Journal of Network and Computer Applications 142 (2019) 76–97

Contents lists available at ScienceDirect

Journal of Network and Computer Applications

journal homepage: www.elsevier.com/locate/jnca

An autonomous resource provisioning framework for massively multiplayer online games in cloud environment

Mostafa Ghobaei-Arani [a,d,**], Reihaneh Khorsand [b,*], Mohammadreza Ramezanpour [c]

[a] Department of Computer Engineering, Qom Branch,Islamic Azad University, Qom, Iran
[b] Department of Computer Engineering, Dolatabad Branch, Islamic Azad University, Isfahan, Iran
[c] Department of Computer Engineering, Mobarakeh Branch, Islamic Azad University, Isfahan, Iran
[d] Young Researchers and Elite Club, Qom Branch, Islamic Azad University, Qom, Iran

ARTICLE INFO

Keywords:
Autonomous resource provisioning
Massively multiplayer online games
ANFIS prediction model
Fuzzy decision tree
Cloud computing

ABSTRACT

Massively Multiplayer Online Games (MMOGs) applications are a class of computationally intensive client-server multi-tier applications with real-time quality of service (QoS) requirements. In these online games, the players all over the world can interact with each other through a virtual environment. To guarantee the QoS requirements to a highly variable number of concurrent players, the game operators statically over-provision a large infrastructure capable of sustaining the game peak load, even though a large portion of the resources is unused most of the time and subsequently leads to incur high resource provisioning costs on MMOG providers. To address this problem, we introduce an autonomous resource-provisioning framework for MMOGs in a Cloud environment. Firstly, a load prediction service anticipates the future game-entity distribution from historical trace data using an adaptive neuro-fuzzy inference system (ANFIS) prediction model. Then, we proposed a fuzzy decision tree algorithm to estimate the proper number of resources to be allocated to each tier in the MMOG application using the predicted workload and user SLA. The experimental results under real and synthetic workloads indicate that the proposed solution outperforms in terms of accuracy and performance metrics compared with the other approaches.

1. Introduction

With the rapid development of communication infrastructures and broadband Internet access, interactive multimedia applications are becoming increasingly popular. One class of these applications is online games, especially Massively Multiplayer Online Games (MMOGs) applications. Several MMOGs have experienced rapid and pervasive growth in recent years. These applications considered as multi-tier applications that offer to their users a huge virtual world in which they can interact with tens of thousands of other players and build virtual identities (Van Den Bossche et al., 2006). From the technical perspective and considering the intense interaction of the player with the host resources of MMOGs, one of their most important characteristics is the arrival rate of players, which cause workload fluctuations. The players naturally expect highly-available services with an acceptable quality of service (QoS), especially in terms of the response time to players (Marzolla et al., 2012). These vital facts raise a resource management issue for MMOG providers as a challenging problem. To guarantee the QoS requirements to a highly variable number of concurrent players, the game operators statically over-provision a large infrastructure capable of sustaining the game peak load, even though a large portion of the resources is unused most of the time. This results in the MMOG providers incurring high resource provisioning costs. As a result, efficient resource provisioning becomes an important challenge, especially for static and inflexible traditional IT infrastructures. Reducing the cost of resource provisioning and maintaining the satisfaction of MMOG players requires the MMOGs to be hosted in an infrastructure featuring elasticity compatible with the dynamic environment of the MMOG and the computational resources that must be managed.

One of the most suitable infrastructures for an MMOG application is a cloud computing infrastructure (Ghobaei-Arani et al., 2016; Aslanpour et al., 2017a). In recent years, the cloud computing paradigm has become an affordable model for autonomous resource provisioning problem (Prodan and Iosup, 2016). Cloud computing allows the users to rent resources from a cloud provider in an on-demand basis and pay for the resources they actually use (Khorsand et al., 2017a, 2017b). One of the key characteristics of cloud computing is its elasticity, which enables the

* Corresponding author.
** Corresponding author. Department of Computer Engineering, Qom Branch,Islamic Azad University, Qom, Iran
E-mail addresses: m.ghobaei@qom-iau.ac.ir (M. Ghobaei-Arani), Reihaneh_khm@yahoo.com (R. Khorsand), ramezanpour@mau.ac.ir (M. Ramezanpour).

https://doi.org/10.1016/j.jnca.2019.06.002
Received 29 September 2018; Received in revised form 13 April 2019; Accepted 1 June 2019
Available online 7 June 2019

MMOG providers to use cloud infrastructures for hosting their MMOG applications due to its elasticity feature. The cloud elasticity feature allows cloud users to acquire or release computing resources on demand, which enables MMOG application providers to auto-scale the resources provisioned to their MMOG applications under workload fluctuations to minimize resource cost while service level agreements (SLAs) is satisfied. These interesting properties of cloud computing attract MMOG providers to utilize cloud hosting so that they can migrate and the rented resources are managed dynamically through auto-scaling mechanisms (Wu et al., 2013).

To the best of our knowledge, none of the previous resource-provisioning approaches considered an autonomous resource-provisioning framework along with the adaptive prediction model and the fuzzy decision tree to estimate the proper number of resources to be allocated to each tier in the MMOG application in the cloud environment, i.e., all these techniques at a same time. Motivated by this, in this paper, we present an autonomous resource provisioning framework for MMOG applications in a cloud environment that follows a reference autonomic computing model proposed by IBM (Maurer et al., 2011; Ghobaei-Arani et al., 2018a, 2018b), which is named the MAPE-k (monitoring-analysis-planning-execution control loop with a shared knowledge-base) control loop. Our proposed framework uses the ANFIS prediction model and the fuzzy decision tree technique into the analysis and planning of MAPE-k control loop, respectively. In the monitoring phase, the current capacity of the system is continuously collected. Consequently, an analyzer component into the analysis phase predicts the future workload using an ANFIS prediction model to proactively control workload fluctuation. Proactive methods are used in the analysis phase to help prevent SLA violations. In the planning phase, a fuzzy decision tree algorithm is proposed to estimate the proper number of resources to be allocated to each tier in the MMOG applications. In the execution phase, an actuator component consists of load balancer and VM manager sub-components to adopt decisions using application-programming interfaces (APIs). The current research focused on the analysis and planning phases of the MAPE-k control loop. The most important mission of the proposed approach is to reduce the cost of resource rental for the MMOG provider when faced with QoS requirements. The main contributions of this research can be summarized as follows:

- We designed an autonomous resource provisioning framework based on an autonomic computing paradigm for MMOG applications in a cloud environment.
- We proposed an ANFIS prediction model to deal with workload fluctuations in order to gain higher prediction accuracy into the analysis phase.
- We utilized a fuzzy decision tree algorithm as a decision maker into the planning phase of the proposed framework to determine the appropriate auto-scaling decisions.
- We conducted a series of experiments under real-world and synthetic workload traces in terms of different metrics to evaluate the performance of workload predictor and the fuzzy decision tree algorithm.

