Received 16 June 2023, accepted 28 July 2023, date of publication 1 August 2023, date of current version 8 August 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3300956

APPLIED RESEARCH

A Case Study of Transactional Workload Running in Virtual Machines: The Performance Evaluation of a Flight Seats Availability Service

CARLOS JUIZ[1], (Senior Member, IEEE), BARTOMEU CAPO[1,2], BELEN BERMEJO[1], (Member, IEEE), ALEJANDRO FERNÁNDEZ-MONTES[3], AND DAMIÁN FERNÁNDEZ-CERERO[3]

[1]Computer Science Department, University of the Balearic Islands, 07122 Palma, Spain
[2]Multinucleo Soluciones de Alto Rendimiento S.L., 07006 Palma, Spain
[3]Department of Computer Languages and Systems, University of Seville, 41012 Seville, Spain

Corresponding author: Carlos Juiz (cjuiz@uib.es)

This publication is part of the TED2021-132695B-I00 project, financed by MCIN/AEI/10.13039/501100011033 and by the European Union "NextGenerationEU"/PRTR.

ABSTRACT Much of the research has focused on performance evaluation and, particularly, the response time of clusters in cloud computing. However, one important topic has hardly been addressed: the impact of virtual machine consolidation on real business cases, on companies driven by requirements for high performance in transaction response time, specifically on intermediation trip companies. The ability to provide quality service, guaranteed within several milliseconds, is crucial to the business success of these cluster platforms. We present a case study for evaluating the performance of the seat availability service used by a flight carrier. The case study is the application of the performance evaluation methodology that ranges from monitoring to tuning options of a real-world service running on virtual machines, to understand capacity planning or possible substitution by other configurations of virtualization or containerization of the architecture of the cloud platform. This case study also proposes a workload characterisation using data clusters, allowing the architecture to be modeled as a simple network of multiclass queues of any virtual machine on the platform. Additionally, we estimate the new transaction response time by the possibility of either reducing or incrementing the number of virtual machines and their replacement by containers

INDEX TERMS Virtual machines, performance evaluation, monitoring, workload characterization, discrete-event simulation, overhead, containers.

I. INTRODUCTION

Cloud computing aims to shift the location of the computing infrastructure to the Internet to reduce the costs of management and maintenance of hardware and software resources [1]. Cloud service providers offer high performance, scalability, security, and availability [2]. However, performance issues lead to the question of how to guarantee that the system can offer the required quality of service. This article presents the performance evaluation case study for a cloud platform based on the transaction response time of a service executed in virtual machines (VM). Response time in this article is considered as the time for a transaction to be serviced (with a requirement of a few milliseconds), in other words, the sum of the delays, waiting and servicing times in the VM. Therefore, this research considers the performance journey, from the transaction arriving on the cloud platform until its service is finalised.

The real-world case study consists of a virtualization platform for calculating the availability of flight inventory in a flag carrier. Only the availability of the inventory is calculated, that is, the calculation of the available seats based

The associate editor coordinating the review of this manuscript and approving it for publication was Tomás F. Pena.

on different B2B (Business-to-Business) demanding systems, for example, Global Distribution Systems (GDSs) to the flight carrier. This is a crucial business activity for companies in the tourism industry, which offer airplane seats, both the companies that mediate in the business (buying and reselling seats for their tourism packages), as well as the airlines that directly market their services.

Thus, the case presented in this article can be directly applied to almost all commercial and public platforms dedicated to serving transactions as quickly as possible on VM platforms.

In summary, this article includes a case study of the performance evaluation of a virtual machine platform in the cloud that is used to execute a critical service for a flight carrier, the engine for determining seat availability. The flight carrier expects this service to be executed, at most, in a response time of a few milliseconds, to be able to attend to the transactions that arrive from the main stakeholders of the airline business, e.g., the GDSs, even the airline's e-commerce. Consequently, this article collects how the performance study problem has been faced to satisfy the following question, the IT supplier and SaaS provider that runs and monitors the service:

How much residual capacity is left in the current virtual machine configuration to maintain the quality of service?

This question brings us to respond to two previous issues; first, *what characterises the transaction workload that the virtual machines receive? And second, is it possible to build a performance model and the corresponding evaluation of the virtual machines running that service?*

As soon as these questions are answered, newer questions may arise as to how to improve the service through tuning and not upgrading the current hardware.

Therefore, applying the method for solving a performance problem and performance evaluation phases described in [3] and [4], in this case study we proceed as follows:

1. Understand the system to be evaluated (scenarios monitoring).
2. Characterize the current workload of the system.
3. Build a workload model.
4. Collect data and parameters of a performance model.
5. Build a performance model.
6. Evaluate the performance model.
7. Analyze the results of the performance evaluation.
8. Determine the current capacity and future capacity planning.
9. Propose options for tuning or upgrading the system.
10. Evaluate the modified performance model.

Thus, the detailed contributions of this performance case study are mainly:

- The current workload characterization of the transactions arriving at the VMs and their clustering in different transaction classes that are not currently differentiated at the SaaS supplier.
- The analytical and simulation models and their corresponding evaluation with the current workload characterization to determine the residual capacity of VMs. The models are extended to consider the new transaction clustering when stressing the VM capacity.
- The example of forecasting study of workload scenario to see the possible seasonality of the transactions and their relationship with their characterization and the proposed clustering.
- The study of the possible tuning of the current architecture estimates the overhead saved or added by either adding or subtracting one VM per physical machine (PM) or even replacing VMs with containers and its estimated effects on the performance of the platform.

Therefore, the remainder of the paper is organized as follows. Section II details the closer related work on queuing system cases to address the problem of providing performance evaluation with VMs. In Section III, the case-study scenarios are presented. This section corresponds with the first phase of the performance evaluation method. Section IV overviews the experimental setup. Section V is devoted to workload monitoring and characterization, covering phases second and third of the methodology. Section VI presents our modelling proposal to design the transactional system with a very simple queueing network. We compare the real workload and the queueing model mean response times, and we show the capacity planning by stressing the queueing network model, i.e., the VMs utilization. This corresponds to phases four to seven in the methodology for evaluating the performance of the transactional system. In Section VII, we show a scenario example of the possible forecasting of the transactional system (phase number eight). Section VIII is devoted to estimating the tuning effects on the original set of VMs with other similar configurations and estimating the overheads, ending the methodology for phases nine to ten. The discussion is in section IX. Finally, section X outlines the main conclusions and future work.

II. RELATED WORK

The problem of cloud computing performance modelling considering the quality-of-service metrics as response time has been extensively studied in [5]. In [6], the authors obtained the response time distribution of a cloud system modelled on a classical open M/M/m network, assuming an exponential density function of arrival interval and service time. Response time distributions were used to determine the optimal service tier and the ratio of the maximum number of tasks to the minimum number of resources (VMs). Response time considers both wait time in queue and service time. For a particular service resource, the author has determined the level of her QoS service that can be guaranteed in terms of response time. In [7] the authors extend the e-health model developed in [8]. The nodes that form the cloud architecture can be analysed individually if they form an open Jackson network. They only consider M/M/1 and M/M/m nodes because they assume that the analysis constraints arrival and service rates have only exponential distributions. Work [7] extends the model presented in [9] by modelling a cloud architecture instead of web servers. The model should also serve as a guide for creating/deleting VMs as in [10] or as a guide for investigating the root cause of bottlenecks and providing solutions as in [11]. In [10], the author presented an integer programming mechanism that automatically scales compute instances to cloud-based workload information and performance requirements. It was based on an activity plan for starting and stopping VM instances. This mechanism allowed cloud applications to complete submitted jobs on time by controlling the number of underlying instances and reducing user costs by choosing the right instance type.

