IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received May 8, 2021, accepted June 13, 2021, date of publication June 18, 2021, date of current version June 30, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3090366

Provisioning Computational Resources for Cloud-Based e-Learning Platforms Using Deep Learning Techniques

JORGE ARIZA, MIGUEL JIMENO, (Member, IEEE), RICARDO VILLANUEVA-POLANCO, AND JOSE CAPACHO, (Member, IEEE)
Department of Computer Science and Engineering, Universidad del Norte, Barranquilla 08001, Colombia
Corresponding author: Ricardo Villanueva-Polanco (rpolanco@uninorte.edu.co)

ABSTRACT The use of e-learning technologies is growing even faster due to the existing conditions where virtual setups temporarily replace traditional classroom environments. Service infrastructure support for e-learning has moved to the cloud. For this reason, the efficient provisioning of resources for such platforms, which is achieved through prediction, is very relevant. The existing techniques for predicting the use of resources in the cloud are not designed with e-learning's specific requirements. This paper presents a neural network-based model for predicting the usage of computational resources for e-learning platforms. This model consists of a series of interconnected neural networks used to predict values for variables of interest, such as Random Access Memory (RAM) usage and Central Processing Unit (CPU) usage. Using data collected from a high school real scenario, we analyzed and used it to train and validate our neural network-based model. This scenario consisted of a Moodle server deployed in a Google Virtual Machine with a configured course and its contents. Each student performed a series of activities while connected to it. Our proposed model achieves high accuracy. The obtained results are promising, paving the way towards constructing software tools for provisioning computational resources on demand for e-learning platforms.

INDEX TERMS Availability, cloud-based e-learning platforms, resource consumption, prediction, deep learning.

I. INTRODUCTION

The global e-learning market was worth $107 billion in 2015, and calculations expected it to grow to more than $300 billion by 2025 (Global Industry Analysts, 2020) [1]. There are more than $600 million e-learning students in India, China, and the U.S. There is a growth in the use of technologies involved in e-learning which include mobile technologies, digital content, and online learning sources and opportunities [2]. Also, with current worldwide conditions, these numbers are expected to grow faster than expected. Past studies in countries where e-learning is highly relevant such as India have found an increasing research interest in e-learning, which has focused mainly on Computer Science but also includes topics such as engineering, medicine, and social science [3]. Existing technologies used in e-learning are based on end-user technologies and server-based technologies. End-user technologies are the ones students use to access the platforms, such as mobile or desktop applications, and recently virtual and augmented technologies for hands-on content delivered remotely. Server-side technologies store the applications and data, where today, cloud-based services are dominant. Courses range from micro-lessons for mobile applications to Massive Open Online Courses [4]. An appropriate design of resource usage of the server-side services has become relevant due to the increasing interest in e-learning and the wide range of technologies used on both sides.

Resource provisioning on the cloud is a topic of study for several years, focusing mainly on predicting virtual machines resource usage in the cloud [5]. More specifically, current work deals with resource placement and consolidation, resource elasticity, and workload analysis and prediction. Most of the existing work focuses on workload prediction that studies historical cloud resources workload, and our work also does so. The workload data used for such studies comes mostly from virtual machine utilization against realistic job arrival statistics and sometimes from websites utilization.

The associate editor coordinating the review of this manuscript and approving it for publication was Yu-Da Lin.

We highlight the following aspects as critical problems for which this paper presents a solution. First, there is no previous study about the prediction of cloud resource consumption in e-learning systems. This kind of system has particularities that would make such a study pertinent. For example, the use of video streaming, and online testing, both of which would need different resource consumption requirements. Previous studies have used dataset focused on jobs running on virtual machines, which is very generic and with no specifications. Multiple studies use Google's traces, but although very comprehensive in terms of time and number of studied machines, there are no specific characteristics about the traced jobs that would make it easy to connect to an e-learning system. Second, there is no study in the literature about resource consumption from a well-known e-learning system such as Moodle. It is necessary to understand the behavior of such systems. Third, existing methodologies to monitor, predict and adjust cloud resource consumption lack an essential step to understanding the characteristics of the studied system. Such a step should be essential to design the monitor phase properly.

We next itemize all contributions of this paper:

- A cyclical methodology for characterizing cloud-based e-learning systems, predicting resource consumption, and applying the changes to the deployed resources
- An in-depth classification of e-learning systems based on some variables of interest
- A neural-network-based model to predict resource usage in e-learning systems
- A test of the model using real data from a Moodle e-learning system

This paper is organized as follows. In Section II, we first present the background work related to resource prediction in the cloud and resource provisioning methodologies. Additionally, we highlight the lack of previous work focusing on predicting resource consumption of e-learning systems. In Section III, we propose a general resource provisioning framework for the cloud that researchers could adapt to the studied system. In Section IV, we then present a characterization of e-learning systems. Later, in Section V, we present the monitoring and analysis components of the proposed monitoring framework, focusing on the consumption prediction for e-learning systems using real data collection. We then evaluate the proposed model in Section VI and finally present a discussion in Section VII and our conclusions in Section VIII.

II. BACKGROUND WORK

A. RESOURCE PREDICTION IN THE CLOUD

The flexibility characteristic of cloud computing and the pay-per-use model can lower the cost of use. However, maintaining service level agreements (SLAs) with end-users forces cloud customers to deal with cost/benefit trade-offs. The trade-offs benefit from calculating the minimum amount of resources that customers need to meet their SLA obligations. Also, cloud clients' workload varies over time. Therefore, providers must justify the exchange according to the workload. Providers design autoscaling systems to automatically balance the cost/benefit trade-off. There are two main classes of autoscaling systems in the infrastructure as a Service (IaaS) layer of cloud computing: reactive and predictive [6]. Reactive autoscaling systems are the most widely used. These systems scale for service according to their performance. Among the different algorithms used for predicting resources in the cloud, we have those based on regression [7]. These algorithms provide an optimal threshold range for the operation; therefore, the SLA requirement is achieved in most cases. The standard error in the prediction is low, and this demonstrates the effectiveness of the method. The classification-based techniques obtain a better performance [8]. Some classification methods include SVM, Randon Forest, and Neural Network. Other works mix approaches, for example, SVR and Kalman Filters. The algorithms' objective is to guarantee QoS, increase performance, and improve return on investment. These management and autoscaling approaches help providers provision and de-provision virtual resources. However, a poor prediction can cause under or over-provisioning problems, reducing cloud performance and leading to SLA violations and wasted resources [8].

B. ALGORITHMS FOR RESOURCE CONSUMPTION PREDICTION

There are many strategies for scheduling existing resources and providing new cloud resources, especially using prediction techniques. The authors of [9] presented an approach to predicting cloud resource utilization at the task resource level. They used ANN with a hidden layer, two input neurons, one output neuron, and ten hidden neurons. Their algorithm did not have a normalization technique to balance neurons' weights when new data was entered. In [10], they proposed estimating resources in the cloud based on QoS and Edge computing. To do this, they classify and compare the resources according to the similarity of the Euclidean distance. Then they used the gray matrix to correct the coincidence function. The next step used the Markov regression prediction method to analyze the available resources' state change and select the appropriate resource.

In [11], the authors try to avoid the overload of physical machines using automated resource management to allocate them dynamically. They developed an algorithm to predict the future load of each virtual machine. For this, they used EWMA to predict the CPU load on the server. The authors of [12] used a Kalman-filter-based algorithm and an Adaptive Neuro-Fuzzy Inference System (ANFIS). They obtained specific loads and eliminated the observation error. The results obtained were compared with the ANFIS and ARIMA algorithms with better outcomes. In [13], the authors used a prediction system based on Functional Link Neural Networks (FLNN). They used a genetic algorithm to train the model and thus increase the effectiveness of the forecast. The system allowed multivariate data, and to test the model, they used Google tracking data. The results show improvements when

compared to traditional techniques. The authors of [14] proposed an autoscaling algorithm based on dynamic thresholds that they predict using Long Short-Term Memory Recurrent Neural Network (LSTM-RNN) and autoscaling facilities based on predicted values. The experiments carried out show substantial improvements in prediction compared to other algorithms. In [15], they proposed a resource provisioning method based on PaaS data processing environments' utilization rate. They used a clustering algorithm to dynamically estimate and schedule batch applications' workload based on current and historical logs. The algorithm allows efficient scheduling decisions. Authors achieved an increment of 10% CPU usage and 20% RAM usage compared to static management systems without significantly reducing service quality. In [16], the authors propose a resource selection algorithm in fog computing (FResS) that allows automatic selection and allocation for IoT systems. The proposed model in the article maintains a repository of performance data in the form of execution records. When a new task starts execution, the system predicts its execution time through these records, which allows calculating an estimate. The results show an improvement in the end-to-end latency time of the system. The work in [17] presents a workload predictor for Edge Data Centers (EDCs). The predictor takes advantage of the similarity between EDC workloads at a close physical distance. It then applies a multivariate LSTM network to achieve workload predictions for each EDC. Another work proposed an energy-efficient virtual machine distribution scheme that reduces communication and energy consumption costs. It used an improved ant colony optimization approach with self-adaptive parameters for fast convergence and high search capacity [18]. The simulations showed improvements in power consumption and communication costs compared to other existing algorithms. The work in [19] proposed a technique that uses the Savitzky-Golay (SG) filter to eliminate extreme points and noise, combined with the LSTM algorithm to predict tasks in the following time interval; the Adam optimizer is used to train the model. The results show better results compared to some commonly used prediction algorithms.

