IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received February 28, 2021, accepted March 8, 2021, date of publication March 10, 2021, date of current version March 22, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3065597

Hybrid Auto-Scaled Service-Cloud-Based Predictive Workload Modeling and Analysis for Smart Campus System

MIRZA ABDUR RAZZAQ[1], JAVED AHMED MAHAR[1], MUNEER AHMAD[2], (Member, IEEE), NAJIA SAHER[3], ARIF MEHMOOD[3], AND GYU SANG CHOI[4], (Member, IEEE)

[1]Department of Computer Science, Shah Abdul Latif University, Khairpur 66020, Pakistan
[2]Department of Information Systems, Faculty of Computer Science and Information Technology, Universiti Malaya, Kuala Lumpur 50603, Malaysia
[3]Department of Computer Science and IT, The Islamia University of Bahawalpur, Bahawalpur 63100, Pakistan
[4]Department of Information and Communication Engineering, Yeungnam University, Gyeongsan 38541, South Korea

Corresponding author: Gyu Sang Choi (castchoi@ynu.ac.kr)

This work was supported in part by the Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education under Grant NRF-2019R1A2C1006159, and in part by the Ministry of Science and ICT (MSIT), Korea, under the Information Technology Research Center (ITRC) Support Program supervised by the Institute for Information and communications Technology Promotion (IITP) under Grant IITP-2020-2016-0-00313.

ABSTRACT The internet of things is an emerging technology used in cloud computing and provides many services of the cloud. The cloud services users mostly suffer from service delays and disruptions due to service cloud resource management based on vertical and horizontal scalable systems. Adding more resources to a single cloud server is called vertical scaling, and an increasing number of servers is known as horizontal scaling. The service-bursts significantly impact the vertical scaled environment where the scale-up degrades the service quality and users' trust after reaching the server's maximum capacity. Besides, the horizontally scaled environment, though being resilient, is cost-inefficient. It is also hard to detect and manage bursts online to sustain application efficiency for complex workloads. Burst detection in real-time workloads is a complicated issue because even in the presence of auto-scaling methods, it can dramatically degrade the application's efficiency. This research study presents a new bursts-aware auto-scaling approach that detects bursts in dynamic workloads using resource estimation, decision-making scaling, and workload forecasting while reducing response time. This study proposes a hybrid auto-scaled service cloud model that ensures the best approximation of vertical and horizontal scalable systems to ensure Quality of Service (QoS) for smart campus-based applications. This study carries out the workload prediction and auto-scaling employing an ensemble algorithm. The model pre-scales the scalable vertical system by leveraging the service-load predictive modeling using an ensemble classification of defined workload estimation. The prediction of the upcoming workload helped scale-up the system, and auto-scaling dynamically scaled the assigned resources to many users' service requests. The proposed model efficiently managed service-bursts by addressing load balancing challenges through horizontal auto-scaling to ensure application consistency and service availability. The study simulated the smart campus environment model to monitor the time-stamped diverse service-requests appearing with different workloads.

INDEX TERMS Auto-scaling, cloud computing, horizontal scalability, the Internet of Things, predictive modeling, quality of service (QoS), smart campus, vertical scalability, workloads.

I. INTRODUCTION

The Internet of Things (IoT) has gained tremendous attention in the last few years. The definition of the IoT was proposed in 1999 by Kevin Ashton. The IoT is seen as the Internet's future. IoT will play a vibrant contribution in the future and improve our living habits, standards, and business patterns. In the next coming years, the use of IoT in various apps is expected to increase rapidly [1]. In every area of life, the IoT is applied to render its vision anywhere for anyone at any

The associate editor coordinating the review of this manuscript and approving it for publication was Giovanni Merlino.

time. Smart devices are utilized to sense in IoT applications to collect data and transmit to servers on the cloud via a communication medium such as Wi-Fi, and information is communicated to smart actuator devices after processing for decision making [2].

The educational campuses are smart using IoT to collect real-time data for decision making to accommodate increasing needs [3].The smart campus results from the rapid growth of computer networks, IoT, and servers' advancement. It has three main features: customized services, information services, and platforms for the workplace. The concept for developing a smart campus is based on IoT to use advancements in information technology to incorporate the digital environment-based application framework to deliver services with better quality [4] It supports the convergence and combination of applications' benefits for the collection of data from campus and exchange to facilitate the teaching process, services, and research facilities [5].

Creating a smart campus is gaining popularity at universities worldwide due to the consequences of diverse use of the campus and load on resources like power, human and financial resources. The IoT will help to provide big data on patterns of use to guide users to take better decisions [3]. The smart campus is an evolving notion that enables educational institutes to upgrade their physical architecture with intelligent technology for better facilities, campus durability, and decision-making. Different technologies have been introduced on campus under the smart campuses' umbrella, such as smart classrooms, intelligent microgrids, resource and employee management, intelligent utility, etc. [6]. The use of IoT technology plays a vital role in advancing the educational environment in universities and colleges to improve teaching staff and education elasticity.

The superior qualities of a smart campus suggest an architecture model in the views of business, technology, and structure. The teaching framework in the smart campus consists of data capture and storage about the research content assessment to measure teaching performance in universities and colleges. The smart campus architecture model consists of four layers, including the intelligent sensing layer, data communication layer, intelligent processing layer, and smart recommendation layer [7]. The first smart sensing layer uses Zigbee protocol, IP CAM, and other sensors to sense the environment [8] for user data collection from the smart campus via the network. The second layer provides efficient data connection services by using different communication networks such as 4G. The third layer builds up comprehensive data support for the smart campus environment. It also integrates intelligent algorithms, raw data management, and other data mining engines to attain effective cloud services and computing and storage support for the smart campus. The fourth layer works with big data that is further combined with smart devices through networks. It provides users with personalized teaching methods, services, and research [7].