In [12] the performance study about a monitoring system for IoP (Internet of People), the integration of chips inside people that link to other chips and the Internet, is evaluated using an M/M/c/K queuing network. The authors then propose a queueing-based model to evaluate the performance of fog-supported IoP systems. They first develop a mathematical model of a fog node that considers the processing time, service time, and arrival rate of data requests. They then extend this model to a multi-fog-node scenario, considering the transmission time and delay between fog nodes. Finally, they evaluate the performance of the proposed model through simulation experiments. The paper "Performance evaluation of message routing strategies in the Internet of robotic things using the D/M/c/K queuing network" by Feitosa et al. [13], focuses on evaluating the performance of message routing strategies in the Internet of Robotic Things (IoRT). The authors begin by discussing the challenges of communication in IoRT and the need for efficient message-routing strategies. They then introduce the D/M/c/K queuing network as a tool for evaluating the performance of these strategies. The article [13] focuses on the use of Internet of Things (IoT) sensor networks in smart buildings and uses queuing models to evaluate their performance. The authors then introduce queuing models to evaluate the performance of IoT sensor networks in smart buildings. They explain how queuing models work and how they can be used to simulate different scenarios and evaluate the performance of the sensor network under different conditions. In [14], the authors present a study on the performance of an Internet of Healthcare Things (IoHT) system for medical monitoring, using M/M/c/K queuing models. The IoHT system consists of a network of interconnected devices and sensors that collect and transmit healthcare data to a central server for analysis and decision-making.

Our research work is a performance engineering case study covering not only the queueing modelling and evaluation but earlier phases of workload monitoring and characterization and posterior phases such as the forecasting and tuning phases. Particularly, our case study is built upon monitoring the VMs at the cloud platform and its workload characterization during several periods. From the monitoring and workload characterization and clustering, we build simple queuing network models for each VM. The network is evaluated through approximate analytical models (relaxed to exponential queues) and discrete-event simulation.

The queueing network model also permits the study of the possible capacity planning future workload and anticipates several VM changes tuning the number of VMs or replacing them with containers. The queueing network model allows the cloud system to be scaled optimally to guarantee the quality of service for the transaction mean response time and plan the proper deployment and removal of VMs or virtual processors according to the workload. To comply with the mean response time expected, the cloud platform should be able to add/remove VMs according to the results obtained by approximate analysis or simulation. We summarize and compare some of the related works with ours in table 1.

III. TRANSACTIONAL SYSTEM FOR FLIGHT SEATS AVAILABILITY SERVICE

The flight seat availability algorithm for an airline carrier runs in an IT supplier datacenter with their PMs hosted in the cloud. In particular, the two PMs are Xeon Gold 6148 processors at 2.40 GHz as IaaS. These PMs are hosting four VMs each, to handle the transactions (availability requests) that come from different organizations and flight business intermediaries, such as Global Distribution Systems (GDS), to the flight carrier. Thus, the seat availability requests arrive in a traditional B2B relationship between the business supplier (the airline), the IT supplier (the datacenter), and the SaaS provider (the flight seats availability service) to connect to other intermediaries as input of other services, e.g., selling aircraft seats to the final customers.

Mainly, for each flight seat request, a thread is launched to a vcore (virtual core) in the VM that attends to that request, the service could be summarized as:

- getting data from persistence in RAM, mainly moving data from RAM to vcore;
- loading local caches in the thread (move data from RAM to TLS (Thread Local Storage) to some object caches for later use;
- executing linear record search operations or using Hash Maps: Operations $O(n)$ or $O(n^2)$ and even $O(1)$, date/time operations, and simple arithmetic operations including a few trigonometric to calculate miles.

Regarding the cloud platform at the IT supplier, one PM hosts four VM, servicing transactions. To guarantee VM availability, there is another twin set of four VM in another PM, that is, the service consists of a total of eight VM, in two sets of four VM each. Each VM consists of 16 vcores, one of which hosts the operating system (in a type II virtualization architecture), and the other 15 handle business transactions in parallel. Figure 1 shows a graphical schematic representation of PM hosting four VM and their 15 service vcores.

IV. EXPERIMENTAL SETUP

To monitor the workload of the four VMs consolidated in both PMs, the Telegraf agent[1] is used, which allows the collection of all the metrics of each VM of the flight availability service

[1]https://www.influxdata.com/time-series-platform/telegraf/

TABLE 1. Comparison among some related research works.

Ref.	Research Type	Scope	Performance Analysis
[6]	Case-study	Cloud-based e-commerce application	Framework to evaluate the performance of the application under various workload conditions.
[7]	Technique	Fog computing system	Queuing model considers a fog computing architecture with multiple fog nodes that serve a set of users generating requests for computation and storage resources.
[8]	Modelling	Cloud computing in e-health.	Different types of cloud computing models and the benefits of each.
[9]	Modelling	Web server.	A client model that generates requests according to a specified load pattern, a server model that processes those requests, and a performance model that measures the response time and throughput of the server under different conditions.
[10]	Algorithm	Cloud system scalability	Dynamically adjust the number of virtual machines (VMs) allocated to a particular workload based on the workload's resource demand, budget, and deadline.
[11]	Modelling	Cloud system	Model for cloud computing based on queueing theory and analyzes the performance of the model using simulation.
[12]	Modelling	Internet of People (IoP) /Fog computing.	Two scenarios: a centralized fog computing system, where all requests are processed by a single fog node, and a distributed fog computing system, where requests are distributed among multiple fog nodes.
[13]	Modelling	Internet of Robotic Things (IoRT)	Queuing network model to analyze the performance of three message routing strategies: shortest path, multi-path, and load balancing.
[15]	Modelling	IoT sensor networks in smart buildings	Queuing model for IoT sensor networks in smart buildings and evaluating its performance under different scenarios.
[14]	Modelling	Internet of Healthcare Things (IoHT)	Queuing models to evaluate the performance of the IoHT system, which involves analyzing the arrival rate of patients, the service rate of the system, and the number of servers available to handle the patients.
This work	Case-study	Cloud IaaS Transactional System	Performance Evaluation based on workload monitoring and characterization, queuing model simulation, capacity planning, forecasting, and tuning of a transactional flight seats availability service in VMs

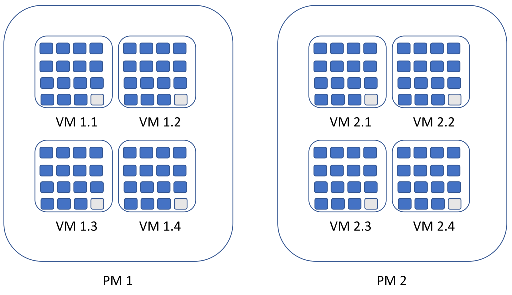

FIGURE 1. Organization of PMs, VMs, and vcores.

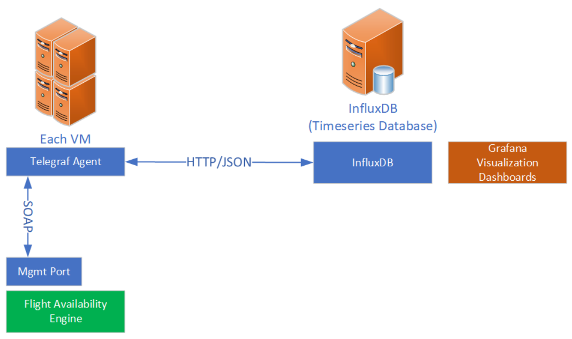

FIGURE 2. Agent monitoring the Flight Availability service.

engine. These data are obtained via SOAP and are sent to the time series database where they can be consulted with Grafana[2] (see figure 2). Regarding the k-means clustering of the monitored transactions we use Weka [16], and the Minitab [17] tool has been used for the analysis of the statistical distributions. Finally, we use the QNAP2 [18] discrete-event simulator and the corresponding analytical solver for the queue modelling and evaluation.