The rest of this paper is organized as follows: In Section 2, we provide the necessary background. Section 3 focuses on a survey of related works. In Section 4, we describe the proposed framework in more detail. In Section 5, we present an evaluation and discuss the experimental results, and finally, in Section 6, we conclude the paper and present future works.

2. Background

The present study resolves a major challenge to MMOGs (QoS-aware resource provisioning through autonomic computing) in the cloud environment using an ANFIS technique and a fuzzy decision tree algorithm. Therefore, in this section, we provide a brief overview of the MMOG application, autonomic computing, prediction methods, and ANFIS theory.

2.1. Massively multiplayer online games

MMOGs are large-scale simulations of persistent game worlds comprising various objects or entities such as avatars (in-game representation of the players), bots or non-player characters (mobile entities that have the ability to act independently), movable objects (passive entities such as boxes or guns which can be manipulated but do not initiate interactions), and immutable entities or decor. The mostly employed architectural model for MMOGs is client/server (Van Den Bossche et al., 2006; Marzolla et al., 2012), with game operators maintaining the servers that simulate a distributed game world. The clients dynamically connect to a joint game session and interact with each other by sending play actions such as movements, shootings, operations on game objects, or chat. The vast majority of game servers follow a similar computational model implementing an infinite loop. In each loop iteration (also called tick), there are certain steps to be performed such as processing events coming from the connected clients and other servers, processing the states of the active entities, and broadcasting state update to the connected clients at a required rate. To accommodate the thousands of concurrent players into one single MMOG session, the current practice is to parallelize it and distribute the load across multiple resources using several parallelization techniques. Spatial scaling of a game session is achieved through a conventional parallelization technique called zoning, based on similar data locality concepts as in scientific parallel processing (see Fig. 1). Zoning is currently being applied by partitioning of the game world into zones to be handled independently by separate machines. Zones are typically predefined static geographical areas, where the transition among zones can only happen through certain portals such as special doors or teleportation. The players can be connected to servers at the time of execution with the accuracy of peer-to-peer quality. When a data connection or data center is dense, players migrate to other servers. The virtualization infrastructure includes geographically distributed hosts (data centers and providers). The hosts provide two services: 1) Load prediction service, which is responsible for communicating the future distribution of entities across the world on the workload. 2) Resource allocation service, which provides additional resources to local servers and the game as is appropriate for user needs and guarantees an instant game response.

2.2. Autonomic computing

Autonomic computing paradigm can be applied to implement a cloud application system, which knows its state and reacts to changes in it. The term autonomic computing paradigm was first used by IBM in 2001 to

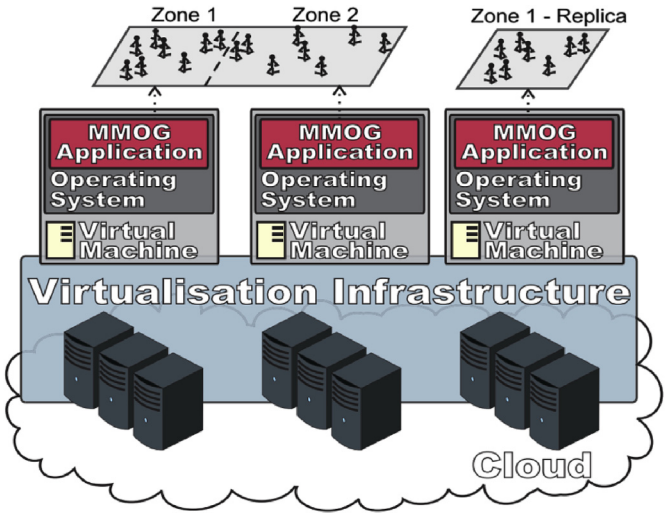


Fig. 1. The MMOG ecosystem model (Prodan and Iosup, 2016).

describe computing systems that are said to be self-managing (Huebscher and McCann, 2008). To achieve autonomic computing, IBM proposed a reference model for autonomic control loops (Jacob et al., 2004) called the MAPE control loop as shown in Fig. 2, which the managed element represents any software or hardware resource such as a cloud application, cloud service, operating system, CPU, storage or virtual machine (VM). Sensors collect information about the managed element. The information can be the response time to the requested cloud services, CPU, and memory utilization. Effectors carry out changes to the managed element. The change can involve adding or removing VMs to a cloud application (e.g., horizontal scaling) or changing configuration parameters in a VM (e.g., vertical scaling). The data collected by the sensors allow the autonomic manager to monitor the managed element and execute changes through effectors. The autonomic manager is a software component for monitoring the system, analysis of metrics, planning corrective actions and executing them. Knowledge in the MAPE control loops takes the form of shared data between the actual MAPE phases. Cloud application deployment is monitored, and the configuration is adjusted according to the metrics reported by sensors attached to the application or its environment. The monitored data is analyzed by the autonomic manager in the corresponding phase of the MAPE control loop. The raw sensor data is turned into knowledge in this phase by monitor component. Analysis of the system model may indicate that one or more of the criteria of acceptable system behavior are no longer being met (reactive trigger) or some metric is about to exceed its tolerated range (proactive trigger). Given such a situation, the autonomic manager will enter the planning phase to create a plan of action to bring the metric values back to or keep them in the tolerance zone. This plan is made by the planner component based on a set of rules that govern the operation of the autonomic manager. The execution phase includes an actuator component is where the autonomic manager or its delegate effector components interface with the application and the cloud environment to carry out actions decided upon in the planning phase. This phase relies on automation APIs available for the environment and the runtime configurability of the application. The executed actions will cause changes in the behavior of the system, which is then reported back to the autonomic manager in subsequent control loops.

2.3. Prediction methods

The prediction method and efficient resource provisioning is a fundamental research challenge in the cloud. Fig. 3 shows that the current research on application prediction can be classified according to the dimensions of the application (Amiri and Mohammad-Khanli, 2017). In general, prediction can be employed on physical machines or VMs. At the host level, the goal is to predict resource utilization in the host to find the appropriate number of physical machines. At the VM level, most methods predict the workload, performance and SLA parameters of cloud applications. The application workload is interpreted as the number of requests of the application, resource utilization of VMs and future demand by VMs. Performance prediction is equivalent to throughput and response time prediction using the resources allocated to it. In the SLA parameter prediction, the SLA violations of the application are predicted in accordance with its workload or the resources allocated to it. The focus of this paper is on the prediction of the number of requests for applications using ANFIS technique.