Smart campus is connected with IoT servers, users, peripherals, and infrastructure [9]. Classroom teaching is based on an IoT model with cloud integration for the educational system in smart campus. In education, IoT offers students access to learning novel technology that helps students build up innovative ideas and make social issues logical. Cloud computing based on IoT technology provides an intelligent framework, integrated portal, services, maintenance, and security systems. Digitally converged campuses strengthen environmental sustainability and learning for students. IoT objects monitor students who are bunking their classes, send notifications to assist students in focusing on academics, and locate missing personal belongings. Payments can be made simply at cafeteria and admin office activities via IoT devices. IoT infrastructure consists of sensors, microcontroller, wired or wireless channel [10]. The information is processed using the application module and communicated to cloud storage collected from the sensor modules. IoT and cloud computing effectively restructure conventional methods of learning and education. The IoT-based classrooms in smart campuses would advance the education system resulting in high productivity and quality of the teaching approach for classrooms [11].

System scalability means a system's capacity to withstand an upsurge in data processing demands. There are two ways to categorize the big data processing platforms. One is Vertical Scalability that includes upgrading one server with extra processors, faster memory, and hardware. The scaling up is the concept referred to this way. Another is Horizontal Scalability that includes the spread of workloads over many servers (these could also be commodity machines). The scaling out is the concept defined in this manner. To maximize data processing efficiency, it requires the stacking of several independent devices. Usually, separate computers run different operating systems [12].

Vertical and horizontal scalabilities are promising strategies to distribute resources efficiently at runtime and in an adaptive means; the earlier diverges the resource shares on each machine individually, while the last deals with selecting the number of virtual machines working. Unfortunately, in an isolated manner, up-to-date techniques primarily implement vertical and horizontal scalability [13]. According to the infrastructure point of view, a new scalability taxonomy is proposed and described in this article and classified into four groups, i.e., server, communication, gateway, and device-level scaling. Vertical and horizontal scaling is a classification according to future flexibility [14]. In this paper, the server level access the vertical scalability. A smart campus transport system is considered a case study—a custom-built simulator and virtual server test the scalability.

In automatic scaling, the server cluster can dynamically add or remove virtual resources according to platform operating states to attain scalability. Adding virtual resources when the system senses that the system's usage exceeds the threshold is the scalability mechanism's general approach. We can then perform the horizontal or vertical scaling. Horizontal scalability (Virtual Machine level scalability) is not concerned with the resource's limitations, but it accomplishes in

a while. It can contribute to violations of user SLA, Service Level Agreement. It is possible to disregard the scalability time taken by the vertical scalability (resource-level scalability, self-healing scalability), but resources bound the vertical scalability. Consequently, it is required if the platform can be pre-scaled earlier based on predictions, recommending a workload prediction approach [15]. The authors in [16] proposed an advanced nova-scheduler that improved CPU utilization by lessoning execution time of applications to 50% resulting in multiple virtual machines running on a cloud server avoiding user SLA.

Autoscaling approach experience multiple burst-workloads. It containerizes microservices through a trace-driven simulation. It enhances efficiency compared with current state-of-the-art autoscaling techniques. Experimental assessment defines specific testbed facilities, and the benchmark microservices check the trace-driven simulation findings. The proposed autoscaling system can manage bursting workloads with Service Level Objectives (SLO) breaches of minimum response time [17].

With IoT based frameworks, predictive analytics dramatically advance smart healthcare systems in predicting death tendencies due to in-time involvement. The model uses an automated technique of gaining knowledge to rate the Information-rich mortality for precise predictions. The classifiers evaluated the real-world datasets, such as the induction of rules, Naive Bayes, Random Forest, decision tree, and perceptron multi-layer, for prediction. The experiment has shown that the Naive Bayes classifier predicts the rate of infant mortality with the maximum average correctness of 96.4 % [18].

The contribution of this paper is as follows

- The model automatically detects service-bursts and handles a large number of requests for services from users.
- The approach builds up a time-stamped server queue that lists incoming users' requests according to service load.
- The load will be shifted to horizontally scaled server in case of any burst otherwise it will be accommodated by vertically scaled server.

The rest of the paper is as follows. Section II discusses the literature review, section III proposed methodology, section IV presents results and discussion, and finally, the article is concluded in section V.

II. LITERATURE REVIEW

In [19], the authors presented proactive and reactive auto-scaling concepts to handle cloud resource allocation/deallocation elasticity. The application serving on the cloud is facing a problem due to workload spikes, thus responding slow or denying requests. In proactive autoscaling, the workload is predicted before providing resources to handle the surge in claims. Whereas in the reactive autoscaling approach, resources are allocated/deallocated on surging workload. Scaling Machine Learning is not just about scaling the datasets to larger ones. Imagine we create an email service, and there are millions of users all of a sudden. We can develop an excellent model for spam detection, and it scales to massive datasets evenly, but it is needed to make hundreds of millions of forecasts a day now. That amounts to more than ten thousand per second. It is required to explore ways to scale the prediction volume and scale the pace when predictions need to be used in real-time. Small computing resources (fog nodes) are positioned next to the end devices through fog computing, and these nodes process data in real-time and control local devices. The smart campus becomes distributed and scalable in this way. Recently, the IoT incorporates cloud computing. More end devices are deployed and linked to the cloud, including IoT devices, power equipment, and smartphone users. Fog-computing processes user data in real-time with distributed resources and uses local contextual knowledge [20].

In virtual platforms to achieve the performance goals of distributed co-existing applications is a challenging task. In this situation, capable approaches include vertical and horizontal scalability. The first adds or removes the resources like processing power, memory, or storage on a specific machine, whereas the other allocates or deallocates the virtual machines. Advanced approaches primarily apply isolated vertical and horizontal scalability, particularly assuming a symmetrical and fixed load balancing system through replicas. Unfortunately, when replicas are executed in environments with various computing resources, this can be unsatisfactory. A new combined runtime strategy is suggested to assess the resource sharing quota and the horizontal load balancing policy to address this constraint to achieve performance goals such as response time and throughput of co-existing applications. Authors further formulate a model predictive control problem starting from a performance model as a multi-class queueing network, which can be effectively resolved by linear programming. A validation conducted on a shared virtualized system hosting two real applications demonstrates that only a combined vertical and horizontal load balancing adaptation can effectively achieve desired performance targets in the presence of heterogeneous computing tools. The main developments were the simultaneous use of autoscaling methods for vertical and horizontal load balancing [13].