V. WORKLOAD MONITORING AND CHARACTERIZATION

The IaaS for processing flight seat availability can receive several million transactions per minute (TPM). The number of requests/s sometimes fluctuates seasonally and during the same day but other times not. In table 2, we collect three statistics of different monitoring periods that correspond to three different scenarios of requests/s received at one PM with four consolidated VMs. Their duration has been adjusted to the availability of PM monitoring.

We monitored the four VMs together, even though there are consolidated in the same PM and sharing all the hardware and software resources, i.e., without any distributed allocation or network or intercommunication among them. On the one hand, the three scenarios present very different mean response times and in the case of the first scenario the standard deviation is very high. On the other hand, it is confirmed that

[2]https://grafana.com/

TABLE 2. Requests arrival statistics of a PM (sampling).

Period (Scenario)	Harmonic Mean Requests/s	Mean Interarrival time (ms)	VM Mean Response time (ms)	VM Response time standard deviation	Inter-VM Response time standard deviation	VM Mean Utilization (%)	Inter-VM Utilization standard deviation (%)
1	6793.4782	0.1472	8.791006	9.194052	0.000433	34.51	0.017853
2	3696.8576	0.2705	2.876544	0.415900	0.000500	16.57	0.004330
3	2274.7952	0.4396	3.074868	0.413745	0.001089	11.41	0.008291

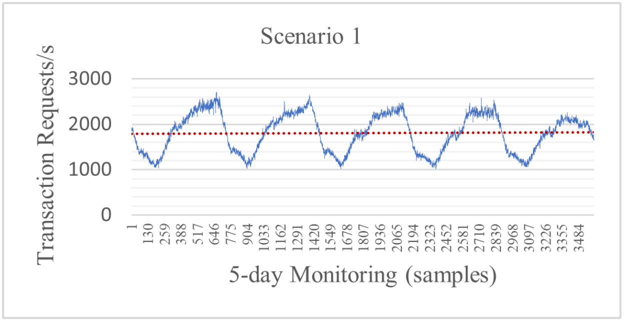

FIGURE 3. Requests/s monitored during scenario 1.

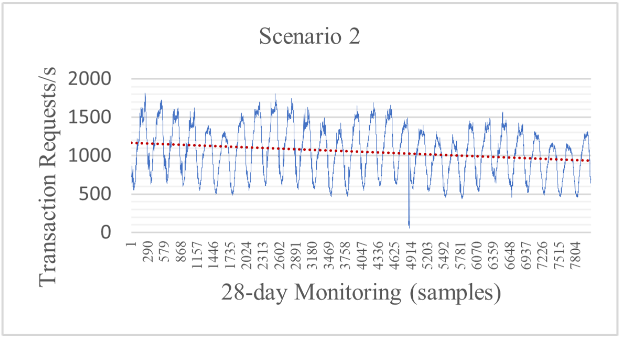

FIGURE 4. Requests/s monitored during scenario 2.

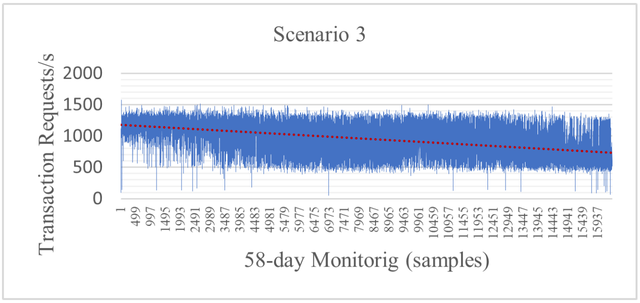

FIGURE 5. Requests/s monitored during scenario 3.

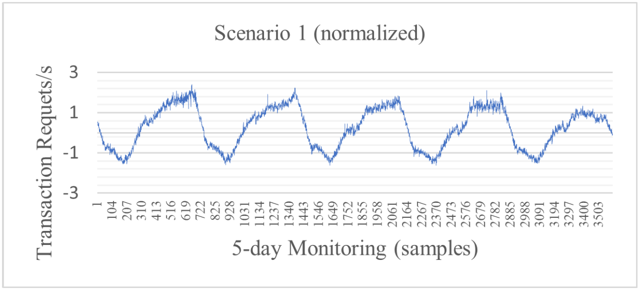

FIGURE 6. Normalized requests/s monitored during scenario 1.

the four VMs consolidated in the same PM behave similarly due to the uniform sharing of transactions since the standard deviation between the four VMs is almost negligible for both the transactions' mean response time and the VMs' mean utilization.

A. VM WORKLOAD MONITORING

Taking any VM of the four consolidated at one PM, the observed behaviour explains the differences between scenarios (see table 3). The three scenarios could be considered different because of the variance in requests/s for a VM. So, the standard deviation of the interarrival times of requests should be considered for the workload characterization. Figures 3 to 5 show the profile of these requests/s for the three scenarios.

In figure 3, the cycles of the graph correspond to the five days of the duration of the observation period of scenario 1. In the same way, in figure 4, each peak and valley correspond to a calendar day of scenario 2 from 28 days. However, for scenario 3, there is no such calendar pattern or seasonality (see figure 5) for 57 days. This can be demonstrated by normalizing each monitoring data (subtracting the mean requests/s and dividing by their standard deviation). We took all the combinations of consecutive five-day periods in scenarios 2 and 3, taking scenario 1 as the cycle to compare.

In figure 6, normalized scenario 1 is shown, whereas the first five-day period of scenarios 2 and 3 are shown in figures 7 and 8, respectively. With these paradigmatic examples, it is observed that scenarios 1 and 2 have a similar pattern of arrival requests/s while scenario 3 is different.

B. VM RESPONSE TIME MONITORING

Once the transaction request arrives at a VM, its service consists of two consecutive phases: processing and sending. Additionally, a transaction can remain stopped by a request scheduler (delayed) until its service can be carried out, that is, the transaction processing and sending. The resulting response time of a transaction executed in a thread by a vcore is the service time plus the delay time. The delay time of a transaction is not a queue time of the transaction waiting for its service as we understand the traditional continuous service of a FIFO queue. This can be verified in the following workload scenarios. The delay time corresponds to a different

TABLE 3. Requests arrival statistics of a VM (sampling).

Period (Scenario)	Days/ Samples	Harmonic Mean Requests/s	Mean Interarrival time (ms)	Interarrival time standard deviation (ms)	Interarrival time Coefficient of Variation
1	5/3600	1698.1852	0.5888	0.1510	0.2565
2	28/8068	924.1460	1.0820	0.5763	0.5326
3	57/16415	568.3772	1.7593	0.5265	0.2993

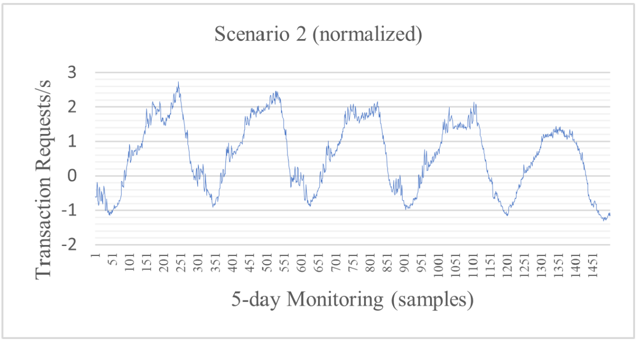

FIGURE 7. Normalized requests/s monitored during scenario 2 (example of five days).