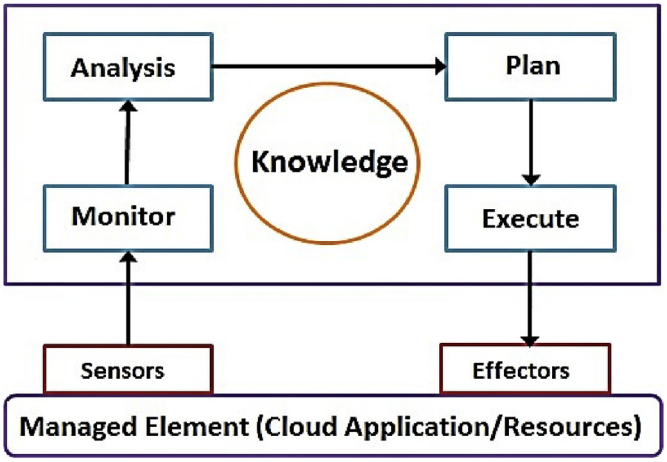


Fig. 2. The MAPE-k control loop.

2.3.1. ANFIS theory

ANFIS is a technique for automatically tuning first-order Sugeno type inference systems based on training data (Amiri and Mohammad-Khanli, 2017). It can be applied in different domains such as nonlinear function modeling (Jang, 1991a), time series prediction (Amiri and Mohammad-Khanli, 2017; Jang and Sun, 1993), parameter identification for control systems (Masugi and Takuma, 2007) and fuzzy controller designing (Jang, 1991b). Without loss of generality, we assume that the fuzzy inference system, under consideration, has two inputs x and y and one output f. We assume for now a first-order Sugeno fuzzy model (Takagi and Sugeno, 1985) of rule base with two fuzzy if-then rules can be expressed as:

Rule 1: If x is A_1 and y is B_1, then $f_1 = p_1 X + q_1 y + r_1$

Rule 2: If x is A_2 and y is B_2, then $f_2 = p_2 X + q_2 y + r_2$

Fig. 4 illustrates the reasoning mechanism for this Sugeno model (Jang, 1997). Where x and y are the inputs, f is the output, A_1, A_2, B_1, B_2 are the input member ship functions, w_1, w_2 are the rules firing strengths and { p_i, q_i, r_i} is the parameter set. The steps of fuzzy reasoning (inference operations upon fuzzy if-then rules) performed by fuzzy inference systems are:

A. On the premise part:
   - The input variables are compared with the member ship functions to obtain the member ship values of each linguistic label (this step is often called fuzzification)
   - The member ship functions are combined to get firing strength (weights) of each rule.

B. On the consequent part:
   - The qualified consequent is generated of each rule depending on the firing strength.
   - The qualified consequents are aggregated to produce a crisp output (this step is called defuzzification).

According to Fig. 5, the antecedent of rules contains fuzzy sets (as member ship function), and the consequent is a first order polynomial (a crisp function) (Jang, 1997). We used the Gaussian member ship function with product inference rule at the fuzzification level. Fuzzifier outputs the firing strengths for each rule. The vector of the firing strengths is normalized, and the obtained vector is defuzzified by utilizing the first order Sugeno model.

The corresponding equivalent ANFIS architecture is drafted in Fig. 5 (Amiri and Mohammad-Khanli, 2017). This consists of five layers. The description of each layer is given below:

Layer 1: Each node in this layer generates the member ship function of a linguistic label. The i^{th}node may perform the following operation. Where x is the input of i^{th} node, a_i, b_i, c_i are parameters to be learnt. These variables are used to adjust the shape of member ship function:

$$O_i^1 = \mu_A(x) = \frac{1}{1 + \left[\left(\frac{x - c_i}{a_i}\right)^2\right]^{b_i}} \quad (1)$$

Layer 2: The firing strength of each rule is calculated by:

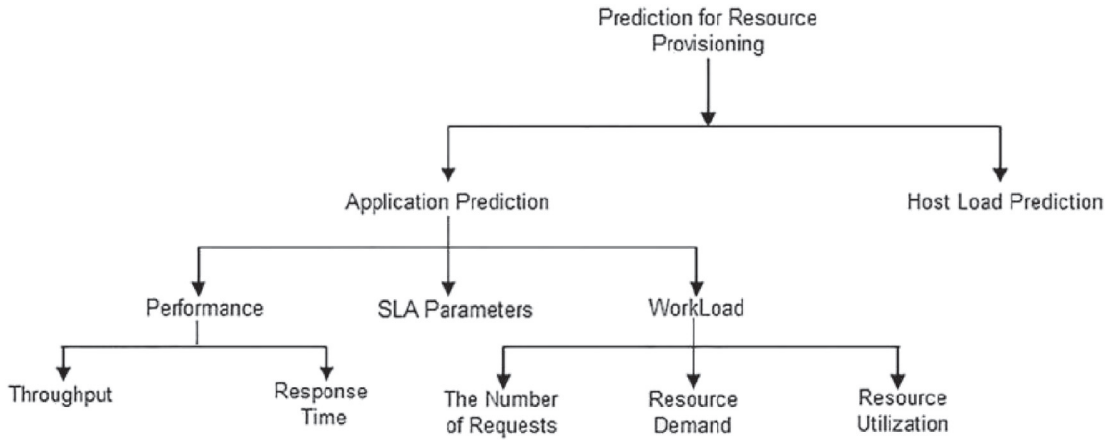


Fig. 3. The taxonomy of prediction levels for resource provisioning (Amiri and Mohammad-Khanli, 2017).

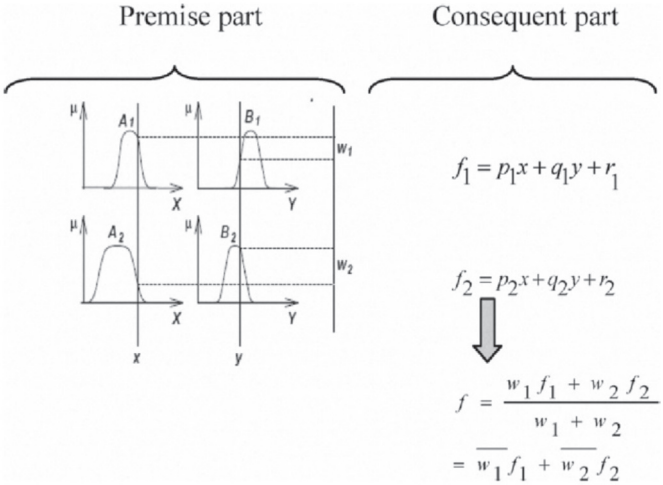


Fig. 4. Two-input first order Sugeno fuzzy model with two rules.