A lightweight approach (vertical scaling) is proposed to allow cloud applications elasticity cost-effectively. In addition to VM-level scaling (horizontal scaling), light scaling executes a fine-grained method at the resource level CPUs, I/O, memory, etc. A design is also presented and implemented for an intelligent framework to manage cloud applications with lightweight resources. Algorithms for lightweight scalability and VM-level scalability are discussed to show their communication. Light scaling can be dependent upon two methods. The first is Self-healing scaling, which is applied when two application servers are installed on a single physical machine. Free resources of one Virtual Machine can be used to free the overcrowded resources of others. It can remove one or more CPUs assigned to a VM with less CPU usage and allocate to others. The second method is Resource level

scaling depends upon the use of free resources accessible by a specific Physical Machine up-scale a Virtual Machine running on it. A Physical Machine with less CPU and memory consumption can assign these kinds of resources to one of the Virtual Machines operating on it and scales up. The strategy can scale up and down the cloud application's help effectively to meet the given quality requirements while reducing cloud vendors' expenses. An intelligent framework based on lightweight scaling has been developed [21].

A model-based technique is presented to guarantee the application's performance achieves the user-defined SLO capability during runtime vertical scalability, i.e., allocating or deallocating resources of distinct Virtual Machines VMs running the program. Using resource request prediction methods, a layered performance framework explains the relationship between the allocation of resources and application observation performance is automatically calculated and modified online. This model adapts the number of virtual CPUs of every VM dynamically [22].

The study [23] proposes an auto-scaling method based on the prediction of workload. It reduces the cost of scalability and improves resource usage without facing any difficulty by the users. The framework is distributed into two fragments, i.e., autoscaling and load prediction. In the auto-scaling environment, the systems add and remove the required resources with minimal vertical and horizontal scalability costs using the integer programming approach. The study presents the auto-scaling approach to decrease the cost by 14% and 11% compared to horizontal and vertical scaling. The authors in [24] discussed Google's autoscaling approach, Autopilot, to organize resources with horizontal scaling in a job of simultaneous threads and with vertical scaling for distinct tasks to control limits of CPU/memory automatically. Autopilot's goal is to minimize slack primarily to limit the use of resources by reducing out-of-memory risk without degradation of CPU performance.

The authors in [15] suggested a linear regression model to forecast the workload of service cloud platforms, and a novel service cloud architecture is presented that provides various services and an auto-scaling approach in service clouds. It incorporates the pre-scaling and real-time scaling method. Three scaling ways are taken into account at different resource levels self-healing scalability, resource-level scalability, and VM-level scalability. The auto-scaling process dynamically adds or releases virtual resources according to the running states of the platform. The scaling method's ultimate strategy is to add virtual resources when it is observed by the system that the system utilization exceeds the threshold. Horizontal and Vertical scaling can be performed. Horizontal scaling (VM-level scaling) is not subject to the resource's limitations, but it takes some time to complete. It can lead to users being violations of the SLA.

Service platforms have drawbacks, such as long construction times, low utilization of resources, and isolated structures. These issues can be addressed by migrating service systems into clouds. Reference [25] proposed an auto-scaling mechanism based on expected workload to scale virtual resources into service clouds at different resource levels. Real-time scaling and pre-scaling are combined with the automated scaling mechanism. A cloud scaling architecture is presented to incorporate these mechanisms. According to the test findings, the solution forecasts reliable and cost-effective ways to lower SLA breaches by the customer. The answer is generic, and in most service cloud situations, it can be used well [25].

In the study [26], the authors introduced a new software-defined networking (SDN) architecture to address the QoS requirements of different IoT services and handle traffic among IoT servers concurrently. The problem is defined as an integer linear programming (ILP) model, which is NP-hard. A proactive and predictive heuristic method is then suggested based on fuzzy logic and analysis on time series. Later, the proposed system is then installed in an actual testbed consisting of the Kaa servers, Open vSwitch, and Floodlight controller. In paper [27], the authors focused on the early spotting variation in the deviation popularity of videos, which is discussed as a statistically Change Point recognition problem. In the parametric frameworks, it is linear in the shape of ARMA, Auto Regressive Moving Average. Without an appropriate procedure to predict, the use of cloud resources may face a scalability problem [28]. To handle the scaling situation, Neural Network's approach based on the programming of cartesian genetic to estimate resources scalability based on the rule for a cloud server. The proposed technique consists of monitoring and estimating resources for the scaling approach. The resource monitor measures the utilization of resources efficiently. The scalability mechanism scales the resources after the estimation of resources.

The authors in [29] used the Autoregressive Integrated Moving Average (ARIMA) model to predict the estimated number of cases of COVID-19 daily in the next four weeks in Saudi Arabia. To find the best fit model, the authors tested four different prediction models: Moving Average, Autoregressive Model, a mixture of ARMA and Integrated ARMA (ARIMA). Finally, the ARIMA model outperformed the other models.

Auto-scaling techniques would pay attention to use resources suitably along with service quality. A procedure to forecast the workload is essential for servers to accommodate users' requests [30]. An auto-scaling arrangement is needed to manage load and balance the service to provide resources with dynamism without disruption. The researchers in [31] proposed a reinforcement learning RL autoscaling approach for servers in the cloud, which is dynamic, transparent, and provides resource management for applications. It used distinctive features in an ad-hoc manner and heuristics based on statistics. Autoscaling optimizes the application execution elastically accordingly given optimal criteria, which decides how and when to scale-up or scale-down resources and how to allocate the resources to the next coming workload.