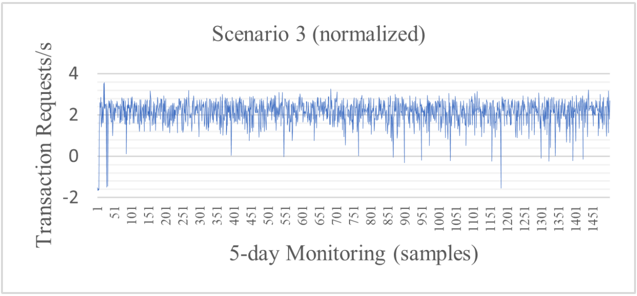

FIGURE 8. Normalized requests/s monitored during scenario 3 (example of five days).

cumulated time due to scheduling or any other VM activities before transactions are queued to receive processing and sending times in a thread executed in a vcore. Between the processing and the sending times, there is no waiting time. So that as soon as a request of flight seat availability arrives at one of the four VMs the transaction request is delayed sometime due to some scheduling actions (and maybe other software cumulated delays) to determine the assigned vcore, that executes the transaction service (the flight seat availability) after some queueing at vcore processor if there would be high vcore utilization.

In table 4 we show the monitoring of one of the threads at vcores in a VM for a transaction in the three scenarios. Firstly, there does not seem to be any noticeable difference in the sending service, since the mean and standard deviation of the three scenarios are very similar and their values are smaller in comparison with process times. Secondly, the processing time is very similar in the three scenarios with a higher standard deviation for scenario 1. However, the biggest difference among the three samples is the delay time. Even though scenarios 2 and 3 have similarly low values compared again to scenario 1, the latter has a lot of variances.

C. WORKLOAD STATISTICAL DISTRIBUTION FITTING

To build even the simplest queueing model and perform the statistical analysis, it is necessary to know the distribution of the arrivals, i.e., the requests/s (or the corresponding interarrival times), the delay and the service (process and sending) times. To avoid inaccurate guessing of any distribution, we may try to confirm whether a certain distribution fits the four monitored timed variables in all the monitored scenarios. Unfortunately, the distribution fitting of the probability distribution to the series of arrivals, i.e., requests/s and the delay, process and send times were not valid for the data of any scenario. In appendix tables A1 to A4, we provide the distribution fitting test [17] of the requests/s, process, delay, and sending times, respectively, in scenario 1 (the rest of the scenarios and delay and sending times are similar). Since the p-value is lower than 0,05 in all four tables, we cannot conclude the requests/s, delay time, process time, and sending time fit with well-known distributions.

D. WORKLOAD AND RESPONSE TIME CLUSTERIZATION

Even if the distributions do not fit, the monitorization of the three scenarios shows some features that may conduct a more detailed classification of the transactions:

- there is an enormous variance in scenario 1 in mean process time and mean delay time;
- scenarios 1 and 2 are similar in arrival frequencies to each VM;
- scenarios 2 and 3 are very similar in mean service times (process plus sending times) and delay time;
- scenario 3 is very different in the frequency of arrivals from the other two;
- scenario 1 is very different in the delay time (and somewhat in the service time).

By applying clustering to the three scenarios, we may represent different classes of transactions by selecting the centroids of the clusters as representatives of a respective class. These classes were not distinguished in the original monitoring. If simple k-means are applied to the three scenarios, the following clusters can be determined based on the Euclidean distance from their centroids [16]. We transform the request/s original data to interarrival times, i.e., transform average frequencies to mean interarrival times to facilitate operationalization.

TABLE 4. Mean delay, process, and sending times of a transaction in a VM.

Period (Scenario)	Mean Process time (ms)		Mean Delay time (ms)		Mean Sending time (ms)	
	mean	sd.	mean	sd.	mean	sd.
1	2.9241	0.8133	5.7427	8.4426	0.1241	0.000033
2	2.5658	0.3169	0.1857	0.1461	0.1249	0.000031
3	2.7505	0.3157	0.1939	0.1257	0.1303	0.000032

Therefore, the monitorization of scenario 1 could include 3 different clusters where the major differences come from the interarrival time, process time, and delay time, respectively, whereas the sending time is almost common to every cluster (see table 5). Selecting more than 4 clusters would produce two or more clusters closer than the rest, and selecting two clusters are too simplistic to characterize scenario 1. Particularly centroid 1.2 represents transactions with more mean process time and mean delay time compared to clusters 1.1 and 1.3. Whereas the delay time of centroid 1.3 doubles the one from cluster centroid 1.1. However, centroids 1.1 and 1.3 have a huge delay time variance compared to centroid 1.3.

In table 5, delay time appears as the variable with more standard deviation in any cluster.

To be coherent, scenario 2 is also classified into 3 clusters, even though the scenario is less variable (see centroids in table 6). Centroid 2.2 has more mean process, mean delay, and mean sending time than the two other clusters even though all three centroids are more similar in comparison with the previous scenario.

Then, scenario 3 is also classified into 3 clusters (see centroids in table 7) than the previous two. Centroid 3.2 is an outlier but important for clusterization since it has a higher mean interarrival time and mean delay time than the two other centroids.

With this data clusterization, we may conclude the following:

- The mean processing time for transactions is higher than 2.1 ms and exceptionally may arrive to double this value (even though the maximum almost arrives at 70 ms).
- The delay time is the variable that defines the transaction's performance.
- The sending time last usually slightly more than 0.1 ms but does not influence the performance of the transaction as the rest of the variables and it is very stable compared with process time and particularly with delay time.

VI. QUEUE MODELLING AND CAPACITY PLANNING

The cloud platform at the datacenter consists of two sets of four VMs. The queuing model is shown in figure 9. Each PM is modeled as an open network [19] where requests are generated by a source queue of customers (transactions), arriving at a H/H/∞ delay queue that sends them in a uniform distribution scheduling. Each VM is a multiserver of 15 vcores. The vcores are modelled as a H/H/15 queue, solved with a discrete-event simulator. All H distributions were assumed hyper exponential with the mean and the standard deviation

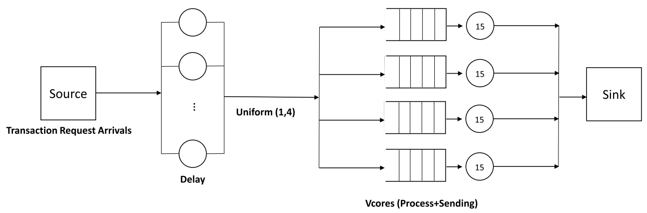

FIGURE 9. One PM modelled as a queue network with transactions as customers.

shown in tables 2, 3 and 4. The hyper-exponential distribution permits to parametrize of both central statistics.

Table 8 shows the comparison between the mean response time measured in the PM with four consolidated VMs and the mean response times of the four simulated VMs. The results show very similar behaviour to the four VMs (results expected by the distribution of the workload), with low relative errors concerning the measured mean response time and negligible differences between VMs.

Since the performance measurements and simulation seem to confirm the similarity among the four VMs consolidated at the same PM sharing uniformly the workload, we concentrate the rest of the case study in one VM but considering the clustering built to distinguish different customer classes in the queueing model.

Thus, each VM is modelled with an open network where requests are generated by a source queue of multiclass customers (transactions) weighted as the data cluster frequencies, arriving at a H/H/∞ delay queue that sends them to vcores. The vcores are modelled as a H/H/15 queue, solved with a discrete-event simulator (relaxed, in general, to exponential queues for approximate analytical solutions). To compare results against the discrete-event simulation approximations, we also model a VM with exponential queues for a unique class customer source and the exponential service of vcores, even for the delay time the hyper-exponential distribution remains used. Thus, the delay queue was modelled as an M/H/∞ and the vcores as H/M/15 with two consecutive exponential services (process and sending times). Assuming that each cluster centroid is representative of a customer class, the multiclass queue model is shown in figure 10.