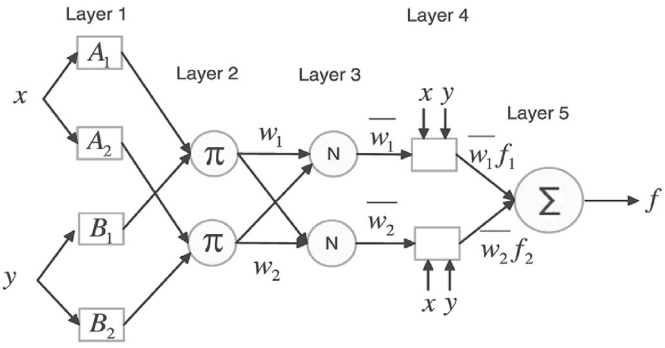


Fig. 5. ANFIS architecture.

$$O_i^2 = w_i = \mu_{A_i}(x) * \mu_{B_i}(y), \qquad i = 1, 2 \tag{2}$$

Layer 3: Layer 3 calculates the ratio of *i*th rule's firing strength to the sum of all rule's firing strength:

$$O_i^3 = \overline{w_i} = \frac{w_1}{w_1 + w_2}, \qquad i = 1, 2 \tag{3}$$

Layer 4: Every node in this layer has the following function:

$$O_i^4 = \overline{w_i} f_i = \overline{w_i}\,(p_i x + q_i y + r_i) \tag{4}$$

Layer 5: Layer 5 aggregates the overall output as the summation of all incoming signals:

$$O_i^5 = \sum_i \overline{w_i} f_i = \frac{\sum_i \overline{w_i} f_i}{\sum_i w_i} \tag{5}$$

Using this method, a fuzzy inference system is designed based on system specifications. This initial model is changed to a neuro-fuzzy network. Then it trained by experimentally measured data from the real system. Therefore, modeling and predicting, using ANFIS, start by obtaining a data set (input/output data points) which is divided into training and validating data sets (Chabaa et al., 2009). The current study used ANFIS to predict workloads (# of user requests).

3. Related works

Dynamic resource provisioning for an MMOG application is a relatively new concept. It has caught the attention of researchers because of the high popularity of this category of games in recent years. Several related works are reviewed in this section, and a comparative study of them is presented.

Marzolla et al. (2012) proposed an adaptive resource provisioning solution, which can reconfigure the resource pool to adapt to workload fluctuations of MMOG applications and satisfy the QoS requirements. Their solution utilized a queueing theory model to compute the response time for various configurations. Besides, they used a greedy heuristic to allocate the minimum number of servers to each tier of MMOG applications in order to meet the deadline constraint. Basiri et al. (Basiri and Rasoolzadegan, 2018) proposed a delay-aware resource provisioning approach for serving the cloud gaming systems. Their proposed approach used two types of VMs for performing game-related actions to increase resource utilization and decrease both the operational cost and the processing delay. Besides, they presented a round trip time delay model, which takes into account different delay including the transmission delay, the propagation delay, and the waiting delay. Khorsand et al.(2018) presented an efficient cloud-based resource provisioning approach of multitier applications. They proposed a hybrid resource provisioning approach for multitier applications based on a fuzzy analytical hierarchy process approach. Their proposed strategy allocates or deallocates cloud infrastructure based on the fluctuation in the number of users in multitier applications. Besides, they evaluated their strategy under different numbers of users and indicated that it outperforms in terms of the resource cost, and the response time compared with the other strategies. Lin et al. (Lin and Shen, 2017) developed a lightweight fog-assisted cloud gaming system named CloudFog for offloading the compute-intensive online game to fog nodes. They proposed a dynamic fog node

provisioning mechanism through predicting the number of players and then determined the appropriate number of fog nodes deal with user churns. Further, they introduced a reputation based fog node selection mechanism for assigning each player to a suitable fog node for guaranteeing QoS requirement. The simulation results demonstrated that their proposed solution reduces the latency and bandwidth usage and increases user coverage. Nae et al. (2011) studied the various types of player interactions, a source of short-term load fluctuations, which complements the long-term load fluctuations due to the size of the player's population. They proposed a dynamic resource-provisioning algorithm in which the amount of resources is predicted, and then virtual machines leased from external data centers for the MMOG applications. Further, they described an analytical model for the load of MMOGs including CPU, memory, and network that considers both the player interaction type and the population size. Their model is utilized to dynamically estimate the MMOG resource demand based on the game world entity distribution. Dhib et al. (2017) studied the resources allocation problem of MMOG services under the density of zones and workload fluctuations over a distributed cloud infrastructure. Their solution utilized a multiple multi-dimensional knapsack model to formulate the cost-aware virtual machines placement problem with aim trade-off between the response time and total cost. Meiländer et al. (Meiländer and Gorlatch, 2018) developed a new scalability model for MMOGs on Clouds that monitors the application performance during the runtime to predict load-balancing decisions. Their proposed model is consist of two sub-models, namely: a computation resources model to add/remove resources according to workload changes and a network resources model to identify the required bandwidth usage for load balancing. Their experimental results on the multi-player shooter online game as an MMOG application indicated that their scalability model is adequate and accurate. Jia et al. (Jia and Liang, 2018) described a new system model for enabling a massively multiplayer game where players interact with each other over deployed cloudlets. They formulated the multiplayer coordination problem to minimize the game frame duration among players. In their algorithm, since each player has a number of nearby cloudlets it can connect to, the proposed algorithm assigns each user to the cloudlet with which it has the strongest wireless connection. Further, they evaluated their algorithm with varying the number of players, the number of regions and the number of cloudlets. They indicated the proposed algorithm outperforms in terms of the game frame duration compared with the other algorithm. Farlow et al. (Farlow and Trahan, 2018) proposed a new periodic approach for load balancing in online games along with a set of heuristics. Their proposed approach uses three actions to reduce the load on the highest loaded host and increase the load on the lightest loaded host: the zone adding/shedding, the proactive zone splitting, and the reactive zone splitting. Gao et al. (2018) studied the energy-aware dynamic resource-provisioning problem for cloud-based MMOG services from the perspective of MMOG providers. They formulated the dynamic resource provisioning problem using Genetic algorithm by considering various types of resources, including CPU, memory, network bandwidth, and power consumption as an optimization problem. Their proposed solution evaluated under the workload traces of the Stendhal and the BZFlag MMOG applications and showed that their solution achieves significant energy savings while satisfying desired QoS requirements for players. The summary of related works in autonomous resource provisioning of massively multiplayer online games is presented in Table 1.

4. Proposed approach

The proposed approach is described in detail in this section. A resource-provisioning framework based on the MAPE-k control loop is introduced, and a problem formulation for resource provisioning is presented. Then, an autonomic resource provisioning algorithm for MMOG application is presented.

4.1. Proposed framework

The resource-provisioning framework is inspired by the cloud layer model and provisions the resource for MMOG applications based on the MAPE-k control loop. Fig. 6 provides an overview of resource provisioning based on autonomic computing for MMOG applications in a cloud environment that consists of players, an MMOG provider and a cloud provider. The dashed rectangle in Fig. 6 shows the focus of this paper. The MMOG provider accommodates a MAPE-k control loop-based resource provisioning mechanism. Two types of SLA exist; user SLA and resource SLA. The user SLA is a contract between the MMOG provider and the players and includes items such as a deadline, budget, penalty rate, and request length. The resource SLA is a contract between the MMOG provider and the cloud provider and includes items such as VM cost, VM type, processing speed, and data transfer speed. The current study focuses on the interaction between the players and the MMOG provider when calculating SLA violations. The framework entities are as follows:

1. **Players** control the virtual world using an appropriate graphical user interface and send their MMOG requests to the MMOG provider. Human players control avatars, which interact with each other.
2. The **MMOG provider** is responsible for the maintenance of gaming conditions and the execution of all necessary interactions between the players and the environment in the virtual world (Marzolla et al., 2012). The MMOG provider stores information about a unique virtual

Table 1
Summary of related works in autonomous resource provisioning of massively multiplayer online games.