Different techniques like linear regression, moving average, ARMA, ARIMA, etc. are used to enhance system's

throughput by scaling up or scaling out. The existing research adopted either horizontal or vertical scaling. We have considered auto-scaling having hybrid characteristics empowered with burst identification using an ensemble classification.

III. METHODOLOGY

This study proposes a hybrid auto-scaled service cloud predictive model that leverages the vertical and horizontal scalable service-models. The model auto-detects the service-bursts and manages a large number of users' services-requests. The model builds a server queue that enlists the incoming users' requests. These service-requests are time-stamped queued with their respective service-load. At the first instance, the model predicts the queued service-requests for any possibility of service-bursts. If the system identifies the service-requests as service-bursts, the system balances the workload by shifting the service-load to horizontally scaled servers. In such a case, the system can dynamically perform the horizontal scaling to distribute the workload among horizontally scaled systems further. On the other hand, if the system doesn't flag the incoming service-requests to be service-bursts, then the vertically scaled server can efficiently entertain the users' requests.

Figure 1 presents the proposed hybrid auto-scaled cloud service model. This model consists of three main components. The first component is the input-channel that enlists the incoming users' requests for different cloud services. The input channel queued the incoming requests with time-stamp and calculated the workloads. The model contains an input queue that supports 750 service requests to simulate the proposed system in a smart campus scenario. We assume that each service request consumes 0.1% of system resources. This way, the 750 service requests can finish, on average, 75% of resources. If the incoming service requests are within this threshold, the vertically scaled system can monitor and assign the system resources. In case the next batch of service requests breach the defined threshold, the system predicts the incoming requests as burst and tries to balance the workload by scaling the service requests using horizontal nodes. The system can perform the auto-scaling to uniformly distribute the workload among the different nodes at the horizontal scaled level.

Figure 2 describes the predictive model of the system for automatic service-burst identification. We have employed an ensemble classification to classify the incoming service-requests as bursts or non-bursts. This predictive model is trained on the historical users' services-request data developed due to 750 initial users' requests. Since we are adopting the smart campus model, we assume that different users request different service requests. Let's take that the server can provide ten different kinds of resources/services. For each resource/service type, the time stamp and workload parameters assign a certain weight to the individual request. The predictive model estimates the bunch of service-requests as bursts based on the availability and assignment of resource/service to each request. Here,

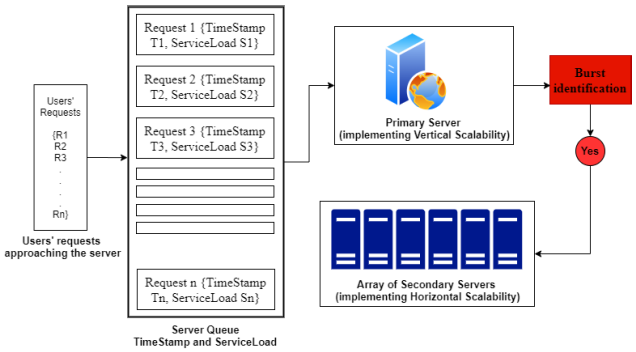

FIGURE 1. The proposed hybrid auto-scaled cloud service model.

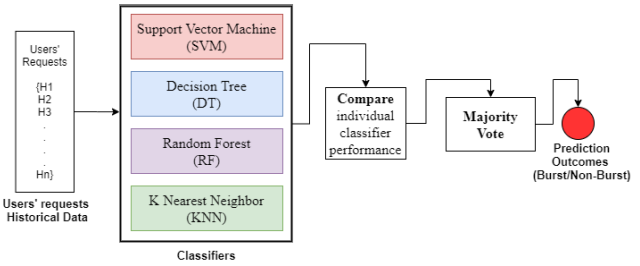

FIGURE 2. Predictive modeling for service-bursts classification.

we adopt another assumption that if the server can assign 75% of its resources/services to individual requests, the individual requests are marked with a probability of resulting in bursts. This way, a historical transactional database is formed that enlists each request as an instance with ten independent features (representing the assignment or otherwise of a system resource to a user request) and once dependent feature (class instance). The class instance carries the flag either set to zero or one, based on the 75% (or more) assignment of system resources to users-requests.

To perform the predictive modeling for auto-bursts identification, we have employed ensemble classification. The research proves that ensemble classifiers outperform the prediction accuracy of individual classifiers. We have taken the popular classifiers, namely, Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and K-nearest neighbor (KNN). The majority voting policy inspires the ensemble classification, and the best prediction outcomes are adopted to enhance the prediction accuracy of individual classification methods. Using different evaluation measures, the study ensures that the ensemble method outperforms the listed individual classifiers. The improved classification outcomes significantly helped estimate the workload of incoming users-requests and further distributed them across vertical and horizontal service managers.

Figure 3 presents the schematic implementation of the proposed model. Let's assume a user request queue R contains N users' requests denoted by $R = \{R_1, R_2, R_3, \ldots R_i\}$ for $i <= N$. The system enqueues the users' requests based on their arrival sequence concerning time and workload. Initially, we set the flag variable to zero. This flag variable

TABLE 1. Service requests with resultant burst.