The queueing network model was simulated with a confidence interval of 95% for all queues and more than 10 million transaction instances (customers) were simulated per scenario, even though results converged with just thousands of customers, like the number of samples in monitoring data.

TABLE 5. Cluster output of scenario 1 through k-means.

	Cluster Instances	Interarrival time (ms)		Process time (ms)		Delay time (ms)		Sending time (ms)	
		mean	sd.	mean	sd.	mean	sd.	mean	sd.
1.1	1218	0.7771	0.0776	2.3797	0.4156	1.2674	1.9050	0.1033	0.0252
1.2	827	0.4413	0.0314	4.1366	0.5038	18.6155	8.7789	0.1320	0.0273
1.3	1555	0.5199	0.0593	2.7058	0.4467	2.4021	2.4939	0.1362	0.0330

TABLE 6. Cluster output of scenario 2 through k-means.

	Cluster Instances	Interarrival time (ms)		Process time (ms)		Delay time (ms)		Sending time (ms)	
		mean	sd.	mean	sd.	mean	sd.	mean	sd.
2.1	3030	1.0276	0.3874	2.5825	0.1416	0.1853	0.0677	0.1271	0.0177
2.2	2253	0.8810	0.4093	2.9595	0.1386	0.2555	0.2397	0.1593	0.0225
2.3	2785	1.3041	0.7607	2.2292	0.1214	0.1299	0.0580	0.0949	0.0169

TABLE 7. Cluster output of scenario 3 through k-means.

	Cluster Instances	Interarrival time (ms)		Process time (ms)		Queue time (ms)		Sending time (ms)	
		mean	sd.	mean	sd.	mean	sd.	mean	sd.
3.1	8999	1.3136	0.4351	2.5082	0.1622	0.1465	0.0596	0.1108	0.0187
3.2	32	7.3478	4.0432	3.0473	0.8254	1.5296	1.4839	0.1235	0.0252
3.3	7384	0.9819	0.3021	3.0446	0.1680	0.2459	0.0916	0.1543	0.0236

TABLE 8. Response times of a PM (monitoring and simulation).

Scenario	Measured Response time (ms)	VM Mean Utilization	VMs Simulation Mean Response times (ms)			
	mean	%	VM1	VM2	VM3	VM4
1	8.7910	34.51	8.9760	8.9760	8.9760	8.9750
2	2.8765	16.57	2.8750	2.8750	2.8760	2.8760
3	3.0748	11.41	3.2040	3.2070	3.2060	3.2060

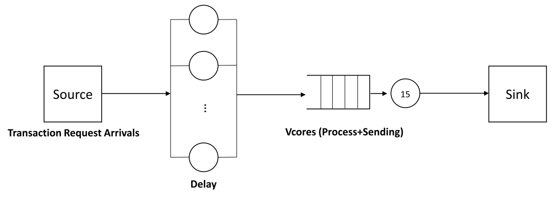

FIGURE 10. One VM modelled as a queue network with transactions as customers.

No exact analytical or numerical solution was applicable but the convolution method was applied to have an analytical approximation. In table 9, we show the mean response times of a VM in the three scenarios. The queuing model approximations work well in discrete-event simulation and in the analytical approximation since errors compared with real data are less than 0.018% in any case, except in scenario 3 but less than 4.24%.

For the data cluster centroids taken as representatives of customer classes, no analytical or approximation for a multiclass model was possible even relaxing disciplines or distributions, so we show the discrete-event simulation results in table 10. Reducing the variance intra-cluster and increasing the variance inter-cluster produces less % relative error per class, compared with the mean response times of a unique transaction class. Moreover, in scenario 3, where the outliers are isolated in class 3.2, the model reduces the aggregated class relative error in the simulation results, ranging from -4.232 % to -0.004 %. However, in scenario 2, the aggregated class relative error increases from 0.018 % to 1.374 % and, in scenario 1 aggregated class simulation results are equal.

The utilization of one VM in the three scenarios during monitoring was 34.51%, 16.57%, and 11.41%, respectively. Since the 4 VM share the workload in each PM, there is still room to receive more requests/s, until PM saturation. If the arrival frequencies monitored in the three scenarios were increasing, capacity planning would be like the ones shown in figures 11 to 13. In these figures, the arrival frequency of transactions has been increasing from 10% to 90%, close to model saturation. The mean response times are almost equal to the service times due to the buffer effect of the scheduler and its corresponding delay time until the utilization of the VM is higher than 70%.

VII. FORECASTING

Scenarios 1 and 2 are seasonal, each day they have a peak and valley with the same shape, although have different intensities. When normalized, both scenarios look very similar. The trend lines are slightly increasing or decreasing but do not provide much information since they are averages (see figures 3 to 8).

To predict the requests/s of these two scenarios, observing the daily transaction response times, we use moving averages to predict the approximate number of requests/s in the

TABLE 9. Response times of a VM (monitoring, simulation, and analytical approximation).

Scenario	Measured Response time (ms)	VM Utilization	Simulation Response time (ms)		Analytical Response time (ms)	
	mean	%	mean	% error	mean	% error
1	8.7910	34.51	8.7920	0.011	8.7910	0.000
2	2.8765	16.57	2.8760	0.018	2.8770	0.015
3	3.0748	11.41	3.2050	-4.232	3.0750	-0,004

TABLE 10. Response times of a multiclass VM (monitoring, clustering, and simulation).

Cluster (Scenario)	Measured Response time (ms)	VM Utilization (computed)	Simulation Response time (ms)	
	mean	%	mean	% error
1.1	3.7504	11.67	3.7520	-0.0420
1.2	22.8841	7.93	22.890	-0.0257
1.3	5.2441	14.91	5.2440	0.0019
(1)	8.7910	34.51	8.7920	-0.0113
2.1	2.8949	6.22	2.7860	3.761
2.2	3.3743	4.62	3.3780	0.109
2.3	2.454	5.73	2.4540	0.000
(2)	2.8765	16.57	2.8370	1.374
3.1	2.7655	6.25	2.7650	0.018
3.2	4.7004	0.02	4.7140	-0.289
3.3	3.4448	5.14	3.4460	-0.034
(3)	3.0748	11.41	3.0750	-0.004

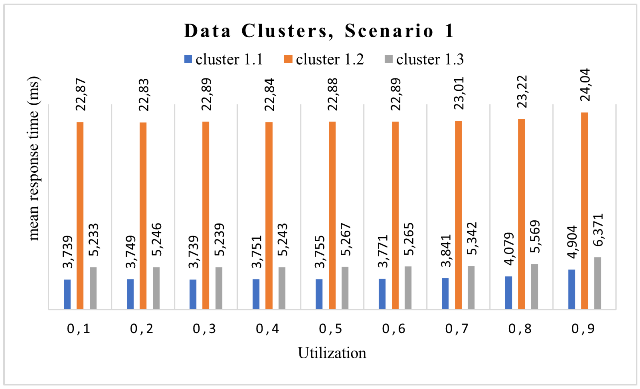

FIGURE 11. Capacity planning for scenario 1.

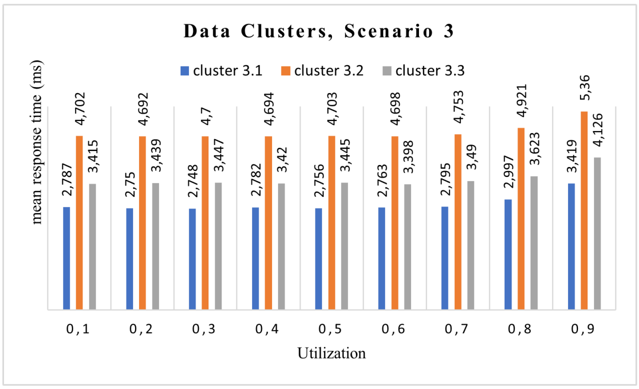

FIGURE 13. Capacity planning for scenario 3.