Ref	Utilized Technique	Performance Metric	Policy	Workload Type
Marzolla (Marzolla et al., 2012)	A queueing theory model	Response time, Cost	Reactive	Synthetic, RuneScape
Basiri (Basiri and Rasoolzadegan, 2018)	A round trip time delay model	Delay, Cost	Proactive	Synthetic
Lin (Lin and Shen, 2017)	A fog node selection mechanism	Response latency, Cost, Bandwidth usage, User coverage ratio	Proactive	PlanetLab workload
Nae (Nae et al., 2011)	Neural Network	Response time, Cost	Proactive	RuneScape
Dhib (Dhib et al., 2017)	A multiple multi-dimensional knapsack model	Memory allocation cost, Delay	Reactive	World of Warcraft
Meiländer (Meiländer and Gorlatch, 2018)	A Model-based approach	Bandwidth usage, CPU usage	Proactive	Steam traces
Jia (Jia and Liang, 2018)	A heuristic-based iterative algorithm	Running time, Game frame duration	Reactive	Synthetic
Farlow (Farlow and Trahan, 2018)	A new periodic heuristic-based approach for load balancing	Average system load, Number of balanced breakpoints	Reactive/ Proactive	Synthetic
Gao (Gao et al., 2018)	ARMA, Neural network predictor and Genetic algorithm	Interaction delay, energy consumption	Proactive	Stendhal, BZFlag traces
Proposed approach	ANFIS predictor and Fuzzy decision tree planner	Mean Square Error, Coefficient of determination, Response Time, Number of VMs, Cost	Proactive	Synthetic, RuneScape

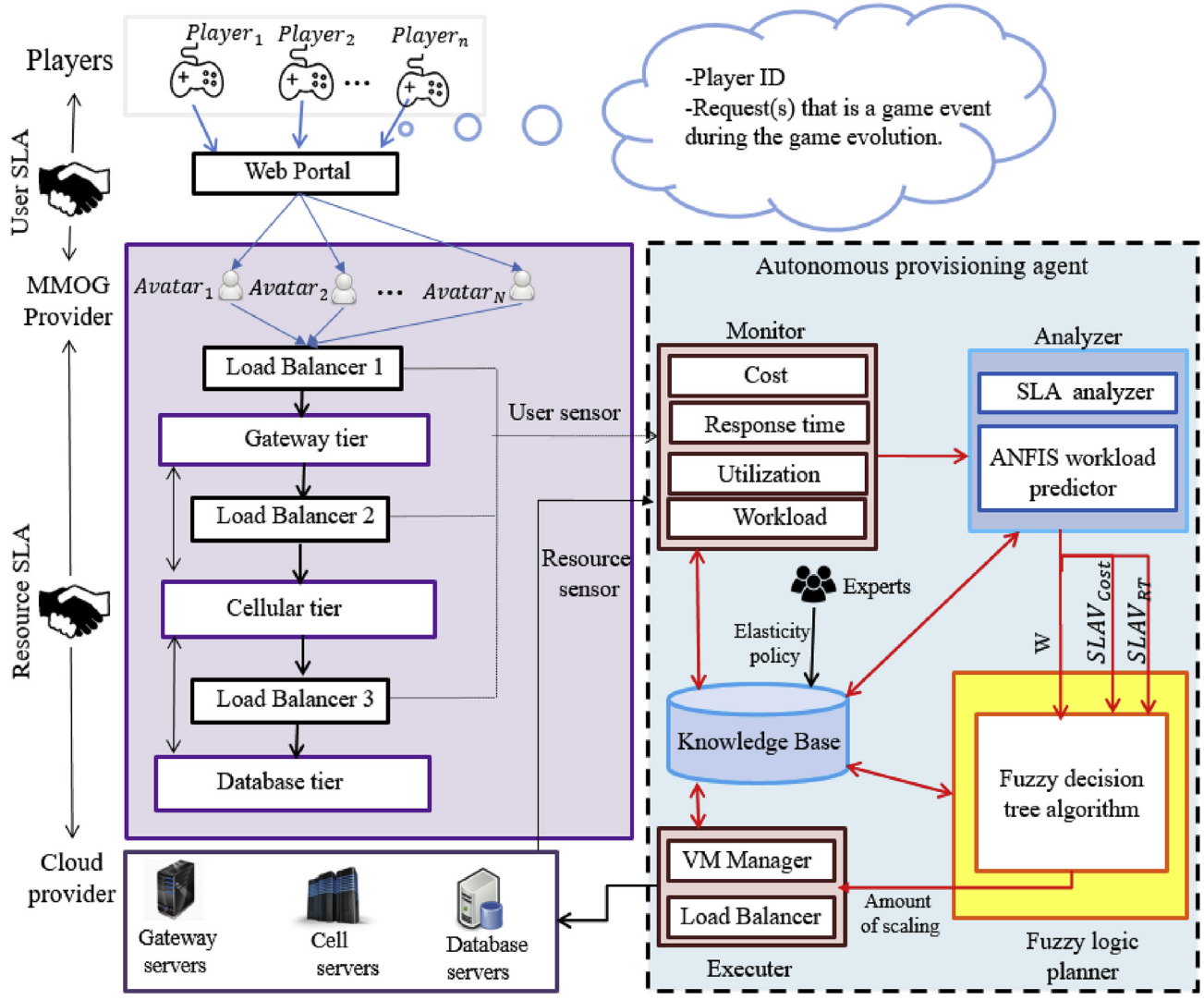


Fig. 6. Autonomous resource provisioning framework.

world. Specifically, each player has a virtual world created by the MMOG provider upon entering the cloud environment. The MMOG provider uses an agent for dynamic resource provisioning which takes actions to remove and add VMs in the different tiers of the MMOG application when faced with growth or to reduce the incoming load. The purpose of this agent is to increase utilization and reduce costs using a MAPE-based control loop. The autonomous provisioning agent either uses internal data center resources that provide the MMOG with dynamic resources or rents resources from a cloud provider (e.g., Amazon EC2) to host the MMOG application.

3. The **cloud provider** is responsible for providing computing resources to the MMOG provider based on the pay-per-use charging model. The cloud providers provision unlimited resources in the form of VMs with different configurations (small, medium and large) to serve the requests of different tiers of the MMOG application to the MMOG provider.

Because this work focuses on improvement of the performance of the MMOG provider (the middle layer of the proposed framework), the MMOG application architecture and autonomic provisioning agent in the proposed framework requires explanation.