C. RESOURCE PROVISIONING METHODOLOGIES

Authors in [20] proposed an autonomous control loop called MAPE (Monitor Analyze Plan Execute). Managed elements represent any software or hardware element in the cloud (CPU, operative system, application in the cloud, service in the cloud, storage, or VM). The sensors collect information from the environment. The Monitor step collects and filters the data collected by the sensors, analyzes, and compares the performance against its objectives. The Plan step determines corrective actions to analyze. Finally, the Execute step executes the reconfiguration, while the Effectors step applies the changes. The proposal of our methodology derives from the original MAPE loop and introduces new steps.

To coordinate the adaptation of services that make up the monitor element of SAS (self-adaptive systems), the authors of [21] proposed an extension of the MAPE-k architecture. They added a second cycle in the upper part of the Monitor element. This cycle manages the monitoring service adaptation process. This design decouples the SAS monitoring services' adaptation logic (Self-Adaptive Systems), allowing the modules' independent development. The work in [22] proposed a self-adaptive software architecture called Proteus. The main objective is to provide a context-sensitive solution for specific quality attributes (availability and reliability). The authors proposed to use it in dynamic systems applying replication and auto-configuration techniques. This architecture consists of three components, which replicate in the nodes that make up the system network, (1) monitoring subsystem, (2) management service, and (3) context management service. Our proposal in this paper works from the original MAPE and proposes new steps that enrich the original to allow the characterization of understudied systems such as e-learning systems. These systems are not well studied from the perspective of resource consumption monitoring and prediction.

III. NEW CLOUD RESOURCES PROVISIONING FRAMEWORK

In this section, we present a framework for provisioning cloud resources for any kind of application. The framework builds from a new version of the MAPE loop that includes new loop steps. The loop is a theoretical proposal of how to monitor and automatically control cloud applications. We show first the new loop and propose the architectural framework that implements it.

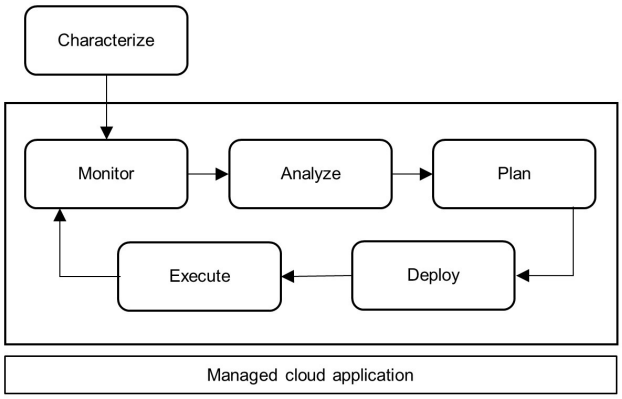

FIGURE 1. Resource monitoring cycle.

A. RESOURCE MONITORING METHODOLOGY

We first explain the methodology which we show in this section. This methodology works as follows, and Figure 1 shows it:

- Characterize: In this step, researchers characterize the type of systems they will study. The step consists of describing the applications or information systems of interest in terms of the kinds of offered services, expected users, and other variables. The result of this

step should be any taxonomy of the studied applications or systems that fall into the studied category. In this paper, we will apply the methodology to e-learning systems.

- Monitor: Once the studied system's characterization is done, researchers will select a list of variables to monitor. These variables of interest represent two aspects of resource consumption: 1) the available computational resources for the running applications, and 2) the communication links at disposal to those applications. Now, monitoring tools could be cumbersome, depending on the accessibility of the studied system. The ideal situation is when system architects can decouple the monitoring process by watching different modules simultaneously. Otherwise, they would have to monitor the system as a whole, and it might not be helpful if different components of the information system have different resource consumption requirements. There is enough literature about where and how to monitor cloud systems, as shown in [23]. A set of continuously monitored variables should generate a dataset that researchers will use in the following phases.
- Analyze: Researchers will now use the data obtained from the monitoring step using appropriate models. These models will predict the behavior of the platform. The works from [24] and [25] are helpful to select the type of algorithms. The first work presents a survey of algorithms used in the literature that perform workload analysis and prediction, based on [25]. That article discussed distinct workload features, characteristics, and which models are more convenient than others. Researchers must consider the studied system's characterization and the monitored variables to select the suitable model.
- Plan: Considering the data analyzed in the previous step, researchers determine the actions to take. The previous step's results might influence workload prediction and virtual machines and container placement and consolidation. According to [24], there is extensive literature about the impact that component placement across the edge-cloud data centers has on the overall system's performance. Architects and researchers could use optimization algorithms to plan the most suitable placement. This paper, however, focuses on workload prediction and how the plans deriving from the monitoring phase decide the number of available resources, but researchers could extend it to component placement or offloading and server consolidation.
- Deploy: Apply the changes and deploy the solution generated in the previous step, selecting the appropriate resources. Depending on the designed plan, the deployment might imply different locations, types of datacenters (edge and cloud), and a higher or lower number of nodes to deploy. Researchers should also consider the deployment costs using optimization algorithms in a mix of the plan and deploy phases.
- Run: This step generates the new system reconfiguration of the virtual machines or containers, depending on the system configuration.

This new loop introduces the Deploy step and differentiates it from the Run step in the original MAPE loop. The new Deploy step considers, when necessary, which cloud resources to deploy. The system should smartly select the resource to deploy, depending on the providers' quality of service. The system characterization is a step we are introducing but is not a looping component of the methodology. The idea of the modified methodology is that researchers and system architects could monitor any system, regardless of previous knowledge about the resource consumption characterization. In this paper, we differentiate e-learning systems because there is no previous work for resource consumption prediction in these systems. Other researchers could select another type with a particular behavior, such as eBroker systems, IoT-based systems, or vehicle network systems, and apply the proposed methodology. A characterization helps determine which subtypes are the most common, around which researchers will create the monitoring architecture.

B. MONITORING FRAMEWORK ARCHITECTURE

Figure 2 shows the architecture of a monitoring system implementation that uses the proposed methodology. The architecture is explained in general terms to monitor any application, and it is inspired by the architecture presented in [26]. On top of the figure, we have users accessing the application through mobile and web clients. Clients access the information system architecture in the cloud service provider, and it is worth depicting a typical tiered design with the following components to clarify how the monitoring architecture deals with each tier:

- Load balancers: they help balance the load between the servers when multiple instances are available. A monitoring architecture should also consider their burden, although they do not represent a costly tier in the cloud service architecture.
- Web servers: These servers compose the first tier to attend and solve requests, usually through simple APIs that the proposed architecture needs to monitor.
- Application servers: many cloud service designs distribute functionalities among microservices. Microservices spread the application's logic into simple, indivisible services [27], [28]. Microservices allow easy scalability and simple integration for easy updates of only the necessary components. Containerization is today used to deploy microservices, which enables lightweight servers with high portability [29]. The architecture needs to take into account all these aspects to allow smooth scheduling of the services.
- Database servers: usually the last tier of the architecture, implemented using dedicated servers for exposing data to the other servers. They are subject to monitoring to predict changes in consumption when the load increases.

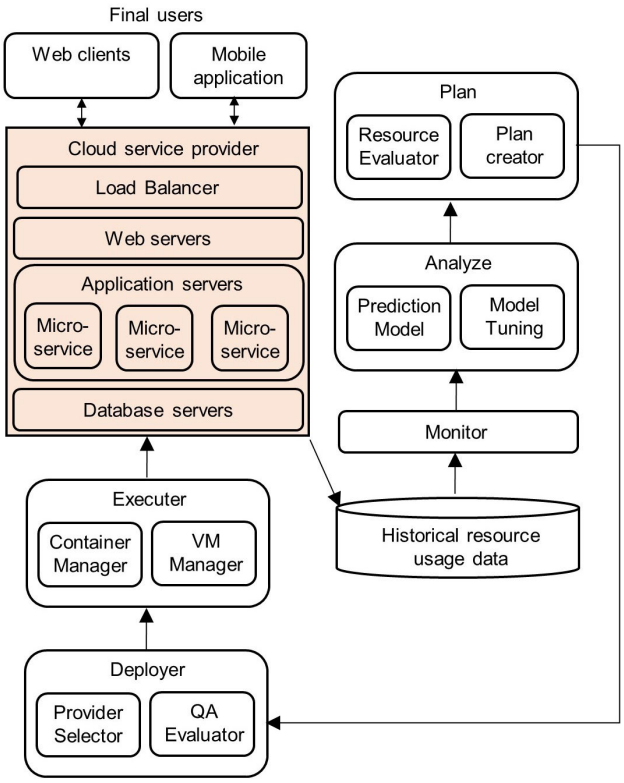

FIGURE 2. Architecture of the monitoring framework.