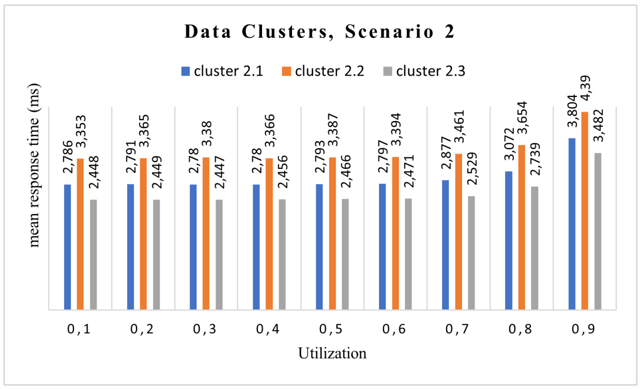

FIGURE 12. Capacity planning for scenario 2.

following instants. This is quite predictable for scenario 1 (see figure 6). In the case of scenario 2, its high coefficient of variation makes the absolute value of immediate future requests/s less predictable. In the case of scenario 3, there is no daily seasonality but the coefficient of variation is close to 30%, and the average starts to lose meaning.

As an example, in figure 14, the individualized response time samples of each scenario 1 have been represented and their corresponding trend through moving averages, with periods of 90 samples. As in requests/s, there is the same seasonal behaviour for sample transaction response times. Thus, the mean and the standard deviation of response time to depict the performance behaviour using just one class of transactions could lose descriptive meaning, not only for the performance evaluation of the VM but also forecasting. This is the reason why we clustered the transactions depending on the four variables monitored. The moving averages with 90 samples period seem to confirm that consider three mean response time centroids with values computed in table 11, for scenario 1, representing the averages of valleys (1.1), peaks (1.2), and ascending and descending values (1.3),

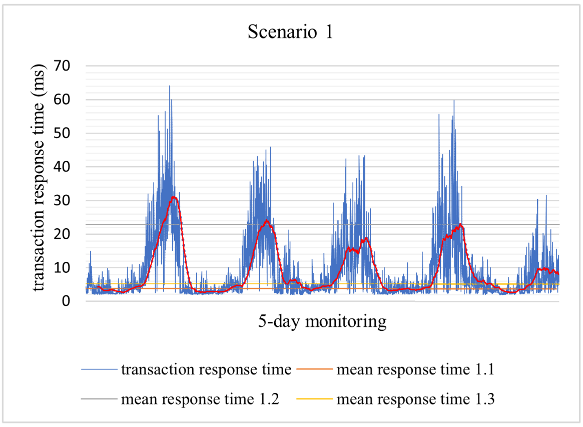

FIGURE 14. Transaction response times (blue) through moving averages (red) of period 90.

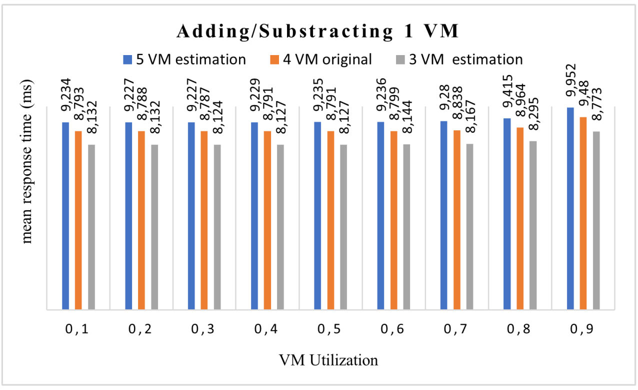

FIGURE 15. Comparison of adding or subtracting 1 VM in scenario 1.

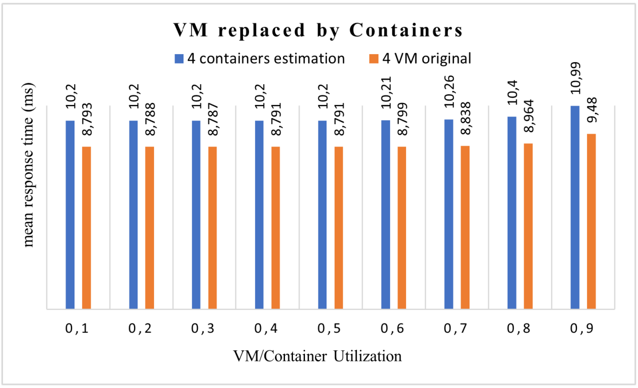

FIGURE 16. Comparison of replacing VMs by containers in scenario 1.

respectively, are good predictors of three-class transactions performance.

VIII. TUNING VM OVERHEAD

Virtualization overhead is an important factor due to the current trends of consolidating servers [20]. The consolidation degree will determine the performance degradation (amount of overhead). Moreover, note that the CPU is the most demanding server device [21]. In [22] determined the overhead classes of server consolidation in PMs. The first one class is the virtualization by the hypervisor, and the second class is the virtualization due to consolidation. These two another overhead classes are in any server independently of the physical server features, hypervisor type and type of user-executed workload. We experimented with several PMs to find out the percentage of overhead that virtualization produced when consolidating servers. That is, the amount of time that was added (overhead) to the execution of transactions, consuming CPU and RAM, as in this case of study, in several VM compared with their execution in the same number of PMs, without virtualization. We named the two types of overhead in servers' consolidation: the overhead due to virtualization itself (Ov) and overhead due to interaction (Oc) with other VM consolidated on the same server (physical or virtual, since they can be nested). In [23] also determined overheads for several virtual machines and/or container combinations and nesting. Therefore, the performance of the two sets of VMs in both PMs, in our study, is also affected by Ov and Ov. The more overhead the less productive work at VMs and then the quality of the service is reduced.

Precisely, for type II virtualizations and servers with a similar number of CPUs, the study [23] indicated an Ov of approximately 2% and an Oc for 4 VM consolidated in the same PM of 72% to 79%. In other words, consolidating 4 VMs in the same PM had a price of more than 80% overhead in comparison to saving 3 PMs in hardware, space, and electrical power consumption, among other gains from consolidating 4 VM (and other losses as energy consumption).

In the case of the flight seat availability service, although there are some differences in the hypervisor used and the number of physical CPUs with the servers used in [22] and [23], we may assume that the amounts of Ov and Oc would be similar. In the same way, in [22] the number of VM consolidated by PM was varied to see the increase or decrease of both types of overhead. Going from consolidating 4 VM to 5 VM and consequently redistributing the workload uniformly, decreased the Ov to 1% but the price to be paid was increasing the Oc by an additional 5%. However, going to 3 consolidated VM increased the Ov to 2-3% but decreased the Oc by 9% approximately. All these variations were in comparison with no virtualization scenarios.

With all these previous experiments done to estimate both overheads, we simulate new sets of VMs in our case study. Figure 15 shows the comparison of adding or subtracting one VM from the original configuration set of consolidating 4 VM in scenario 1, estimating the variation of Ov and Oc and the uniform sharing of request/s workload among the number of VMs. The mean response time with a set of 3 VM permits saving around 0.65 ms, in opposition to a set of 5 VM that would add around 0.45 ms to the mean response time for all utilizations. Of course, having 3 VM will decrease the elasticity and the security and even the energy trade-off [24].

