, issues, and directions," *IEEE Internet Computing*, vol. 19, no. 5, pp. 46–56, 2015.

[28] F. Fakhfakh, H. H. Kacem, and A. H. Kacem, "Workflow scheduling in cloud computing: a survey," in *Proceedings of the 2014 IEEE 18th International Enterprise Distributed Object Computing Conference Workshops and Demonstrations (EDOCW)*, pp. 372–378, IEEE, Ulm, Germany, September 2014.

[29] H. Xiong, C. Filelis-Papadopoulos, D. Dong, G. G. Castañé, and J. P. Morrison, "Towards a scalable and adaptable resource allocation framework in cloud environments," in *Proceedings of the 2017 46th International Conference on Parallel Processing Workshops (ICPPW)*, pp. 137–144, IEEE, Bristol, UK, August 2017.

[30] K. M. Maiyama, D. D. Kouvatsos, B. Mohammed, M. Kiran, and M. A. Kamala, "Performance modelling and analysis of an OpenStack IaaS cloud computing platform," in *Proceeding of the 2017 IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud)*, pp. 198–205, IEEE, Prague, Czech Republic, August 2017.

[31] J. Chase and D. Niyato, "Joint optimization of resource provisioning in cloud computing," *IEEE Transactions on Services Computing*, vol. 10, no. 3, pp. 396–409, 2017.

[32] G. Li, S. Xu, J. Wu, and H. Ding, "Resource scheduling based on improved spectral clustering algorithm in edge computing," *Scientific Programming*, vol. 2018, Article ID 6860359, 13 pages, 2018.

[33] S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," *Future Generation Computer Systems*, vol. 28, no. 1, pp. 155–162, 2012.

[34] M. Borkowski, S. Schulte, and C. Hochreiner, "Predicting cloud resource utilization," in *Proceedings of the 2016 IEEE/ACM 9th International Conference on Utility and Cloud Computing (UCC)*, pp. 37–42, IEEE, Shanghai, China, December 2016.

[35] C. Fehling, F. Leymann, R. Retter, P. Arbitter, and W. Schupeck, *Cloud Computing Patterns: Fundamentals to Design, Build, and Manage Cloud Applications*, Springer Science & Business Media, Berlin, Germany, 2014.

[36] B. A. Wolfe, B. W. Mamlin, P. G. Biondich et al., "The OpenMRS system: collaborating toward an open source EMR for developing countries," in *Proceedings of the AMIA Annual Symposium Proceedings*, vol. 2006, American Medical Informatics Association, Washington, DC, USA, November 2006.

[37] Nimble Streamer, 2018, https://wmspanel.com/nimble.

[38] OpenStack, 2018, http://www.openstack.org.

[39] Apache jMeter, 2018, http://jmeter.apache.org/.

[40] CenturyLink Price Calculator, 2018, https://www.ctl.io/estimator/.

[41] Google Price Calculator, 2018, https://cloud.google.com/products/calculator/.

[42] Amazon Price Calculator, 2018, https://calculator.s3.amazonaws.com/index.html.

[43] Alibaba Cloud Price Calculator, 2018, https://www.alibabacloud.com/pricing.

[44] Digital Ocean Price Calculator, 2018, https://www.digitalocean.com/pricing/.

[45] Azure Price Calculator, 2018, https://azure.microsoft.com/en-us/pricing/calculator/.

[46] Oracle Price Calculator, 2018, https://cloud.oracle.com/en_US/cost-estimator.

[47] I. Stupar and D. Huljenić, "Analyzing service resource usage profiles for optimization of cloud service execution cost," in *Proceedings of the IEEE EUROCON 2017-17th International Conference on Smart Technologies*, pp. 79–84, IEEE, Ohrid, Macedonia, July 2017.

[48] M. H. Kutner, C. J. Nachtsheim, J. Neter, and W. Li, *Applied Linear Statistical Models*, McGraw-Hill, Irwin, NY, USA, 2005.

[49] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical Learning with Sparsity: The Lasso and Generalizations*, CRC Press, Boca Raton, FL, USA, 2015.

[50] R. Tibshirani, B. Efron, T. Hastie, and I. Johnstone, "Least angle regression," *The Annals of Statistics*, vol. 32, no. 2, pp. 407–499, 2004.

[51] J. Friedman, T. Hastie, and R. Tibshirani, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, Springer Series in Statistics, Berlin, Germany, Second edition, 2009.

[52] A. De Myttenaere, B. Golden, B. Le Grand, and F. Rossi, "Mean absolute percentage error for regression models," *Neurocomputing*, vol. 192, pp. 38–48, 2016.

[53] J. S. Bendat and A. G. Piersol, *Random Data: Analysis and Measurement Procedures*, Vol. 729, John Wiley & Sons, Hoboken, NJ, USA, 2011.

[54] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001.

[55] F. Pedregosa, G. Varoquaux, A. Gramfort et al., "Scikit-learn: machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[56] Py-Earth Library Documentation, 2019, https://contrib.scikit-learn.org/py-earth/.

[57] A. Gulli and S. Pal, *Deep Learning with Keras*, Packt Publishing Ltd, Birmingham, UK, 2017.

[58] B. Karlik and A. Vehbi Olgac, "Performance analysis of various activation functions in generalized MLP architectures of neural networks," *International Journal of Artificial Intelligence and Expert Systems*, vol. 1, no. 4, pp. 111–122, 2011.

[59] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, Sardinia, Italy, May 2010.

[60] V. Nair and G. E. Hinton, "Rectified linear units improve restricted Boltzmann machines," in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pp. 807–814, Haifa, Israel, June 2010.

[61] S. B. Green, "How many subjects does it take to do a regression analysis," *Multivariate Behavioral Research*, vol. 26, no. 3, pp. 499–510, 1991.

[62] ELI5 Library Documentation, 2019, https://eli5.readthedocs.io/en/latest/.