We design the architecture to monitor all of these components. We simplify the implementation and evaluation of the architecture in the following sections by monitoring servers on only one level of the service provider architecture. Figure 2 also shows the monitoring methodology components that compose the loop part, depicted as boxes for each step of the methodology. The characterization step is the first step and researchers must perform it before entering the monitoring loop. We explain now the system components that implement the methodology in the architecture:

1) Monitor component: it feeds from a dataset generated in real-time with cloud resources consumption historical data. The arrow that exits the cloud service provider box brings the real-time measurement of the selected variables, such as CPU, RAM, number of users, network utilization, disk usage, among other variables. This monitored data may be useful to identify incidents that deviate from a dataset's normal behavior, which may indicate critical incidents, such as a technical malfunction of a component or denial of service attacks [30].
2) Analyze components: here is where the prediction model implementation lies. It feeds from the monitored data to predict new values. And the model tuning element refines the model every time usage information arrives. This component outputs a plan that consists of the predicted number of machines (either virtual or containers) necessary to adapt to resource demand changes.
3) Plan components: This component aims to filter the scheduling plan for a posterior evaluation, and it is composed of two subcomponents. The resource evaluator determines the availability of the requested resources. The plan creator then creates the final version of the scheduling plan
4) Deployer components: this important component evaluates the most convenient deployment configuration of containers or virtual machines. The Quality Assurance (QA) Evaluator measures quality variables from the cloud service providers, such as response times, costs of available resources, network connection quality, among other variables. An optimization algorithm could combine the results from the QA Evaluator and optimize the component's outcome to select the most convenient configuration. The Provider Selector subcomponent uses the previous subcomponent result to prepare resources, which might imply restarting dormant services or moving configuration data.
5) Executer components: the architecture needs either a Virtual Machine (VM) or Container Manager. They will be in charge of executing the plan and making the configurations requested by the previous component. These managers are in direct connection with the servers and determine which instances will be initiated or terminated.

The components create a cycle by finally starting or stopping services in the cloud service providers. Microservices implemented in containers facilitate resource usage scheduling plans, given that restarting or stopping containers is much faster than with virtual machines. This architecture assumes a large deployment of services across multiple containers or virtual machines and the possibility of evaluating even several cloud providers for an optimal selection of resources.

IV. CHARACTERIZATION OF E-LEARNING SYSTEMS

This section aims at characterizing the studied system, in our case, an e-Learning system. The use of technological tools in educational settings began several decades ago and has evolved like technology. This constant evolution has raised the need to evaluate systems from different perspectives. Said perspectives had focused mainly on the educational objectives of these tools and their usability [31], [32]. However, there is no detailed monitoring of the resource consumption of these services. This step is essential because e-learning tools continuously migrate to the cloud [33] as software companies offer their services to the cloud. Educational organizations are using the cloud to store their educational tools for all the benefits it provides. In this section, we are creating a characterization of the e-learning systems available now. We made a list of systems with no intention of being exhaustive, given that a complete list would be too extensive. Instead, the idea was to list examples of the popular e-learning system types and determine which one might be the most popular. The final objective is to select a type from

which the prediction model will calculate future resource consumptions.

A. DEFINITIONS

The following is a list of essential definitions. We are not defining all terms for space reasons, but only those that could help clarify some topics.

- Learning Content Management System (LCMS): These systems allow users to create and manage the learning content, which enables users to acquire only one tool, but at the same time, they centralize the service demand on one tool. They are not as common as LMS, however.
- Learning Management System (LMS): This is the most common category. For example, Google Classroom classifies as an LMS. Google integrates it with its multiple set of tools for content creation, project management, and scheduling. However, those are tools outside Google Classroom.
- Student Information System (SIS): these are systems that schools use to manage students' information. They are not directly related to the teaching process, but to keep grading, transcripts, registering, attendance, among other functionalities.
- Gamified learning tools: This is a type of software for teaching students using a gamification methodology.
- Massive Open Online Course (MOOC): These are courses that target the largest audiences, well beyond what schools usually target. Architects need to design these platforms for large audiences, where thousands might take their classes. Given such expectations, they need to invest more in infrastructure and not have resource availability problems that other systems encounter.

B. VARIABLES OF INTEREST

The following is a list of variables that will help classify e-learning systems. The idea is not a classification based on the tools' educational perspectives but from the point of view of computational resources consumption. The front-end perspective considers the variables that researchers can evaluate in the front-end of the e-learning system. The variables are the following:

1) TYPE OF END-USER CLIENT

Software companies develop some e-learning systems with only a mobile end-user in mind. This client removes some of the cloud server's computational load because a mobile application will perform some of the end user's tasks. On the other hand, some companies develop their e-learning systems mainly for web clients. Although many of them offer mobile clients, most of their users connect through web clients. The load of the servers in the cloud will be larger in these cases ref.

2) TYPE OF CONTENT

From a back-end perspective, the variables taken into account evaluate the code located on the cloud servers. The functionalities offered by the e-learning system may differ depending on the size or specialization of the software. The following are the typical types:

- Real-time video and audio sessions
- Content creation and sharing
- Tests and activities
- Bidirectional asynchronous interaction

To classify a type of content, we should consider also computational requirements of the offered services, number of users and expected usage time for that service. From these characteristics, we will proceed now to list and classify several well-known e-learning systems and applications.

C. SYSTEMS CLASSIFICATION

Table 1 shows the list of evaluated e-learning systems using the previously defined variables of interest. Most of them offer web interfaces for their users, while some of them also use mobile applications. The video conference capabilities for e-learning are helpful. However, it is uncommon to find systems that offer video on top of all the other features. Most of them rely on third-party services for video, which frees computational resources on their cloud providers.

Figure 3 shows the distribution of system types, being the LMS type the most common. As a conclusion of this characterization, we can see that the monitoring and data analysis of an LMS system would be reasonably representative of e-learning systems. Therefore, we select Moodle as a candidate for applying the proposed control loop and the framework implemented in this paper. The same approach followed here applies to other LMS systems, for which other researchers should get similar results.

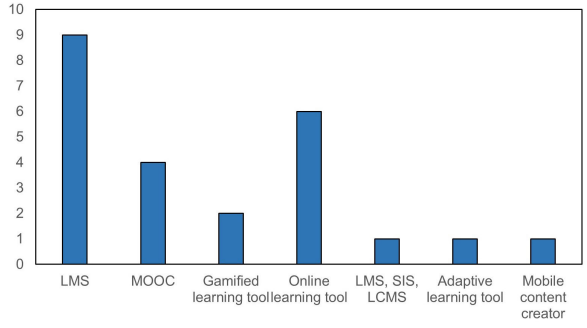

FIGURE 3. Distribution of the examined e-learning systems.

V. MONITORING AND ANALYSIS OF THE INFORMATION SYSTEM

In this section, we present the monitoring and analysis components of the monitoring framework proposed previously. First, the monitoring component is based on a setup for data collection from an e-learning cloud service about the variables of interest. We then present the analysis component with a neural network-based model for predicting the usage of computational resources for e-learning platforms.

TABLE 1. List of e-learning systems.