4.1.1. MMOG architecture

The MMOG application architecture is composed of the gateway, cellular and database tiers as follow:

- **Gateway tier:** The gateway layer functions as a bridge between a client application and the game layer. It handles all the connections to the clients and makes sure all the packets get forwarded to the appropriate servers in the game layer. Furthermore, it can deal with security issues in the form of a firewall and make sure only paying customers can actually play the game.
- **Cellular tier:** This layer forms the heart of the game, as this is where the virtual world is housed, maintained and executed. All requests and commands sent by the players through their clients get forwarded to this layer. The game servers process this command, verify if they are legal, compute the changes to the virtual world, update the game state and forward the changes to all the clients that they affect. The main task of this layer is to maintain the state of all players and other objects in the world. It manages the position in the world of all players and controls monsters, non-player characters, treasures, and all other dynamic aspects.
- **Database tier:** Every move a player makes, every item, object, and the monster is a data element in the game's database. However, even the largest game worlds are minuscule compared to data warehouses employed by thousands of companies around the world. Furthermore, the schemas and structures of the game databases are also simple. Because of this, they can take advantage of Cloud-based database

technology and other large memory systems to speed up the processing of transactions.

➢ **Load balancers:** Load balancers are used to evenly distribute requests among servers across layers.

A component of the proposed resource-provisioning framework based on autonomic computing is the negotiation between players and MMOG game providers (Ghobaei-Arani et al., 2016; Aslanpour et al., 2017a). The interaction between the player and the game provider requires human intervention only from the player. The relationship is regulated by the player account, created through a Web portal upon agreeing on a contract with the MMOG game provider. The contract includes generic mutual obligations valid for all MMOGs, while further refinements and extensions can be added for particular MMOGs in the form of annexes. Typical player obligations include subscription costs, player interaction rules, and costs for accessing MMOG sessions. Typical game provider obligations include guaranteed services, such as community support, player support (player status and achievements, inventory, and detailed play statistics), mediation of player connections to MMOGs, access and availability to game world areas, and compensation in case of contract violations. A design of SLA-based negotiation protocol between the MMOG game providers, and the players fall outside the scope of this work, but solutions already exist (Prodan and Iosup, 2016).

A request is an active player that periodically generate a game event during the game evolution. A population of N requests continuously circulate in the system, N being the total number of players currently connected to the system. We allow the value of N to change over time, as users join and leave the system.

The proposed approach assumes that the game is divided into a number of zones and a data center is only responsible for one zone in the virtual world. Because the number of incoming requests to a tier could be more than the computing power of the VMs provisioned for that tier, a queue of requests in the form of a load balancer accrues before entering each tier. Controlling resources in the three tiers and fluctuating player behavior are management problems that make the MMOG provider use an autonomous provisioning agent for dynamic and targeted management of rented cloud resources. When the workload increases, the MMOG provider can request additional resources from the cloud provider, so that the response time stays below the limit. When the workload is reduced, the MMOG provider can release additional resources to reduce costs. The following provides more details about the autonomous provisioning agent based on the MAPE-k control loop.

4.1.2. Autonomous provisioning agent

Fig. 5 shows that the proposed autonomic resource provisioning mechanism complies with the MAPE-k control loop concept. It includes the phases of monitoring, analysis, planning, and execution.

Monitor: The monitor component is responsible for collecting the metrics of the request cost, response time and workload. The user and resource sensors collect sufficient information to determine these metrics from the virtual game and cloud environment and are available to the monitor component. This information is monitored, integrated and stored in the knowledge base to be used by other phases. The resource sensor is responsible for gathering information about the computing resources, storage and network (such as CPU utilization, memory usage, and network traffic) from the cloud provider. The user sensor is responsible for gathering information about the workload of players (request rate, request type, request size, etc.) from the queues of requests assigned to load balancers 1, 2 and 3.

Analyzer: The analyzer component is responsible for processing information gathered from the monitor component. Data obtained from the monitoring sensors are examined by the analyzer component in order to employ the required adaptive actions for ensuring the QoS. The workload analyzer component uses prediction techniques to predict the number of requests and deal with workload fluctuations. The SLA analyzer also determines the SLA violation rate according to response time.

Planner: The planner component determines when and how many VMs must be allocated to the MMOG tiers in order to find a trade-off between cost and SLA violation. Based on the predictions of the analyzer component and the SLA violation rate, the planning component uses a fuzzy decision tree algorithm to decide how many new VMs should be added or existing ones removed. Moreover, the planner component estimates the response time after the change.

Executor: The executor component consists of a load balancer and VM manager sub-components. Because the load balancer component receives all incoming requests from players and distributes them to appropriate VMs according to load balancing policies (such as round robin or random), the VM manager component is directly connected to the VM of the cloud provider. This component is responsible for real execution of actions (scale out/scale in) of the decision made by the planner component.

Knowledge Base: In an autonomic system, a component Knowledge Base (KB) maintains data of the managed system and environment, adaptation goals, and relevant states that are shared by MAPE components. The MAPE components can communicate explicitly or indirectly by sharing information in the knowledge repository. In this paper, it is assumed that the knowledge base has four parts: 1) The knowledge about the components and their construction and configuration 2) The knowledge about the elasticity policies 3) The knowledge created by the monitor part 4) The knowledge updated by the execution part.

4.2. Problem formulation

This section provides the notations used in the proposed approach as shown in Table 2. Let N represent the number of active players ($Player = \{Player_1, Player_2, Player_3, \ldots, Player_N\}$) that periodically generate game events during the game evolution. Since the number of requests of each player is unique, $Count\text{-}req_n$ denotes the number of requests by the nth player. The total number of requests is $M = \sum_{i=1}^{N} Count\_req_i$.

The MMOG provider owns an i-tier application, each tier providing specific functionality to the preceding tier in the form of a processing pipeline. C_iDenotees the ith tier of MMOG application. There are.

$VM\_Allocate\_Tire_i$ Identical VMs in the ith tier, but the VMs of one tier may be different from those of another tier. The Mean Value Analysis (MVA) algorithm (Pattipati et al., 1990) can be used to compute individual VM utilizations with n requests given the average queue lengths

Table 2
Notations and definitions.

N	The number of concurrent requests (the number of active players)
$Count\_req_n$	Number of requests by the nth player
M	Total number of requests for all players ($M = \sum_{i=1}^{N} Count\_req_i$)
D	Number of points of a Δ spaced time series
A_i and B_i	Two variable in workload prediction with linguistic values (high, middle, low)
R	Measured system response time
C_i	the ith tier of MMOG application
$VM\_Allocate\_Tier_i$	Number of virtual machines provisioned to the ith tier
U_i	Estimated average utilization of nodes at tier i
$SLAV_{Cost}$	The value of the SLA violation in terms of cost
$SLAV_{RT}$	The value of the SLA violation in terms of response time
$SLAV_{RT_i}$	The value of the SLA violation in terms of response time for the ith request
x_t	The number of requests in time t
$VM\ Price_i$	The cost of the VM for the ith request
$VM\_Init\_Cost_i$	The cost of the initial configuration of the VM for the ith request
VM_Time_i	The length of time each VM has been rented from the cloud provider to service
Penalty_Cost	The penalty the provider pays to the player for an SLA violation
SV	scaling value

with n-1 requests. The monitor collects the tier utilizations U= (U_1, U_2, U_3), where U_i is the utilization of tier i. Analyzer computes $SLAV_{Cost}$ and $SLAV_{RT}$ to represent SLA violation variables. The sum of the cost of allocating a VM and the cost of penalties for SLA violation are computed using Eq. (17-20). The SV denotes the amount for scaling decision-making (*Scale.Value*).