Request Number	R_1	R_2	R_3	R_4	R_5	R_6	R_7	R_8	R_9	R_{10}	Burst
1	0	0	1	0	1	1	0	1	0	1	0
2	0	0	0	0	0	1	0	1	1	0	0
3	1	1	0	0	0	0	1	1	1	0	1
4	1	1	1	1	1	1	1	1	0	0	0
5	0	0	0	1	1	0	0	0	0	0	0
6	0	1	1	1	1	1	1	1	1	0	0
7	1	1	1	1	1	1	1	0	0	0	1
8	0	0	1	1	1	1	1	0	0	0	0
9	0	1	1	0	1	0	1	1	1	0	0
10	1	1	1	0	0	1	1	0	0	0	0
11	0	0	0	1	1	0	0	0	0	0	0
12	0	0	0	1	1	0	0	1	1	1	1
13	0	0	0	1	1	1	1	1	0	0	0
14	0	0	0	1	1	0	0	1	0	0	1
15	0	0	0	0	1	1	1	1	0	0	1

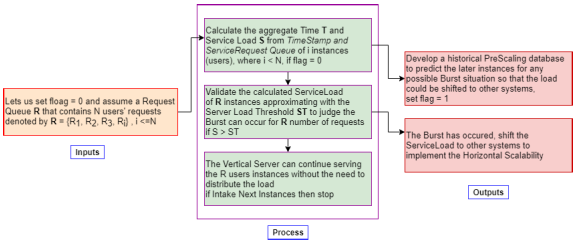

FIGURE 3. The schematic implementation of the proposed model.

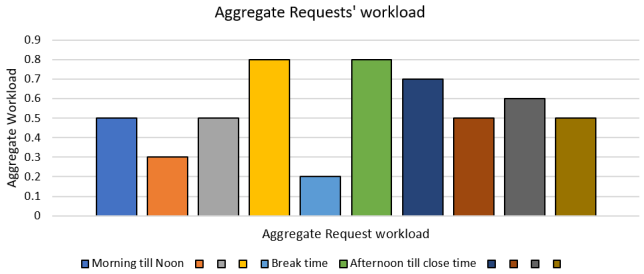

FIGURE 4. The aggregate workload of users requests for server resources.

helps us identify that many users' requests are incoming for the first time or some other time. We then calculate the aggregate time T and service load S from the time stamp and service-request queue of i instances (users), where $i < N$. The model ensures that the flag variable has the default zero value. If the flag meets the default condition, the system develops a historical Pre-Scaling database to predict the later instances for any possible burst situation so that the load could be shifted to other systems. The system then sets the flag variable to one so that the subsequent bunch of users' requests could be used for the allocation of resources/services. The new incoming requests are validated against R instances' service load approximating with the server load threshold ST to judge the burst that can occur for R number of requests. Here, we estimate $S > ST$. If this happens to true, the burst occurs, and the system shifts the service load to other systems to implement the horizontal scalability. On the other hand, if the system does not detect a burst, the Vertical Server can continue serving the R users instances without distributing the load.

IV. RESULTS AND DISCUSSION

This study implements the hybrid auto-scaling model of users' service requests in a smart campus scenario. In the intelligent campus concept, various users are accessing web services in a variety of different ways. Although various web applications are entertaining users' demands in such an environment, we assumed an aggregate service workload from a large number of users' service requests.

The experimentation involves a dataset of 7500 instances with a prototype service model of 10 user requests ranging from R_1 to R_{10}. We treat each service request with a binary value "0" or "1" corresponding to a respective service request, shown in Table 1. The prototype model evaluates the number of requests and maps to the available virtual resources.

We implement it, considering that the server maintains 75% of its resources to meet users' demands. Each service request from a user is treated as consuming 0.1% of the vertically scalable server-resources. This way, a service queue of length 750 is maintained at the server. When the user requests' service load exceeds this threshold, the server auto-scales the incoming request to other nodes arranged in a horizontally scalable manner.

Figure 4 describes the aggregate workload of users' service requests. The first four bars present the workload of campus operating hours from morning until noon. The fifth bar shows a minimum workload due to break time. From bar six to bar ten, we can see an increase in the service requests, which gradually adopts a decreasing load trend towards campus activities' closing time.

Figure 5 shows the cutoff line (threshold) of approximately 75% is maintained to ensure that any extra or exceeding workload will be scaled to horizontal nodes. It ensures that the vertical server never remains scales-out while serving the users' requests. Any additional workload is implicated to the other systems. The red line represents the average

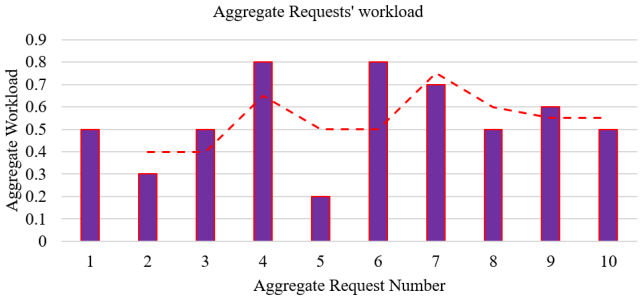

FIGURE 5. A cutoff line of the aggregate workload of users requests for server resources.

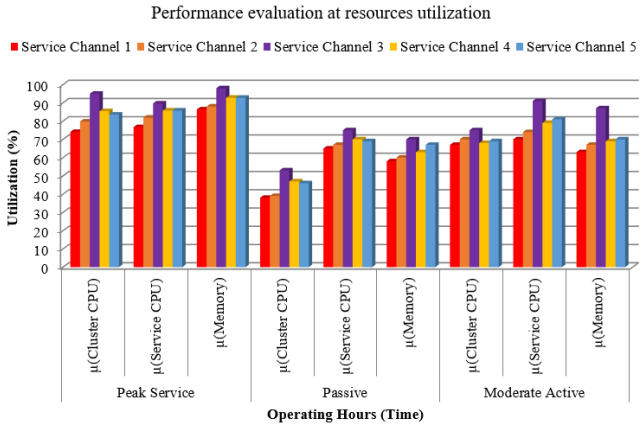

FIGURE 6. Utilization of system resources during operating hours.

service-burst that usually can occur at the peak load times during the campus operative hours.