E-learning system	Description	System type	Client types	Content types
Blackboard [34]	Online classroom creation	LMS	web, Mobile	Surveys, audio, video, documents
Canvas [35]	Online classroom creation	LMS	Web, Mobile	Surveys, documents, message boards, audio, video
Chamilo [36]	Online classroom creation	LMS	Web, Mobile	Surveys, documents, message boards
Coursera [37]	MOOC provider	MOOC	Web, Mobile	Surveys, documents, message boards, audio, video
Duolingo [38]	Language learning tool	Gamified learning tool	Web, Mobile	Audio, Video, Documents
Edmodo [39]	Online classroom creation	LMS	Web, Mobile	Surveys, documents, message boards
edX [40]	MOOC provider	MOOC	Web, Mobile	Surveys, documents, message boards, audio, video
Evolcampus [41]	Online classroom creation	LMS	Web	Surveys, documents, message boards, audio, video
GeoGebra [42]	Online Math resources	Online learning tool	Web, Mobile	documents
Google classroom [43]	Online classroom creation	LMS	Web, Mobile	Surveys, documents, message boards
IXL [44]	Online science and arts learning tool	Online learning tool	Web, Mobile	Surveys, documents, message boards
Khan Academy [45]	Video lessons provider	Online learning tool	Web, Mobile	documents, surveys, video
Knewton [46]	Online personalized study application	Adaptive learning tool	Web	Surveys, documents, message boards, video
Kornukopia [47]	Online classroom creation and administration	LMS, SIS, LCMS	Web	Surveys, documents, message boards
Moodle [48]	Online classroom creation	LMS	Web, Mobile	Surveys, documents, message boards
NEO LMS [49]	Online classroom creation	LMS	Web, Mobile	Surveys, documents, message boards
Quizlet [50]	Online study application	Gamified learning tool	Web, Mobile	Surveys and tests
Schoology [51]	Online classroom creation	LMS	Web, Mobile	Surveys, documents, message boards
Scratch [52]	Online programming learning tool	Online learning tool	Web, Mobile	Audio, Video
studyX [53]	Educational mobile applications creator	Mobile content creator	Web, Mobile	Surveys, documents, message boards, video
Udemy [54]	MOOC provider	MOOC	Web, Mobile	Surveys, documents, message boards, audio, video
Udacity [55]	MOOC provider	MOOC	Web	Surveys, documents, message boards, audio, video
web assign [56]	Online homework application	Online learning tool	Web	Surveys, documents
Wolfram Alpha [57]	Online science applications tool	Online learning tool	Web, Mobile	Videos, audios, surveys

For the monitoring component, we describe the methodology to gather our dataset and describe the dataset structure. We then describe our methods for data pre-processing and present our neural-network-based model for predicting the usage of computational resources for the studied system.

A. MONITORING DATASET COLLECTION

In order to collect our data, we setup a Google cloud platform (GCP) account and installed and deployed a cloud-based Moodle server (version 3.6.8). On this Moodle server, we created six courses, with each having conferencing rooms, questionnaires, activities such as assignments, forums, and resources for the students to read online. The number of participants was 234, and each participant was registered as a student to one of the courses. A course had 39 students on average. During two weeks, we collected data using GCP software tools generated by the activities performed by the students. The trace resulted in a dataset containing 17281 registers divided into 10 fields. Table 2 describes the structure of the dataset, i.e., each field within a register, field data type, unit, its minimum value, and maximum value.

B. DATA PREPROCESSING

To improve the quality of our data, we first remove registers where NaN values were found and additionally apply a linear transformation to the variables X_3, X_4, X_5, X_6, X_7 by using the method Min-Max Scaler [58]. This transformation method takes values $x_1, x_2, \ldots, x_m$ and transforms them to lie in a given interval $[min, max]$. Basically it calculates the maximum x_{max} and the minimum x_{min} from the given values, and then transforms x_i to $y_i = \frac{x_i - x_{min}}{x_{max} - x_{min}}(max - min) + min$.

TABLE 2. Description of data fields.

Identifier	Field name	Data type	Description	Range		Unit
				Min	Max	
	Date	Datetime	Register date	26/10/2020	11/10/2020	
	Hour	Datetime	Register Hour	00:00:00	23:59:59	
X_1	num_event	Int64	Number of events	1	524	
X_2	user_live	Int64	Number of connected users	0	104	
X_3	disk_write	Float64	Amount of disk-written data	139.01	31584.62	Bytes
X_4	ram_usage	Float64	RAM usage	149	2600.15	Megabytes
X_5	disk_read	Float64	Amount of disk-read data	0	2904.60	Bytes
X_6	Instance_network	Float64	Amount of network-sent data	295.08	207762.17	Bytes
X_7	Instance_network_rec	Float64	Amount of network-received data	25.15	207965.80	Bytes
X_8	cpu_usage	Float64	Percentage of CPU usage	0.1507	0.475	

This is applied to the mentioned features to transform their corresponding values to lie in the [0, 1] interval.

C. NEURAL-NETOWORK-BASED MODEL

In this subsection, we introduce a neural network architecture for predicting the usage of computational resources. Our architecture consists of two components: The first component of this architecture consists of two recurrent neural networks (RNNs) with Long Short-Term Memory (LSTM) units used to predict the number of events (X_1) and the number of connected users (X_2) per minute respectively. The second component consists of interconnected Fully Connected (FC) neural networks for predicting various variables representing computational resources.

We first describe the two RNNs that constitute the first component. We use two recurrent networks with LSTM units to predict the variables *num_event* and *user_live*. In particular, we see the problem of predicting these variables as a sequence modeling problem [58].

The first RNN, denoted as $\mathcal{N}_1$, consists of several stacked layers, and its goal is to predict the number of events (X_1) per minute. Let `bs` denote the batch size and `sl` denote the sequence length. It has the following stacked layers: a conv1D layer, three stacked bidirectional RNN layers with LSTM units, three stacked dense layers, and a custom layer. The conv1D layer extracts features from the input sequence (of length 100 during training). Its configuration sets the number of filters to 128, denoted as `f`, the kernel size to 5, the number of strides to 1, padding to *causal*, and the activation function to the rectified linear unit (ReLU) function [58]. This conv1D layer receives as input a three dimensional array of size `bs` $\times$ `sl` $\times$ 1 and outputs a three dimensional array with shape `bs` $\times$ `sl` $\times$ `f` that is fed to three stacked bidirectional layers having 128, 60, and 60 LSTM units respectively. Each layer returns its full output to the next layer, i.e., an three-dimensional array of shape `bs` $\times$ `sl` $\times (2 \cdot$ `nu`$)$, where `nu` denotes the number of units. The last output is next fed to three stacked dense layers with 30, 10, and one unit(s) respectively, through which the three dimensional array input is transformed to a final three-dimensional output of shape `bs` $\times$ `sl` $\times$ 1. Each unit within the dense layers employs the ReLU function as activation function. Finally, the custom layer linearly transforms the entries of the last dense layer's output.

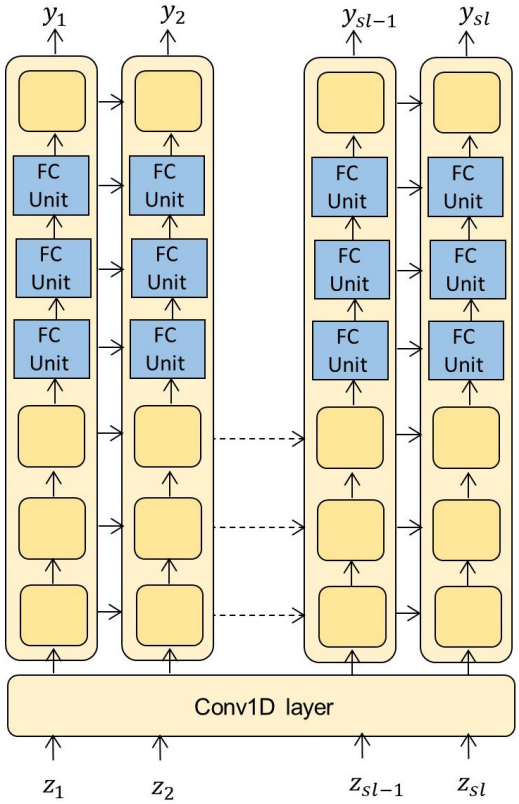

FIGURE 4. The second RNN of the first component of the neural network architecture.

The second RNN, denoted as $\mathcal{N}_2$, is shown in Figure 4. It is architecturally similar to the previous one. It consists of several layers, and its goal is to predict the number of connected users (X_2) per minute. In particular, this second network has the following stacked layers: a conv1D layer, three stacked RNN layers with LSTM units, three stacked dense layers, and a custom layer. The conv1D layer extracts features from the input sequence (of length 100 during training). Its configuration sets the number of filters to 128, denoted as `f`, the kernel size to 5, the number of strides to 1, padding to *causal*, and the activation function to the rectified linear unit (ReLU) function [58]. This conv1D layer receives as input a three dimensional array of size `bs` $\times$ `sl` $\times$ 1 and outputs a three dimensional array with shape `bs` $\times$ `sl` $\times$ `f` that is fed to three stacked bidirectional layers having 128, 60, and

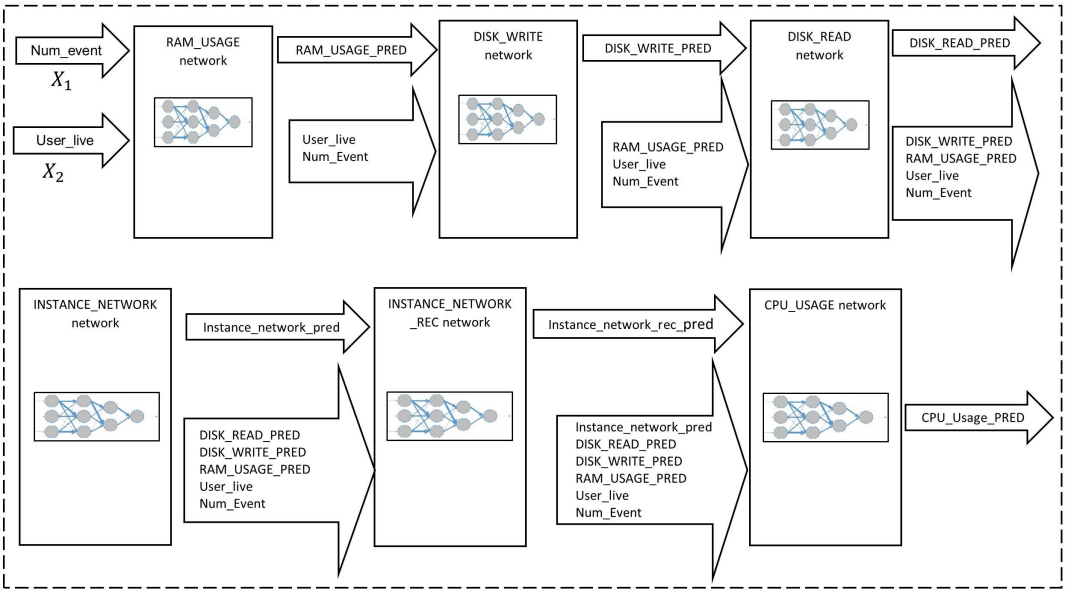

FIGURE 5. The second component of the neural network architecture.