4.3. Proposed algorithm

The proposed resource provisioning approach is based on autonomic computing for the MMOG application. In the proposed algorithm, the requests are put into a queue and sent to the appropriate tier of the MMOG application according to the processing requirements. At first, requests enter in the request queue as a pre-processing phase. Because users have various requests, each user request is placed in the request queue. This process is repeated for all users. Then, the need for each request for a certain tier or tiers is extracted. The user request is created according to SLA specifications. Each SLA is composed of a number of service level objectives (SLO). Each request includes a user ID and the amount of cost/time considered based on the user's group for that request. The users also can simultaneously send multiple requests to the cloud provider. If the cost/time of serving the request exceeds the agreed cost, the SLA is violated.

Because the proposed approach focuses on dynamic resource provisioning using the MAPE-k control loop, the sub-algorithms are defined for the monitoring, analysis, planning and execution phases as shown in algorithm 1. First, the monitoring phase extracts the required information for the analysis phase. This information includes cost, response time, the tier utilizations and incoming workload to the each tier (line 4). This information is provided to the analysis phase to allow prediction of the future workload and the calculation of the SLA violation rate. In other words, the value of the *Predict* and SLA violation (*SLAV)* variables are set (line 5). The predicted workload and the calculated the SLA violation are available for the planning phase in which decisions about the amount of scaling are made; this is the new number of VMs for the scale-in/scale-out (line 6). The execution phase applies the required changes based on the decisions made in the planning phase on the cloud infrastructure by adding or releasing VMs (line 7).

Algorithm 1: Pseudo code for Autonomic Resource Provisioning

Input: User requests, SLA specifications
Output: New system configuration (apply required changes based on scaling value)
1: Initialization: boots an appropriate number of VMs for each tier Ci
2: while (the system is running and at the beginning of interval Δt) do
3: for i ←1 to 3//count of application tiers is 3
4: REC←Monitoring (Incoming workload, response time, cost, tier utilizations)
5: [Predict, SLAV] ←Analysis ([REC])
6: [scale_value] ← Planning (Predict,SLAV)
7: Execution ([scale_value])
8: end for
9: end while

4.3.1. Monitoring phase

The monitor is a passive observer, which is responsible for receiving the workload and using it to calculate the response time and the cost of each request. The monitor component has access to the knowledge base as a shared data between the MAPE-k control loop phases. It checks the system status at time intervals, collects its required information, and stores them in the knowledge base. In this paper, it is assumed that the knowledge base has two parts. The first part keeps the workload and the second part keeps the response time, request cost, throughput, and a number of VMs metrics. The knowledge base records for both parts are shown in Fig. 7.

4.3.2. Analysis phase

The analysis phase processes knowledge-base records gathered directly from the monitoring phase. The data obtained in the monitoring phase (average response time, average cost and request arrival rate of each tier) are examined in this phase to determine whether or not adaptive actions are required to guarantee the requested QoS. In this phase, the first item examined is SLA violations by the SLA analyzer component and the second item is the prediction of the future workload by the workload analyzer component. In the proposed approach, the workload analyzer uses the ANFIS technique to predict future workload based on the number of requests collected by each tier. In addition, if the cost and response time to serve a request is higher than the considered cost and response time, the SLA violation will be calculated in terms of cost and response time.

4.3.2.1. Examination of SLA violation. The SLA analyzer uses Eqs. (7) and (9) to calculate the amount of SLA violation in terms of cost and response time, respectively.

$$SLAV_{Cost} = \sum_{i=1}^{3} C_{C_i} - Cost_{Req} \tag{6}$$

$$SLAV\_Count_{Cost} = Count(SLA\_Violation_{Cost} > 0) \tag{7}$$

$$SLAV_{RT} = \sum_{i=1}^{3} R\_C_i - Time\_Limit_{Req} \tag{8}$$

$$SLAV\_Count_{RT} = Count\left(SLA\_Violation_{Response} > 0\right) \tag{9}$$

where $Cost_{Req}$ is the initial cost considered for the request and $\sum_{i=1}^{3} C_{C_i}$ is the final cost of serving the request in Eq. (6). $Time\_Limit_{Req}$ is the maximum time (deadline) that the player would prefer to wait for the result and $\sum_{i=1}^{3} R\_C_i$ is the amount of time taken to process the a player request from tier 1 to tier 3 in Eq. (8).

4.3.2.2. Workload prediction using ANFIS. From literature review, there are two drawbacks in prediction models: (1) conventional statistical methods, such as regression models are unable to deal with the nonlinear relationships well; and (2) the rules generated from conventional statistical methods (i.e., ARIMA), and artificial intelligence technologies (i.e., support vector regression (SVR) and artificial neural networks (ANN) are not easily understandable for researchers. In order to deal with these drawbacks above, this paper utilizes a fuzzy inference system (employing fuzzy if-then rules) to model the qualitative aspects of human knowledge, then uses the adaptive network to optimize the fuzzy inference system parameters, which can deal with the nonlinear relationships (Cheng and Wei, 2010). The proposed ANFIS workload predictor utilizes Sugeno-type Fuzzy Inference System (FIS) controller, with the parameters inside the FIS decided by the neural-network back propagation method. The ANFIS workload predictor is designed by request numbers as the inputs and the future request number as the output. The output is calculated using the fuzzy member ship functions depending on the input variables. The effectiveness of the proposed solution to the modeling and simulation of the workload predictor is implemented in the Simulink environment of MATLAB. The ANFIS-Editor in MATLAB is used for realizing the system and implementation of the proposed ANFIS workload predictor. In a conventional fuzzy approach, the model designer according to a priori knowledge fixes the member ship functions and the consequent models. If this set is not available, but a set of input-output data is observed from the process, the components of a fuzzy system (i.e., member ship and consequent models) can be represented in a parametric form, and neural networks tune the parameters. In that case, the fuzzy system turns into an ANFIS system. For a better explanation, the overall flowchart of the proposed ANFIS workload prediction model is shown in Fig. 8. To easy understand the proposed ANFIS workload