Figure 6 depicts the computational cost in terms of system resources (CPU and memory utilization). The average cluster CPU utilization for reservation of user requests is the mean of all CPU utilization in such a resource allocation cluster. The average CPU utilization for processing requests is the mean utilization of CPU units during operative hours. We considered three services operating hours where the cost of resources varies according to services demand. The peak working hours demonstrate a relatively higher price, the break time shows a passive operating behavior, while moderate operations illustrate an average utilization of system resources. Aligned to the cutoff point, the system can balance the load by shifting the requests breaching the defined threshold.

This study develops a predictive model based on an ensemble classifier. We evaluated the performance of ensemble classification in comparison with the performance of individual classifiers, namely, Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and K-nearest neighbor (KNN). The majority voting policy inspires the ensemble classification, and the best prediction outcomes are adopted to enhance the prediction accuracy of individual classification methods. Using different evaluation measures, the study ensures that the ensemble method outperforms the listed individual classifiers. The improved classification

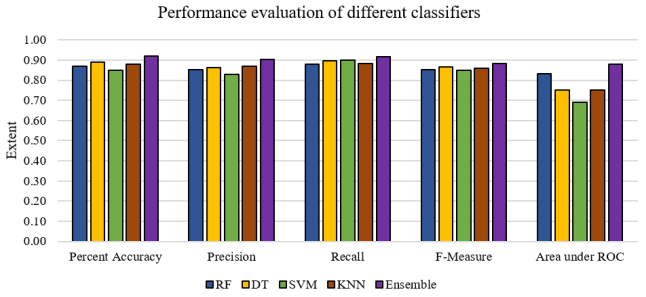

FIGURE 7. A comparative analysis of classifiers.

outcomes significantly helped estimate the workload of incoming users-requests and further distributed them across vertical and horizontal service managers.

The study conducted 10-fold cross-validations to measure the significance of different classification methods to identify the specific workload leading to burst. We have used the following metrics to measure the performance of different classifiers.

$$Accuracy = \frac{TP}{TP + FN} \times 100\%. \tag{1}$$

TP is the number of true positives and FN stands for false negative.

$$Precision = \frac{TP}{PP} \times 100\%. \tag{2}$$

PP is the number of predicted positives, and TP is the number of true positives.

$$Recall = \frac{TP}{AP} \times 100\%. \tag{3}$$

TP is the number of true positives, and AP is the number of actual positives.

$$F1 - Score = \frac{2TP}{2TP + FP + FN} \times 100\%. \tag{4}$$

FP is the number of false positives, FN is the number of false negatives, and TP is the number of true positives. The area under ROC as an evaluation measure is defined as a ratio of true positive rate to (1-false positive rate).

We evaluated the performance of different classifiers for the prediction of burst or non-burst situations. Here accuracy is the percentage of test instances correctly identified. Sensitivity and specificity relate to imbalance situations when class labels are highly biased. In our case, since the burst situation occurs rarely, the target variable's data is more inclined towards ''non-bursts'' scenarios. Further, we considered precision to find the exactness between our 10-fold cross-validations. Recall and F-Score are demonstrating completeness and harmonic mean of precision.

Figure 7 presents the comparative analysis of different classifiers in the classification of the service workload that appears as burst to auto-scale the servers accordingly. We can observe that the ensemble classifier outperforms the other classifiers. The lowest precision was recorded for the SVM classifier. The random forest and decision tree performed

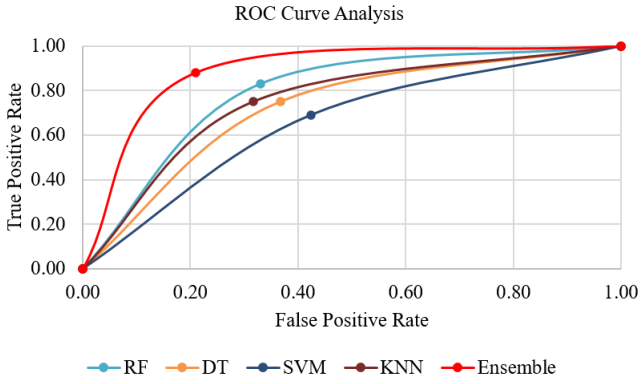

FIGURE 8. Comparative analysis of ROC curves of classifiers.

more likely in a similar fashion. The KNN performed better than SVM and decision trees. The average recall of each classifier is the aggregate of individual readings. The decision tree and SVM had the same performance, while the random forest and KNN turned out to be the same. Regarding AUC analysis, the SVM classifier remained under-performed while the decision tree's performance and KNN remained the same.

Figure 8 presents the comparative analysis of different classifiers in terms of ROC curve analysis. We can see that the ensemble classifier performs better than the individual classifiers. The curve is more leaned towards the top left corner that bestows it significantly different from other curves.

V. CONCLUSION

This study presented a hybrid and auto-scalable cloud service model inspired by integrating the features of vertical and horizontal scalable models. This study develops a predictive model based on an ensemble classifier. We evaluated the performance of ensemble classification in comparison with the performance of individual classifiers, namely, Support Vector Machine (SVM), Decision Tree (DT), Random Forest (RF), and K-nearest neighbor (KNN). The vertical service model develops a users' request service queue maintaining the time stamp and service load of queries. The model auto-scales the resources depending on the incoming requests from users. The model calculates the aggregate workload and predicts the scalability factor. A cutoff line (threshold) of approximately 75% is maintained to ensure that any extra or exceeding workload will be scaled to horizontal nodes. It ensures that the vertical server never remains scales-out while serving the users' requests. Any additional workload is implicated to the other systems. The input channel queued the incoming requests with time-stamped and calculated the workloads. The model contains an input queue that supports 750 service requests to simulate the smart campus scenario's proposed approach. We assume that each service request consumes 0.1% of system resources. This way, the 750 service requests can consume, on average, 75% of resources. If the incoming service requests are within this threshold, the vertically scaled system can monitor and assign the system resources. In case the next batch of service requests breach the defined threshold, the system predicts the incoming requests as burst and tries to balance the workload by scaling the service requests using horizontal nodes. The system can perform the auto-scaling to uniformly distribute the workload among the different nodes at the horizontal scaled level. Since we conducted a prototype study at the campus level, we did not consider the SLA aspect, and we hope to include this aspect as a future direction of work.