60 LSTM units respectively. Each layer returns its full output to the next layer, i.e., an three-dimensional array of shape $\texttt{bs} \times \texttt{sl} \times \texttt{nu}$, where nu denotes the number of units. The last output is next fed to three stacked dense layers with 30, 10, and one unit(s) respectively, through which the three dimensional array input is transformed to a final three-dimensional output of shape $\texttt{bs} \times \texttt{sl} \times 1$. Each unit within the dense layers employs the ReLU function as the activation function. Finally, the custom layer linearly transforms the entries of the last dense layer's output.

These networks were tuned manually after testing with different possible configurations. Additionally, we use Huber Loss (MSE) as loss function [58] and Adaptive Moment Estimation (ADAM) algorithm as an optimization algorithm during the training of each recurrent neural network since these hyper-parameters are commonly used in prediction/regression settings.

We now describe the second component of our neural network architecture. It consists of a series of interconnected FC neural networks (a box as shown by Figure 5). The first FC neural network, denoted by $\mathcal{N}_3$, is fed the number of events (X_1) and the number of connected users (X_2) per minute (Output by $\mathcal{N}_1$ and $\mathcal{N}_2$ respectively) and returns the amount of disk-written data in bytes per minute. The value predicted by this first neural network and the number of events and the number of connected users is passed to the next neural network to predict ram usage in megabytes. This prediction, along with the other three values, are passed to the next neural network, $\mathcal{N}_4$, to obtain the amount of read-written data in bytes. This algorithm continues until the CPU usage variable is predicted. The reason for this particular order to make predictions lies in the correlation matrix shown in Figure 6. If we refer to the two independent variables *user_live* and *num_event*, the correlation matrix suggests

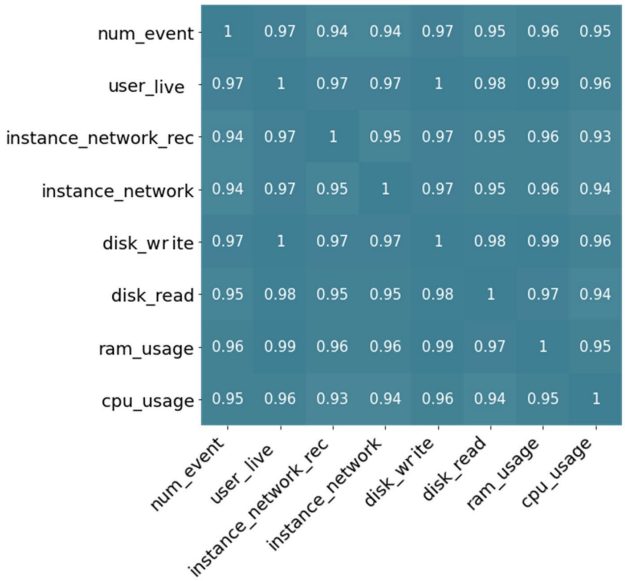

FIGURE 6. Correlation matrix.

a way to set an order for prediction based on correlation. In summary, the variables are predicted in the following order *disk_write*, *ram_usage*, *disk_read*, *instance_network*, *instance_network_rec* and finally *cpu_usage*.

Let us denote $\mathcal{N}_i$ to be the FC neural network for $i = 3, 4, \ldots, 8$. Each $\mathcal{N}_i$ receives $i-1$ input variables, i.e. the values from $X_1, X_2, \mathcal{N}_3(X_1, X_2), \ldots, \mathcal{N}_{i-1}(X_1, \ldots, \mathcal{N}_{i-2}(\ldots))$, where the first two are the variables *user_live* and *num_event* and the remaining $i-3$ values are the previous predictions. Additionally, each FC neural network has four hidden layers with 256, 128, 64 and 32 units respectively and a output layer that consists of only one unit. Each unit within the hidden

layers employs the ReLU function as activation function, while the single unit of the output layer employs the identity function as activation function.

The number of units for each hidden layer was chosen after running a grid-search-like tuning algorithm as shown by Algorithm 1. This algorithm constructs FC neural networks with $m \in \{3, 4\}$ hidden layers with each having a varying number of units, trains them for a fixed number of epochs and validates them, and then computes their corresponding losses, choosing the configuration giving the minimum loss value. After running it, we found that $N_u = [i-1, 256, 128, 64, 32, 2]$ for $i \in \{3, 4, 5, 6, 7, 8\}$. Figure 7 shows the FC neural network $\mathcal{N}_3$.

Algorithm 1 Tunes Some Hyper-Parameters for FC Neural Networks

```
function TUNINGFC(i, α, e, TrainingSet, ValidationSet)
    min_loss ← ∞
    N_min = []
    for m ∈ {3, 4} do
        for j = 1 to 4 do
            Create a FC neural network N_i with m + 2 layers
            Set the number of input units for the layer 1 to i − 1
            N_u ← [i − 1]
            n_u ← 2^(m+j)
            for l = 2 to m + 1 do
                Set the number of units for the layer l to n_u
                Set ReLU as activation function for each unit
                N_u.append(n_u)
                n_u ← n_u/2
            end for
            Set the number of units for the layer m + 2 to 1
            Set identity function as activation function for the unit.
            N_u.append(1)
            Set ADAM as optimizer with learning rate α.
            Set loss function as MSE.
            Train N_i with TrainingSet for e epochs.
            Validate N_i with ValidationSet.
            loss ← N_i.loss()
            if loss < min_loss then
                N_min ← N_u
                min_loss ← loss
            end if
        end for
    end for
    return N_min
end function
```

We use mean square error (MSE) as loss function [58] and Adaptive Moment Estimation (ADAM) algorithm as an optimization algorithm during the training of each FC neural network, since these hyper-parameters are commonly used in prediction/regression settings.

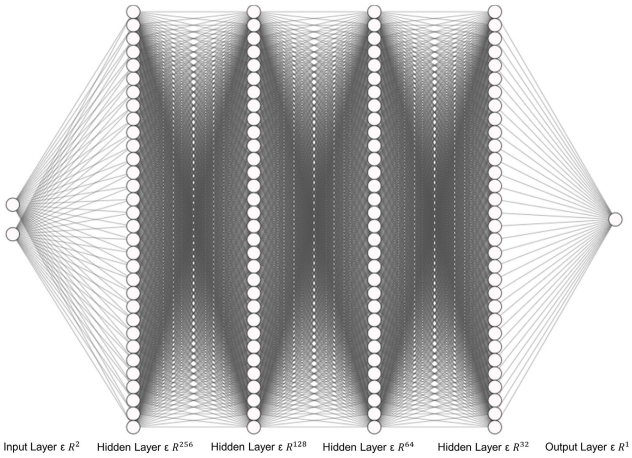

FIGURE 7. FC neural network $\mathcal{N}_3$.

VI. EVALUATION

In this section, we will present several scenarios designed to evaluate our model.

A. FIRST SCENARIO

In this scenario, we describe the training and validation of each neural network independently. Recall that the dataset is divided into two parts: a training dataset consisting of 70% of the original dataset and a validation dataset consisting of the rest 30% of the original dataset.

We first described how we trained and validated the RNNs $\mathcal{N}_1$ and $\mathcal{N}_2$. Recall that these are used for predicting variables X_1 and X_2, respectively. Since we modeled predicting these variables as a sequence modeling problem, we used a sequence of length 100 during training and validation. In particular, for $\mathcal{N}_1$ we got a Huber loss of 1.4754 for training and about 1.9755 for validation. On the other hand, for $\mathcal{N}_2$, we got a Huber Loss of about 1.4411 for training and about 1.8755 for validation.