Regarding containerization for servers (in the example we used Dockers), the studies [22], [23] indicated an OV of approximately 20-22% and an Oc for 4 containers consolidated in the same PM of 74%, comparing against having 4 PM without containerization. Figure 16 shows the comparison of replacing the original set consolidating 4 VM configurations in scenario 1 by four containers, estimating the variation of OV and O_c. Simulating the estimated containerization of the original set, the mean response time with a set of 4 containers adds around 2.5 ms because of the additional software layers to implement, even though this may vary depending on the nature of the transactional service.

IX. DISCUSSION

This case study is based on the useful real data we obtained from the flight seat availability service company between the users (GDS and other intermediation organizations) and the flight carrier. These data are request/s, delay, process, and send times. The data were sufficient for the case study but having more information from the PM and VM provider (IT supplier) for example about the configuration of the hypervisor, would precise some information, e.g., for the overhead estimations. These are some main limitations of this case study.

Regarding, the scenarios' duration and samples, we have been to accept what we got, since the monitoring overhead may reduce the performance not only of the real B2B transactions but also from the personnel at the case service provider company. The same applies to the number of scenarios. In future works, we should analyze additional scenarios if available.

Regarding the queueing network model, we know that a queue with blocking before service [25] at delay time should be a possible analytical solution but the logic of the transaction scheduler is IP protected and we cannot guess any probability guessing for unblocking the transactions at scheduler. However, the infinite queue as a delay works well when comparing mean response times monitoring and experimentation. The data clusterization of transactions and the corresponding multiclass customer queue network simulations reduced the errors in comparison with mono-class customer simulations. However, the original transactions are not classified, so we did not have feedback about the precision of our centroids as part of a preemptive modification of scheduler and differentiated routing for transaction classes.

In regards to tuning the number of VM or substituting with containers, the case study uses other configurations of PM and hypervisors so estimations are also based on previous research works. We should compare our estimations with the same configurations at the service provider, to confirm our guessing, even though the method of estimating the overhead is applicable.

X. CONCLUSION

In this article, we addressed a complete case study for the performance evaluation of a real-world B2B implemented with VMs, a flight seat availability service for a flight carrier. This case study reveals, even though we do not have precise information from the hardware and hypervisor data from the IT supplier, the monitoring of transaction arrival frequency to VMs and their execution times at VMs from the service provider is sufficient to characterize, model and tune the transaction journey in the VM platform. The provided case study is especially useful not only for this flight seat availability service but also for any commercial services based on CPU and memory transactions in consolidated VMs, as in the tourism industry.

We follow a performance engineering methodology fully established and cited in the literature. We use workload characterization, clustering, performance evaluation with queueing network discrete-event simulation, capacity planning, and forecasting in this work. We also show that analytic models may be used to compute mean response times, relaxing the queueing distribution disciplines with customers without classification. The case study also uses previous research works to estimate the overhead for tuning the original set of VMs and their possible containerization.

TABLE 11. Distribution fitting for requests/s in scenario 1.

Distribution	AD	P	LRT P
Normal	69,370	$<$0,005	
Box-Cox transformation	69,370	$<$0,005	
lognormal	89,013	$<$0,005	
3-parameter lognormal	69,466	*	0,000
Exponential	961,345	$<$0,003	
2-parameter exponential	293,059	$<$0,010	0,000
Weibull	62,266	$<$0,010	
3-parameter Weibull	74,900	$<$0,005	0,000
smallest extreme value	55,863	$<$0,010	
Extreme value by maxima	91,054	$<$0,010	
gamma	80,569	$<$0,005	
3-parameter gamma	73,131	*	0,000
Logistics	70,939	$<$0,005	
log-logistics	86,699	$<$0,005	
3-parameter log-logistics	70,988	*	0,000

TABLE 12. Distribution fitting for process time in scenario 1.

Distribution	AD	P	LRT P
Normal	72,928	$<$0,005	
Box-Cox transformation	25,624	$<$0,005	
lognormal	33,357	$<$0,005	
3-parameter lognormal	20,832	*	0,000
Exponential	900,718	$<$0,003	
2-parameter exponential	162,198	$<$0,010	0,000
Weibull	77,052	$<$0,010	
3-parameter Weibull	13,016	$<$0,005	0,000
smallest extreme value	156,118	$<$0,010	
Extreme value by maxima	31,799	$<$0,010	
gamma	42,837	$<$0,005	
3-parameter gamma	12,748	*	0,000
Logistics	60,968	$<$0,005	
log-logistics	35,167	$<$0,005	
3-parameter log-logistics	27,888	*	0,000

TABLE 13. Distribution fitting for delay time in scenario 1.

Distribution	AD	P	LRT P
Normal	379,633	<0,005	
Box-Cox transformation	52,428	<0,005	
lognormal	52,428	<0,005	
3-parameter lognormal	40,568	*	0,000
Exponential	627,890	<0,003	
2-parameter exponential	769,506	<0,010	0,000
Weibull	54,736	<0,010	
3-parameter Weibull	31,728	<0,005	0,000
smallest extreme value	493,328	<0,010	
Extreme value by maxima	301,475	<0,010	
gamma	76,036	<0,005	
3-parameter gamma	43,796	*	0,000
Logistics	311,208	<0,005	
log-logistics	54,808	<0,005	
3-parameter log-logistics	43,887	*	0,000

TABLE 14. Distribution fitting for sending time in scenario 1.

Distribution	AD	P	LRT P
Normal	11,920	<0,005	
Box-Cox transformation	1,378	<0,005	
lognormal	5,803	<0,005	
3-parameter lognormal	11,818	*	1,000
Exponential	923,771	<0,003	
2-parameter exponential	789,128	<0,010	0,000
Weibull	34,330	<0,010	
3-parameter Weibull	201,163	<0,005	1,000
smallest extreme value	202,091	<0,010	
Extreme value by maxima	14,315	<0,010	
gamma	1,590	<0,005	
3-parameter gamma	1,336	*	0,002
Logistics	6,425	<0,005	
log-logistics	3,315	<0,005	
3-parameter log-logistics	6,387	*	1,000

Future work includes to have feedback on our insights in the real system to demonstrate what the experimentation in our laboratory concludes: flight seat availability service is far from performance saturation but some system tuning can improve the enormous variance in transaction response times (maximum response time is sometimes almost 10 times higher than the average). First, the predictability of some daily scenarios (like scenarios 1 and 2) may be considered to utilize more VMs in parallel in a different PM, in the IT supplier datacenter, for the local maxima or frequency peaks. Second, reducing the delay time produced by the transaction scheduler would have a huge reduction in mean response times (see scenario 1). Third, when there is no clear seasonality in the requests/s (scenario 3), the number of VM would be relaxed to take advantage of the elasticity based on cloud platform deployment. Finally, our data clusterization of transactions reveals that identifying the classes of requests may reduce drastically not only the modification of the scheduling (scenario 3), diverting expected longer transactions to priority VMs, but also managing the seasonality peaks when occurring (scenarios 1 and 2).

APPENDIX

This section is composed by Table 11, Table 12, Table 13 and Table 14.

REFERENCES

[1] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, "A break in the clouds: Towards a cloud definition," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, Dec. 2008.

[2] J. Varia, "Architecting for the cloud: Best practices," *Amazon Web Services*, vol. 1, pp. 1–21, Jan. 2010.

[3] R. Jain, "The art of computer systems performance analysis: Techniques for experimental design, measurement, simulation, and modeling," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 50–55, 1990.

[4] X. Molero, C. Juiz, and M. Rodeño, *Evaluación y Modelado Del Rendimiento de Los Sistemas Informáticos*. London, U.K.: Pearson, 2004.

[5] E. J. Ghomi, A. M. Rahmani, and N. N. Qader, "Applying queue theory for modeling of cloud computing: A systematic review," *Concurrency Comput., Pract. Exper.*, vol. 31, no. 17, p. e5186, Sep. 2019.