REFERENCES

[1] M. Abdur, S. Habib, M. Ali, and S. Ullah, "Security issues in the Internet of Things (IoT): A comprehensive study," *Int. J. Adv. Comput. Sci. Appl.*, vol. 8, no. 6, pp. 383–388, 2017.
[2] A. Razzaq, "Internet of Things (Iot) applications an overview," *Int. J. Comput. Sci. Emerg. Technol.*, vol. 1, no. 1, pp. 43–48, 2017.
[3] B. Valks, M. H. Arkesteijn, A. Koutamanis, and A. C. den Heijer, "Towards a smart campus: Supporting campus decisions with Internet of Things applications," *Building Res. Inf.*, vol. 49, no. 1, pp. 1–20, 2020, doi: 10.1080/09613218.2020.1784702.
[4] A. Alghamdi and S. Shetty, "Survey toward a smart campus using the Internet of Things," in *Proc. IEEE 4th Int. Conf. Future Internet Things Cloud (FiCloud)*, Aug. 2016, pp. 235–239, doi: 10.1109/FiCloud.2016.41.
[5] T. Bi, X. Yang, and M. Ren, "The design and implementation of smart campus system," *JCP*, vol. 12, no. 6, pp. 527–533, 2017.
[6] N. Min-Allah and S. Alrashed, "Smart campus—A sketch," *Sustain. Cities Soc.*, vol. 59, Aug. 2020, Art. no. 102231, doi: 10.1016/j.scs.2020.102231.
[7] X. Xu, Y. Wang, and S. Yu, "Teaching performance evaluation in smart campus," *IEEE Access*, vol. 6, pp. 77754–77766, 2018, doi: 10.1109/ACCESS.2018.2884022.
[8] M. Srivastava and R. Kumar, "Smart environmental monitoring based on IoT: Architecture, issues, and challenges," in *Advances in Computational Intelligence and Communication Technology*, vol. 1086. Singapore: Springer, 2021, pp. 349–358, doi: 10.1007/978-981-15-1275-9_28.
[9] V. Subbarao, K. Srinivas, and R. S. Pavithr, "A survey on Internet of Things based smart, digital green and intelligent campus," in *Proc. 4th Int. Conf. Internet Things: Smart Innov. Usages (IoT-SIU)*, Apr. 2019, pp. 1–6, doi: 10.1109/IoT-SIU.2019.8777476.
[10] A. Majeed and M. Ali, "How Internet-of-Things (IoT) making the university campuses smart? QA higher education (QAHE) perspective," in *Proc. IEEE 8th Annu. Comput. Commun. Workshop Conf. (CCWC)*, Jan. 2018, pp. 646–648, doi: 10.1109/CCWC.2018.8301774.
[11] R. Revathi, M. Suganya, and N. R. G. Merlin, "IoT based cloud integrated smart classroom for smart and a sustainable campus," *Procedia Comput. Sci.*, vol. 172, pp. 77–81, Jan. 2020.
[12] A. H. Ali and M. Z. Abdullah, "A survey on vertical and horizontal scaling platforms for big data analytics," *Int. J. Integr. Eng.*, vol. 11, no. 6, pp. 138–150, Sep. 2019.
[13] E. Incerto, M. Tribastone, and C. Trubiani, "Combined vertical and horizontal autoscaling through model predictive control," in *Proc. Eur. Conf. Parallel Process.*, Cham, Switzerland: Springer, 2018, pp. 147–159.
[14] M. A. Razzaq, J. A. Mahar, M. A. Qureshi, and Z. Abidin, "Smart campus system using Internet of Things: Simulation and assessment of vertical scalability," *Indian J. Sci. Technol.*, vol. 13, no. 28, pp. 2902–2910, 2020, doi: 10.17485/IJST/v13i28.1034.
[15] J. Yang, C. Liu, Y. Shang, Z. Mao, and B. Cheng, J. A. Chen. (2013). *A Cost-Aware Auto-Scaling Approach Based on Workload Predictions in Service Clouds*. [Online]. Available: https://infocom2013.ieee-infocom.org/images/stories/infocom/studentposter/1569715211.pdf
[16] M. Liaqat, A. Naveed, R. L. Ali, J. Shuja, and K.-M. Ko, "Characterizing dynamic load balancing in cloud environments using virtual machine deployment models," *IEEE Access*, vol. 7, pp. 145767–145776, 2019, doi: 10.1109/ACCESS.2019.2945499.
[17] M. Abdullah, W. Iqbal, J. L. Berral, J. Polo, and D. Carrera, "Burst-aware predictive autoscaling for containerized microservices," *IEEE Trans. Services Comput.*, early access, May 20, 2020, doi: 10.1109/TSC.2020.2995937.
[18] M. Islam, M. Usman, A. Mahmood, A. A. Abbasi, and O.-Y. Song, "Predictive analytics framework for accurate estimation of child mortality rates for Internet of Things enabled smart healthcare systems," *Int. J. Distrib. Sensor Netw.*, vol. 16, no. 5, May 2020, Art. no. 155014772092889, doi: 10.1177/1550147720928897.