To train and validate our FC networks $\mathcal{N}_i$ for $i = 3, 4, \ldots 8$ we run Algorithm 1. After running it, we found each FC neural network should have four hidden layers with 256, 128, 64 and 32 units respectively. Table 3 shows the MSE losses obtained for training and validation.

On the other hand, Figures (a), (b), (c) and (d) of 8 show plots comparing the original values as stored in the validation dataset and the corresponding value predicted by the respective neural network.

B. SECOND SCENARIO

In this second scenario, we describe how our complete model is validated. Let us set $V = [V_1, V_2, V_3, \ldots, V_m]$, where $V_i = [v_{i,1}, v_{i,2}, \ldots, v_{i,8}]$ are the entries of our validation set sorted by time. Let $S_1 = [s_{1,1}, s_{1,2}, \ldots, s_{1,\mathrm{sl}}]$ be a sequence of values (ordered by minute) from X_1 preceding $v_{1,1}$, and

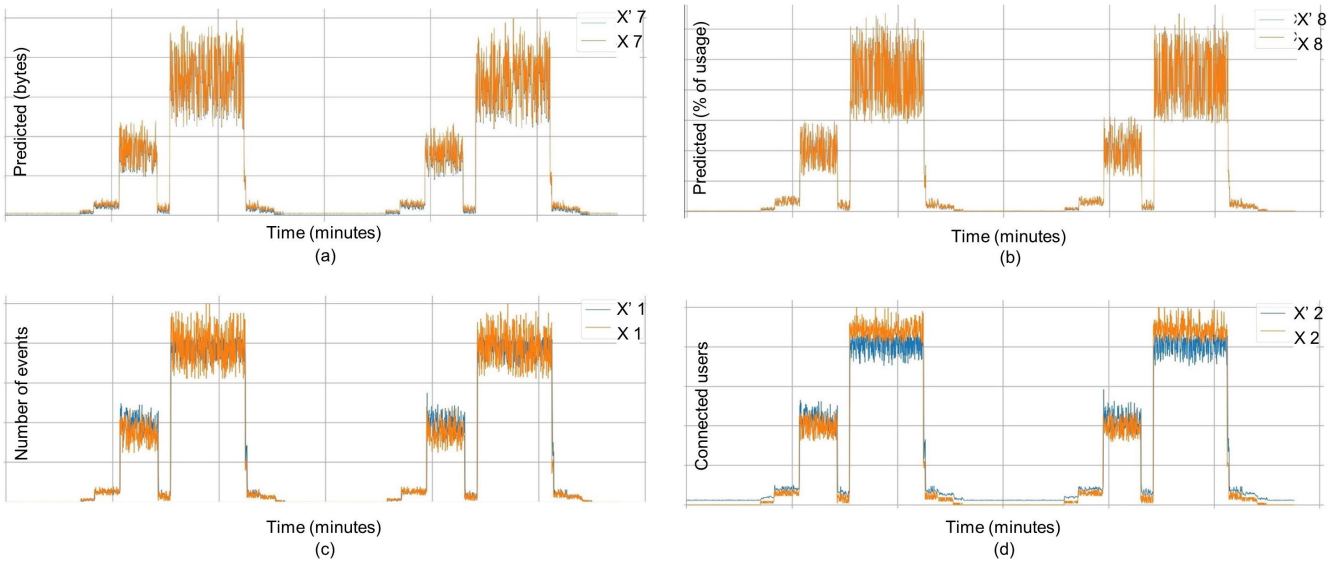

FIGURE 8. Comparison of the validation values with predicted values output by FC and RNN networks on input values from the validation dataset.

let $S_2 = [s_{2,1}, s_{2,2}, \ldots, s_{2,\mathtt{s1}}]$ be a sequence of values (ordered by minute) from X_2 preceding $v_{1,2}$. Algorithm 2 describes the process to validate our complete model.

Algorithm 2 Computes Measures Between the Validation Values and the Predicted Values by the Complete Model

function VALIDATECM(S_1, S_2, V)
  $\widehat{X} \leftarrow []$
  for $i \leftarrow 1$ to m **do**
    $\widehat{x_1} \leftarrow \mathcal{N}_2(S_1)$
    $\widehat{x_2} \leftarrow \mathcal{N}_2(S_2)$
    $\widehat{x_3} \leftarrow \mathcal{N}_3(\widehat{x_1}, \widehat{x_2})$
    $\widehat{x_4} \leftarrow \mathcal{N}_4(\widehat{x_1}, \widehat{x_2}, \widehat{x_3})$
    $\widehat{x_5} \leftarrow \mathcal{N}_5(\widehat{x_1}, \widehat{x_2}, \widehat{x_3}, \widehat{x_4})$
    $\widehat{x_6} \leftarrow \mathcal{N}_6(\widehat{x_1}, \widehat{x_2}, \widehat{x_3}, \widehat{x_4}, \widehat{x_5})$
    $\widehat{x_7} \leftarrow \mathcal{N}_7(\widehat{x_1}, \widehat{x_2}, \widehat{x_3}, \widehat{x_4}, \widehat{x_5}, \widehat{x_6})$
    $\widehat{x_8} \leftarrow \mathcal{N}_8(\widehat{x_1}, \widehat{x_2}, \widehat{x_3}, \widehat{x_4}, \widehat{x_5}, \widehat{x_6}, \widehat{x_7})$
    $\widehat{X}$.append($[\widehat{x_1}, \widehat{x_2}, \widehat{x_3}, \widehat{x_4}, \widehat{x_5}, \widehat{x_6}, \widehat{x_7}, \widehat{x_8}]$)
    S_1.remove(1) // remove the first entry of S_1.
    S_1.append($\widehat{x_1}$) // append $\widehat{x_1}$ to S_1.
    S_2.remove(1) // remove the first entry of S_1.
    S_2.append($\widehat{x_2}$) //append $\widehat{x_2}$ to S_2.
  end for
  $L = []$
  for $j \leftarrow 3$ to 8 **do**
    $I_1 = [V[i, j]$ for $i \leftarrow 1$ to $m]$;
    $I_2 = [\widehat{X}[i, j]$ for $i \leftarrow 1$ to $m]$;
    L.append([MSE(I_1, I_2), MAE(I_1, I_2)]);
    plot(I_1, I_2);
  end for
  return L
end function

Running Algorithm 2 computes two measures between (MAE and MSE) the validation values for the variable X_i and the corresponding values $\widehat{X_i}$ predicted by the complete model for $3 \leq i \leq 8$. In particular, Figure 9 shows plots comparing the original values for X_7 and X_8, as stored in the validation dataset, and the corresponding values $\widehat{X_7}$ and $\widehat{X_8}$ predicted by the complete model.

C. THIRD SCENARIO

In this third scenario, we compare the values predicted by our complete model with the actual values from our validation dataset and values predicted by computing a moving average with different window sizes. In particular, Figure 10 shows plots comparing the validation values for X_7 and X_8, the corresponding values $\widehat{X_7}$ and $\widehat{X_8}$ predicted by the complete model and the moving average with a window size of 5, 10, 20 data points.

VII. DISCUSSION

An important question is comparing the possibility of using Edge versus traditional cloud to deploy an e-learning system. Even the consideration of a hybrid architecture mixing cloud and edge services is possible. For example, some schools in developing countries such as Colombia might benefit from hybrid architectures to take advantage of existing in-house infrastructure to decrease deployment costs. This alternative, however, needs to be studied carefully to avoid paying extra unnecessarily. Another option is the use of public access points to deploy certain services from the e-learning systems. For example, people in rural underserved locations across Colombia widely use public access points. With the help of local and national education agencies, such deployments might be possible and decrease deployment costs. Such scenarios are beyond the scope of this paper but are very interesting for future research works.

Researchers can use the contribution of this paper to estimate the resource usage costs around which the architects

TABLE 3. MSE losses for training and validation.