[6] K. Xiong and H. Perros, "Service performance and analysis in cloud computing," in *Proc. Congr. Services*, Jul. 2009, pp. 693–700.

[7] L. Mas, J. Vilaplana, J. Mateo, and F. Solsona, "A queuing theory model for fog computing," *J. Supercomput.*, vol. 78, no. 8, pp. 11138–11155, May 2022.

[8] J. Vilaplana, F. Solsona, F. Abella, R. Filgueira, and J. Rius, "The cloud paradigm applied to e-health," *BMC Med. Informat. Decis. Making*, vol. 13, no. 1, pp. 1–10, Dec. 2013.

[9] L. P. Slothouber, "A model of web server performance," in *Proc. 5th Int. World Wide Web Conf.*, 1996, pp. 571–576.

[10] M. Mao, J. Li, and M. Humphrey, "Cloud auto-scaling with deadline and budget constraints," in *Proc. 11th IEEE/ACM Int. Conf. Grid Comput.*, Oct. 2010, pp. 41–48.

[11] T. S. Sowjanya, D. Praveen, K. Satish, and A. Rahiman, "The queueing theory in cloud computing to reduce the waiting time," *Int. J. Comput. Sci. Eng. Technol.*, vol. 1, no. 3, pp. 110–112, 2011.

[12] L. Rodrigues, J. J. P. C. Rodrigues, A. D. B. Serra, and F. A. Silva, "A queueing-based model performance evaluation for Internet of People supported by fog computing," *Future Internet*, vol. 14, no. 1, p. 23, Jan. 2022.

[13] L. Feitosa, G. Gonçalves, T. A. Nguyen, J. W. Lee, and F. A. Silva, "Performance evaluation of message routing strategies in the Internet of Robotic things using the D/M/c/K/FCFS queuing network," *Electronics*, vol. 10, no. 21, p. 2626, Oct. 2021.

[14] B. Santos, A. Soares, T.-A. Nguyen, D.-K. Min, J.-W. Lee, and F.-A. Silva, "IoT sensor networks in smart buildings: A performance assessment using queuing models," *Sensors*, vol. 21, no. 16, p. 5660, Aug. 2021.

[15] B. Santos, A. Soares, T.-A. Nguyen, D.-K. Min, J.-W. Lee, and F.-A. Silva, "IoT sensor networks in smart buildings: A performance assessment using queuing models," *Sensors*, vol. 21, no. 16, p. 5660, Aug. 2021.

[16] G. Holmes, A. Donkin, and I. H. Witten, "WEKA: A machine learning workbench," in *Proc. Austral. New Zealnd Intell. Inf. Syst. Conf.*, 1994, pp. 357–361.

[17] B. F. Ryan, B. L. Joiner, and J. D. Cryer, *MINITAB Handbook: Update for Release*. Boston, MA, USA: Cengage Learning, 2012.

[18] M. Véran and D. Potier, "QNAP 2: A portable environment for queueing systems modelling," Ph.D. thesis, Institut Nat. de Recherche en Informatique et en Automatique, Paris, France, 1984.

[19] J. R. Jackson, "Jobshop-like queueing systems," *Manage. Sci.*, vol. 10, no. 1, pp. 131–142, Oct. 1963.

[20] N. Huber, M. von Quast, M. Hauck, and S. Kounev, "Evaluating and modeling virtualization performance overhead for cloud environments," *Closer*, vol. 11, pp. 563–573, Jan. 2011.

[21] B. Bermejo and C. Juiz, "Virtual machine consolidation: A systematic review of its overhead influencing factors," *J. Supercomput.*, vol. 76, no. 1, pp. 324–361, Jan. 2020.

[22] B. Bermejo and C. Juiz, "A general method for evaluating the overhead when consolidating servers: Performance degradation in virtual machines and containers," *J. Supercomput.*, vol. 2022, pp. 1–28, Jan. 2022.

[23] B. Bermejo and C. Juiz, "On the classification and quantification of server consolidation overheads," *J. Supercomput.*, vol. 77, no. 1, pp. 23–43, Jan. 2021.

[24] C. Juiz and B. Bermejo, "The *CIS*^{2}: A new metric for performance and energy trade-off in consolidated servers," *Cluster Comput.*, vol. 23, no. 4, pp. 2769–2788, 2020.
[25] H. G. Perros, *Queueing Networks With Blocking*. Oxford, U.K.: Oxford Univ. Press, 1994.

CARLOS JUIZ (Senior Member, IEEE) was born in Santiago de Compostela, Spain, in 1966. He received the Ph.D. degree in computer science from the University of the Balearic Islands (UIB), Spain, in 2001. He was a systems analyst in the private sector for more than ten years. He was a Visiting Researcher with the University of Vienna, in 2003, and a Visiting Associate Professor with Stanford University, in 2010. He is currently a Full Professor in computer architecture and technology with the Department of Mathematics and Computer Science, UIB. He is the coauthor of more than 200 international papers (including journals, published reviews and proceedings), 30 book chapters, and three books, and he has been very active in international conferences, where he was a program committee member on more than 400 occasions. He is heading the ACSIC Research Group (https://acsic.uib.eu), UIB. His research interests include performance engineering, green IT, and governance of IT. He is also a Senior Member of ACM and a member of the Domain Committee on Cloud Computing from IFIP, until 2017.

BARTOMEU CAPO received the M.Sc. degree. He is currently a Software Engineering Team Lead with Multinucleo, High-Performance Solutions. He has been working in the information technology industry for more than 20 years, including several roles in different business types, such as water distribution, the tourism sector (wholesalers and hoteliers), builders, and contractors. He is also focused mainly on software engineering oriented to airline product inventory and also help define a technological architecture for improving performance in high-volume traffic environments and high availability scenarios. He was a technical consultant for integrating different systems and services with different technologies. He has cross skills in different IT areas.

BELEN BERMEJO (Member, IEEE) was born in Merida, Spain, in 1992. She received the Ph.D. degree in computer science from the University of Balearic Islands (UIB), Spain, in 2020. She was a Lecturer in computer architecture and technology with the Department of Mathematics and Computer Science, UIB. She is the coauthor of more than 30 papers (including journals, book chapters, and proceedings). She is a member of the ACSIC Research Group, UIB. Her research interests include performance engineering, green IT, and cloud datacenters. She is also the Chair of the Women in Engineering Group, IEEE Spain Section.

ALEJANDRO FERNÁNDEZ-MONTES received the B.E. degree in computer science and the M.Tech. and international Ph.D. degrees in software engineering from the University of Seville, Spain. In 2006, he joined the Department of Computer Languages and Systems, University of Seville, as an Intern, and he became a Lecturer and an Assistant Professor, in 2013. In 2018, he became an Associate Professor (a Professor Titular de Universidad). In 2008 and 2009, he was invited to work with the École Normale Suprieure de Lyon (ENS de Lyon) in saving energy solutions for Grid5000 infrastructure. In 2012, he was invited to work with Universitat Politnica de Barcelona to share experiences in saving energy and doctoral methodologies. In 2016, he was invited to work with Shanghai Jiao Tong University, China, to initiate collaborations in distributed computing. He currently teaches and conducts research with the University of Seville, where he is also the principal investigator of several projects. His research interests include energy efficiency in distributed computing, applying prediction models to balance load, and applying ON/OFF policies to the internet data centers.

DAMIÁN FERNÁNDEZ-CERERO received the Ph.D. degree in computer science from the University of Seville, in 2018. He is currently an Associate Professor and a Researcher with the University of Seville. He was granted with a Marie Curie Postdoctoral Fellowship with Dublin City University. He has collaborated in several National and European Projects and with European research groups in multiple research stages. As a result of his research interests, he has published numerous high-impact peer-review journal articles in various international conferences.