[19] R. Y. Secaran and E. Sathiyamoorthy, "A survey on workload prediction models in cloud based on spot instances for proactive auto scaling strategy," *J. Crit. Rev.*, vol. 7, no. 4, pp. 791–795, 2020.
[20] U. Na and E.-K. Lee, "Fog BEMS: An agent-based hierarchical fog layer architecture for improving scalability in a building energy management system," *Sustainability*, vol. 12, no. 7, p. 2831, Apr. 2020, doi: 10.3390/su12072831.
[21] R. Han, L. Guo, M. M. Ghanem, and Y. Guo, "Lightweight resource scaling for cloud applications," in *Proc. 12th IEEE/ACM Int. Symp. Cluster, Cloud Grid Comput. (CCGRID)*, May 2012, pp. 644–651.
[22] S. Spinner, S. Kounev, X. Zhu, L. Lu, M. Uysal, A. Holler, and R. Griffith, "Runtime vertical scaling of virtualized applications via online model estimation," in *Proc. IEEE 8th Int. Conf. Self-Adaptive Self-Organizing Syst.*, Sep. 2014, pp. 157–166.
[23] C. Li, J. Tang, and Y. Luo, "Elastic edge cloud resource management based on horizontal and vertical scaling," *The J. Supercomputing*, vol. 76, pp. 1–26, 2020, doi: 10.1007/s11227-020-03192-3.
[24] K. Rzadca, P. Findeisen, J. Swiderski, P. Zych, P. Broniek, J. Kusmierek, P. Nowak, B. Strack, P. Witusowski, S. Hand, and J. Wilkes, "Autopilot: Workload autoscaling at Google," in *Proc. 15th Eur. Conf. Comput. Syst.*, Apr. 2020, pp. 1–16.
[25] J. Yang, C. Liu, Y. Shang, Z. Mao, and J. Chen, "Workload predicting-based automatic scaling in service clouds," in *Proc. IEEE 6th Int. Conf. Cloud Comput.*, Jun. 2013, pp. 810–815.
[26] A. Montazerolghaem and M. H. Yaghmaee, "Load-balanced and QoS-aware software-defined Internet of Things," *IEEE Internet Things J.*, vol. 7, no. 4, pp. 3323–3337, 2020, doi: 10.1109/JIOT.2020.2967081.
[27] S. Skaperas, L. Mamatas, and A. Chorti, "Real-time algorithms for the detection of changes in the variance of video content popularity," *IEEE Access*, vol. 8, pp. 30445–30457, 2020, doi: 10.1109/ACCESS.2020.2972640.
[28] Q. Z. Ullah, G. M. Khan, and S. Hassan, "Cloud infrastructure estimation and auto-scaling using recurrent Cartesian genetic programming-based ANN," *IEEE Access*, vol. 8, pp. 17965–17985, 2020, doi: 10.1109/ACCESS.2020.2966678.
[29] S. I. Alzahrani, I. A. Aljamaan, and E. A. Al-Fakih, "Forecasting the spread of the COVID-19 pandemic in Saudi Arabia using ARIMA prediction model under current public health interventions," *J. Infection Public Health*, vol. 13, no. 7, pp. 914–919, 2020.
[30] S. Bhagavathiperumal, "Auto scaling of cloud resources using time series and machine learning prediction," M.S. thesis, School Comput. Sci., Univ. Technol. Sydney, Ultimo, NSW, Australia, 2020.
[31] Y. Garí, D. A. Monge, E. Pacini, C. Mateos, and C. García Garino, "Reinforcement learning-based application autoscaling in the cloud: A survey," 2020, *arXiv:2001.09957*. [Online]. Available: http://arxiv.org/abs/2001.09957

MIRZA ABDUR RAZZAQ received the Master of Computer Science degree from The Islamia University of Bahawalpur, Bahawalpur, Pakistan, in 2002, and the M.S. degree in computer science from the Virtual University of Pakistan, Lahore, Pakistan, in 2015. He is currently pursuing the Ph.D. degree with Shah Abdul Latif University, Khairpur, Pakistan. He is serving as an Assistant Professor of computer science and the Head of the Department of Computer Science, with the Government Associate College, Sadiqabad, Pakistan. His research interests include the Internet of Things, Smart Campus, the IoT Security, and Watermarking.

JAVED AHMED MAHAR received the M.Sc. degree in computer science from Shah Abdul Latif University, Khairpur, Pakistan, in 1996, the M.S. degree in computer science from the Karachi Institute of Economics and Technology (PAF-KIET), Karachi, Pakistan, in 2007, and the Ph.D. degree from Hamdard University, Karachi, in 2012. He is currently a Professor with the Department of Computer Science, Shah Abdul Latif University, Khairpur. He has got more than 22 years of teaching experience. He has published several articles in national and international journals and peer-reviewed conference proceedings. His research interests include natural language processing, speech and image processing, big data analytics, software engineering, and programming paradigms.

MUNEER AHMAD (Member, IEEE) received the Ph.D. degree in computer science from Universiti Teknologi PETRONAS, Malaysia, in 2014. He has 18 years of teaching, research, and administrative experience internationally. He has authored numerous research articles in refereed research journals, international conferences and books. He successfully completed several funded research projects. His research interests include data science, big data analysis, machine learning, bioinformatics, and medical informatics.

NAJIA SAHER received the degree in software project management from the FAST-National University of Computer and Emerging Science, Pakistan, in 2012, and the Ph.D. degree in computer science from the School of Computing, Universiti Utara Malaysia. She is currently an Assistant Professor with the Department of Computer Science and IT, The Islamia University Bahawalpur, Pakistan. Her research interest includes human and social aspects of software engineering, empirical software engineering, software project management, agile software development, and software quality.

ARIF MEHMOOD received the Ph.D. degree from the Department of Information and Communication Engineering, Yeungnam University, South Korea, in November 2017. He is currently working as an Assistant Professor with the Department of Computer Science and IT, The Islamia University of Bahawalpur, Pakistan. His research interests include data mining, mainly working on AI and deep learning-based text mining, and data science management technologies.

GYU SANG CHOI (Member, IEEE) received the Ph.D. degree from the Department of Computer Science and Engineering, Pennsylvania State University, University Park, USA, in 2005. He was a Research Staff Member with the Samsung Advanced Institute of Technology (SAIT), Samsung Electronics, from 2006 to 2009. Since 2009, he has been a Faculty Member with the Department of Information and Communication, Yeungnam University, South Korea. His research interests include AI and data mining. He is a member of ACM.