Neural Network	Inputs	Output	Training MSE	Validation MSE
$\mathcal{N}_3$	X_1, X_2	X_3	$2,19 \times 10^{-3}$	$2,37 \times 10^{-3}$
$\mathcal{N}_4$	X_1, X_2, X_3	X_4	$7,34 \times 10^{-8}$	$6,88 \times 10^{-8}$
$\mathcal{N}_5$	X_1, X_2, X_3, X_4	X_5	$1,89 \times 10^{-8}$	$1,82 \times 10^{-8}$
$\mathcal{N}_6$	X_1, X_2, X_3, X_4, X_5	X_6	$3,57 \times 10^{-7}$	$3,99 \times 10 - 7$
$\mathcal{N}_7$	$X_1, X_2, X_3, X_4, X_5, X_6$	X_7	$2,22 \times 10^{-8}$	$2,86 \times 10^{-8}$
$\mathcal{N}_8$	$X_1, X_2, X_3, X_4, X_5, X_6, X_7$	X_8	$4,95 \times 10^{-8}$	$5,46 \times 10^{-8}$

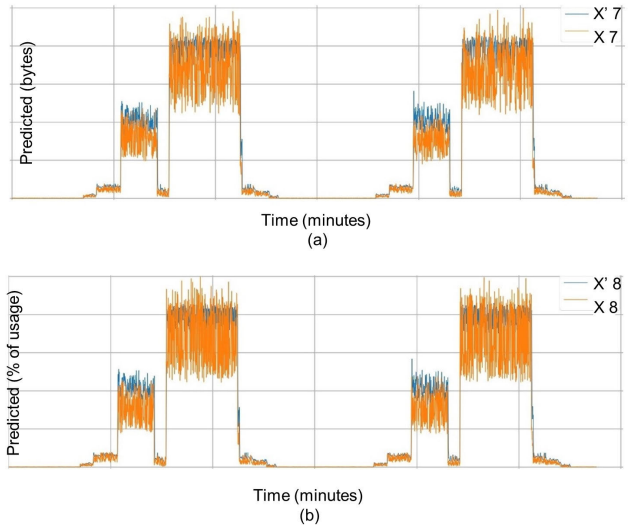

FIGURE 9. Comparison of the validation values and predicted values output by the complete model.

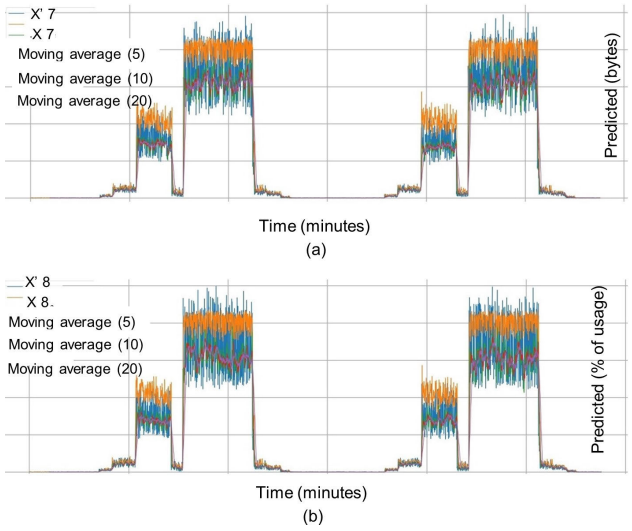

FIGURE 10. Comparison of the validation values with the predicted values output by the complete model and moving averages for various window sizes.

could design their deployments properly. Also, the prediction model may be extended to predict resources for the following days with a granularity of hours and even minutes. This prediction granularity helps turn on or off virtual machines' containers as necessary before the consumption peaks arrive to have just the necessary resources available. An appropriate granularity selection derives from the most cost-efficient consumption, and architects could optimize this towards reducing costs.

There are some points worth discussing in this work that might improve future researches. First, while Moodle is one of the most used platforms, studying the system under multiple scenarios might be instrumental. Such scenarios might include several ranges of concurrent users, several amounts and types of educational resources, and extended periods of trace time. The types of educational resources might imply different types of requests to the servers and different computational loads. Another aspect that might be considered a limitation is the available cloud resources for the study. We used Google's computational resources, although a combination of multiple cloud providers might be interesting to observe possible changes in the results. We consider, however, that this consideration did not affect our results.

VIII. CONCLUSION AND FUTURE WORK

This work proposed a new framework for monitoring, measuring, predicting, and deploying computational resources in the cloud for e-learning information systems. The framework builds from a resource monitoring cycle, an improved version of the MAPE monitoring tool from IBM. This monitoring cycle has the potential for applying it to multiple types of applications. The main differentiating feature is the first step, which characterizes the information system to be monitored. This step ensures that architects or researchers take a moment to describe clearly the system from a computational resource consumption perspective. This perspective allows them to compare systems with others for behavior similarities. Also, such similarities could help classify new systems from other systems, for which there are plenty of algorithms and collected data, e.g., traditional websites. Therefore, this characterization step might result in a new execution of the monitoring cycle where there is no closely relatable collected data. This is the case presented in this paper.

We achieved the implementation of the monitoring cycle using a framework deployed in the cloud. We implemented a small subset of the proposed framework components. A future contribution would be a detailed implementation that shows interesting features. For example, some issues exist regarding the difference in startup and stop times between containers and virtual machine deployments. Researchers should take these differences into account when running the executor component of the framework. An optimization algorithm could be used to balance the use of both platforms

and help architects decide which combination to use in case multiple nodes are necessary. Additionally, our model may be enhanced to be able to detect anomaly behaviors, which may help in identifying emergent cyber-attacks [30].

We devise the potential of the proposed framework to understand particular information systems in terms of resource consumption. The monitoring cycle should help researchers better grasp the studied system and decide which components to deploy with better information at hand. The literature also shows the different prediction algorithms that researchers can use depending on the variables available and the target in mind.

REFERENCES

[1] *E-Learning—Market Study by Global Industry Analysts, Inc*. Accessed: Feb. 26, 2021. [Online]. Available: https://www.strategyr.com/market-report-e-learning-forecasts-global-industry-analysts-inc.asp
[2] A. Brown and T. Green, "Issues and trends in instructional technology: Access to mobile technologies, digital content, and online learning opportunities continues as spending on IT remains steady," in *Educational Media and Technology Yearbook*, vol. 42, R. M. Branch, H. Lee, and S. S. Tseng, Eds. New York, NY, USA: Springer, 2019, pp. 3–12.
[3] S. Gupta and S. Pandey, "Mapping of research publication on elearning in India during 2009-2018: A scientometric study," *Library Philosophy Pract.*, vol. 2624, pp. 1–12, Sep. 2019.
[4] D. Ni and J. Lee, "An analysis of research trends in mobile learning through comparison between Korea and China," *Educ. Technol. Int.*, vol. 20, no. 2, pp. 169–194, 2019.
[5] J. Zhang, H. Huang, and X. Wang, "Resource provision algorithms in cloud computing: A survey," *J. Netw. Comput. Appl.*, vol. 64, pp. 23–42, Apr. 2016.
[6] A. Y. Nikravesh, S. A. Ajila, and C.-H. Lung, "Measuring prediction sensitivity of a cloud auto-scaling system," in *Proc. IEEE 38th Int. Comput. Softw. Appl. Conf. Workshops*, Jul. 2014, pp. 690–695.
[7] W. Zhong, Y. Zhuang, J. Sun, and J. Gu, "A load prediction model for cloud computing using PSO-based weighted wavelet support vector machine," *Appl. Intell.*, vol. 48, no. 11, pp. 4072–4083, 2018,
[8] M. Masdari and A. Khoshnevis, "A survey and classification of the workload forecasting methods in cloud computing," *Cluster Comput.*, vol. 23, no. 4, pp. 2399–2424, Dec. 2020.
[9] M. Borkowski, S. Schulte, and C. Hochreiner, "Predicting cloud resource utilization," in *Proc. 9th Int. Conf. Utility Cloud Comput.*, Dec. 2016, pp. 37–42.
[10] G. Li, J. Song, J. Wu, and J. Wang, "Method of resource estimation based on QoS in edge computing," *Wireless Commun. Mobile Comput.*, vol. 2018, pp. 1–9, Jan. 2018.
[11] P. Vignesh and S. A. M. Musthafa, "Enhanced efficient load prediction algorithm in cloud computing," *Int. J. Grid Distrib. Comput.*, vol. 8, no. 5, pp. 9–14, 2015.
[12] J. Sun and Y. Zhuang, "The cloud computing load forecasting algorithm based on Kalman filter and ANFIS," in *Proc. 4th Int. Conf. Machinery, Mater. Comput. Technol.*, 2016, pp. 565–569.
[13] T. Nguyen, N. Tran, B. M. Nguyen, and G. Nguyen, "A resource usage prediction system using functional-link and genetic algorithm neural network for multivariate cloud metrics," in *Proc. IEEE 11th Conf. Service-Oriented Comput. Appl. (SOCA)*, Nov. 2018, pp. 49–56.
[14] A. A. Shahin, "Automatic cloud resource scaling algorithm based on long short-term memory recurrent neural network," *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, no. 12, pp. 1–7, 2016.
[15] H. Truta, J. L. Vivas, A. Brito, and T. Nobrega, "A predictive approach for enhancing resource utilization in PaaS clouds," in *Proc. Symp. Appl. Comput.*, Apr. 2017, pp. 384–391.
[16] N. Mostafa, I. A. Ridhawi, and M. Aloqaily, "Fog resource selection using historical executions," in *Proc. 3rd Int. Conf. Fog Mobile Edge Comput. (FMEC)*, Apr. 2018, pp. 272–276.
[17] C. Nguyen, C. Klein, and E. Elmroth, "Multivariate LSTM-based location-aware workload prediction for edge data centers," in *Proc. 19th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2019, pp. 341–350.
[18] W. Wei, H. Gu, W. Lu, T. Zhou, and X. Liu, "Energy efficient virtual machine placement with an improved ant colony optimization over data center networks," *IEEE Access*, vol. 7, pp. 60617–60625, 2019.
[19] J. Bi, S. Li, H. Yuan, Z. Zhao, and H. Liu, "Deep neural networks for predicting task time series in cloud computing systems," in *Proc. IEEE 16th Int. Conf. Netw., Sens. Control (ICNSC)*, May 2019, pp. 86–91.
[20] I. B. M. Redbooks, *A Practical Guide to IBM Autonomic Computing Toolkit*. New York, NY, USA: IBM, 2004.
[21] E. Zavala, "Towards adaptive monitoring services for self-adaptive software systems," in *Proc. Int. Conf. Service-Oriented Comput.* in Lecture Notes in Computer Science, L. Braubach, J. M. Murillo, N. Kaviani, M. Lama, L. Burgueño, N. Moha, and M. Oriol, Eds. New York, NY, USA: Springer, 2018, pp. 357–362.
[22] G. J. Guerrero, "Proteo a self-adaptive software architecture to support quality attributes in ubiquitous systems," Ph.D. dissertation, Dept. Softw. Eng., Univ. Granada, Granada, Spain, 2018. Accessed: Jun. 21, 2021. [Online]. Available: https://dialnet.unirioja.es/servlet/tesis?codigo=150988
[23] K. Fatema, V. C. Emeakaroha, P. D. Healy, J. P. Morrison, and T. Lynn, "A survey of cloud monitoring tools: Taxonomy, capabilities and objectives," *J. Parallel Distrib. Comput.*, vol. 74, no. 10, pp. 2918–2933, Oct. 2014.
[24] T. L. Duc, R. G. Leiva, P. Casari, and P.-O. Östberg, "Machine learning methods for reliable resource provisioning in edge-cloud computing: A survey," *ACM Comput. Surv.*, vol. 52, no. 5, pp. 1–39, Oct. 2019.
[25] M. C. Calzarossa, L. Massari, and D. Tessera, "Workload characterization: A survey revisited," *ACM Comput. Surv.*, vol. 48, no. 3, pp. 1–43, Feb. 2016.
[26] M. Ghobaei-Arani, R. Khorsand, and M. Ramezanpour, "An autonomous resource provisioning framework for massively multiplayer online games in cloud environment," *J. Netw. Comput. Appl.*, vol. 142, pp. 76–97, Sep. 2019.
[27] J. Carvalho, D. Vieira, and F. Trinta, "Greedy multi-cloud selection approach to deploy an application based on microservices," in *Proc. 27th Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process. (PDP)*, Feb. 2019, pp. 93–100.
[28] P. Di Francesco, P. Lago, and I. Malavolta, "Architecting with microservices: A systematic mapping study," *J. Syst. Softw.*, vol. 150, pp. 77–97, Apr. 2019.
[29] J. Rufino, M. Alam, J. Ferreira, A. Rehman, and K. F. Tsang, "Orchestration of containerized microservices for IIoT using docker," in *Proc. IEEE Int. Conf. Ind. Technol. (ICIT)*, Mar. 2017, pp. 1532–1536.
[30] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 3rd ed. Hoboken, NJ, USA: Wiley, 2020.
[31] A. Oztekin, Z. J. Kong, and O. Uysal, "UseLearn: A novel checklist and usability evaluation method for eLearning systems by criticality metric analysis," *Int. J. Ind. Ergonom.*, vol. 40, no. 4, pp. 455–469, Jul. 2010.
[32] A. Oztekin, D. Delen, A. Turkyilmaz, and S. Zaim, "A machine learning-based usability evaluation method for eLearning systems," *Decis. Support Syst.*, vol. 56, pp. 63–73, Dec. 2013.
[33] N. Ganesan, "Migration of an e-learning model to the cloud," *J. Int. Technol. Inf. Manage.*, vol. 22, no. 3, pp. 19–36, 2013.
[34] *Blackboard CourseSites*. [Online]. Available: https://www.coursesites.com/
[35] *Canvas Overview*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.instructure.com/canvas
[36] *Chamilo.org Asociación Chamilo*. Accessed: Mar. 3, 2021. [Online]. Available: https://chamilo.org/en/
[37] *Coursera | Build Skills With Online Courses From Top Institutions*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.coursera.org/
[38] *Duolingo—Learn a Language for Free*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.duolingo.com/
[39] *Edmodo*. Accessed: Mar. 3, 2021. [Online]. Available: https://new.edmodo.com/
[40] *edX | Free Online Courses by Harvard, MIT, & More*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.edx.org/
[41] EvolMind. *EvolMind—The LMS*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.evolmind.com/en
[42] *GeoGebra | Free Math Apps—Used by Over 100 Million Students & Teachers Worlwide*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.geogebra.org/
[43] *Google Classroom*. Accessed: Mar. 3, 2021. [Online]. Available: https://classroom.google.com/u/0/h
[44] *IXL | Maths and English Practice*. Accessed: Mar. 3, 2021. [Online]. Available: https://uk.ixl.com/
[45] *Khan Academy | Free Online Courses, Lessons & Practice*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.khanacademy.org/

[46] *Knewton—Achievement Within Reach*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.knewton.com/
[47] *Kornukopia—Cloud Based Learning Management System*. Accessed: Mar. 3, 2021. [Online]. Available: https://kornukopia.com/.
[48] *Moodle—Open-Source Learning Platform | Moodle.org*. Accessed: Mar. 3, 2021. [Online]. Available: https://moodle.org/
[49] *NEO LMS—The World's Best LMS for Schools and Universities*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.neolms.com/
[50] *Quizlet*. Accessed: Mar. 3, 2021. [Online]. Available: https://quizlet.com/
[51] *Schoology—Learning Management System*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.schoology.com/homepage
[52] *Scratch—Imagine, Program, Share*. Accessed: Mar. 3, 2021. [Online]. Available: https://scratch.mit.edu/
[53] *Studyx—Global Platform for Education Exchange*. Accessed: Mar. 3, 2021. [Online]. Available: https://studyx.co/en/
[54] *Udemy—Online Courses*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.udemy.com/
[55] *Udacity*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.udacity.com
[56] *WebAssign*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.webassign.net/
[57] *Wolframalpha: Making the World's Knowledge Computable*. Accessed: Mar. 3, 2021. [Online]. Available: https://www.wolframalpha.com
[58] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. [Online]. Available: http://www.deeplearningbook.org

JORGE ARIZA received the bachelor's degree in computer science and engineering from Universidad Simón Bolívar, in 2007. He is currently pursuing the master's degree in computer science and engineering with the Universidad del Norte. He has been working as a Software Engineer and a DevOps Engineer with a public high school in Colombia, where he also teaches programming courses. His e-mail address is marizaj@uninorte.edu.co.

MIGUEL JIMENO (Member, IEEE) was born in Barranquilla, Colombia, in 1979. He received the degree in computer science and engineering from the Universidad del Norte, Colombia, in 2002, and the master's degree in computer science and engineering and the Ph.D. degree in computer science from the University of South Florida, in 2007 and 2010, respectively. His research interests include cloud computing, deep learning, and cloud deployments of e-learning systems and healthcare information systems, from which he has published multiple articles. His experience started as a Software Engineer, then a Research Assistant as a Ph.D. Student and currently as an Associate Professor at the Universidad del Norte. He was awarded a grant from Fulbright to execute an applied research project in the area of healthcare in rural settings, in 2011.

RICARDO VILLANUEVA-POLANCO received the bachelor's degree from the Universidad del Norte, in 2008, the master's degree in computer science and engineering from the Universidad de los Andes, in 2010, and the Ph.D. degree in information security from Royal Holloway, University of London, in 2018. He is currently an Assistant Professor at the Department of Computer Science and Engineering, Universidad del Norte, Barranquilla, Colombia. His research interest includes cyber-security, in particular reliability of distributed systems (e.g., peer-to-peer systems and cloud systems) and applied cryptography (e.g., post-quantum cryptography, cryptographic protocols, and side-channel attacks on cryptographic implementations).

JOSE CAPACHO (Member, IEEE) received the bachelor's degree in systems engineering from the Universidad Industrial de Santander (UIS), Colombia, in 1982, the master's degree in education from Pontificia Universidad Javeriana, Colombia, in 1996, and the Ph.D. degree in learning processes in virtual spaces from the University of Salamanca, Spain, in 2008. He is currently an Assistant Professor at the Department of Computer Science and Engineering, Universidad del Norte, Barranquilla, Colombia. He has participated in the renewal accreditation process of the Systems Engineering Program with Accreditation Board for Engineering and Technology, Inc., from 2013 to 2015 and in 2021. His current research interests include e-learning and virtual education.