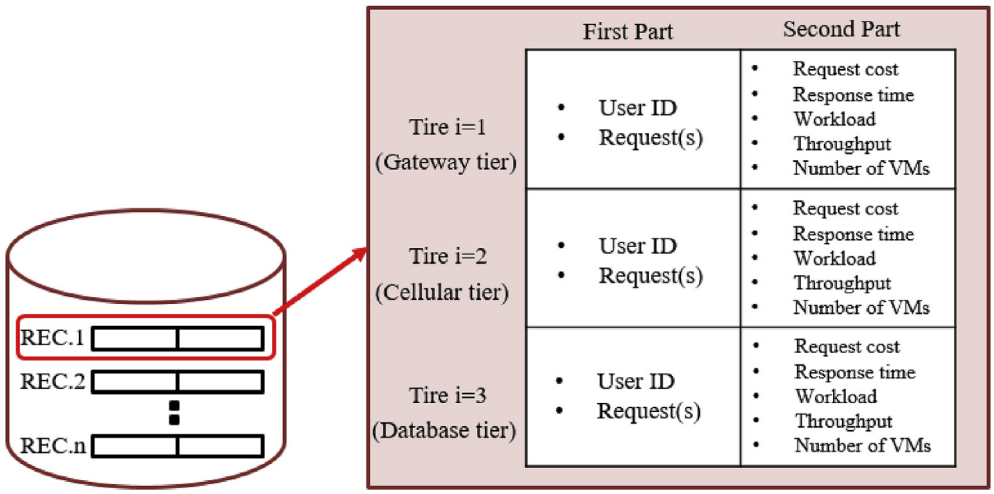


Fig. 7. The structure of knowledge base records.

prediction model, it is introduced step by step in the following:

(1) **Collect dataset:** In this step, the MMOG workloads are collected as research data. The aim is to use the values of the time series (i.e., the workload of MMOG) known at a point x = t to predict the value of the series at some future point x = t + P. Actually, the training data have been generated by sampling workload of MMOG.

(2) **Build "one-step-ahead" forecast model:** In this step, the standard method for the prediction is to create a mapping from D points of a Δ spaced time series, is (x (t - (D - 1) Δ),…, x(t - Δ), x(t)), to a predicted future value x(t + P). In the proposed workload prediction algorithm, the requested number of each tier is collected to predict the requested number of services at the next time interval, and the values D = 11 and Δ = P = 1 are used.

We predict x (t) from the ten past values of the time series, that is, x (t −10), x (t −9), …, and x (t −1). Therefore the format of the training data is:

$$[x\,(t-10), x(t-9), \ldots, x(t-1); x(t)] \tag{10}$$

Each input set is divided into two sets. The first input set is used for training the network, and the other input set is used to test the ANFIS network (Amiri and Mohammad-Khanli, 2017).

(3) **Generate ANFIS forecast model:** The ANFIS forecast model will be introduced in Step 3.1 to Step 3.4, and the ANFIS method uses

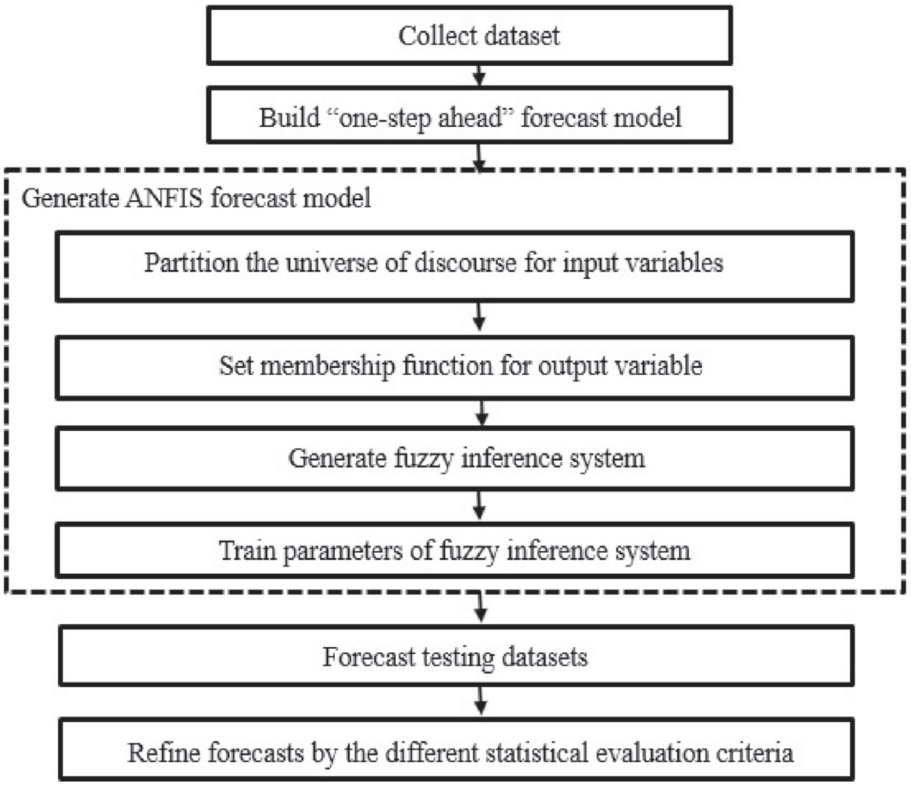


Fig. 8. Flowchart of the proposed ANFIS prediction model.

subtractive clustering to partition the universe of discourse for input variables and then generates the fuzzy inference system. The sub-steps of Step 3 are described as follows:

Step 3.1: Partition the universe of discourse for input variables. Firstly, we define each universe of discourse for variables according to the minimum and maximum value in each variable. Secondly, partition the universe of discourse by subtractive clustering (i.e., Gaussian member ship function) (Chiu, 1994).

Step 3.2: Set member ship function for output variable. In this step, we set Gaussian-type member ship function for output variables. For example, there are two-period inputs (Xt−1,Xt−2) and three linguistic intervals (i = 1, 2, 3) partitioned by subtractive clustering in each input variable. Therefore, a typical rule in a Sugeno fuzzy model is described as follows:

If

$$x(x_{t-1}) = A_i \quad \text{and} \quad y(x_{t-2}) = B_i \quad \text{then} \quad f_i(x_t) = p_i X + q_i y + r_i \tag{11}$$

where x (x_{t-1}), y (x_{t-2}), are linguistic variables, A_i and B_i, are the linguistic values (high, middle, low), f_i (x_t) denotes the i-th output value, p_i, q_i, r_i are the parameters (i = 1, 2, 3).

Step 3.3: Generate a fuzzy inference system. Firstly, from Step 3.1, we can get the linguistic intervals as input member ship functions, and the output member ship functions are set by Step 3.2. Secondly, generate fuzzy if-then rules, where the linguistic values (A_i, B_i) from input member ship functions are used as the if-condition part, and the output member ship functions (f_i (x_t)) is the then part.

Step 3.4: Train parameters of fuzzy inference system. In this step, we employ the least-squares method and the backpropagation gradient descent method for training the forecasting model. This study sets epoch as 50 (the process is executed for the predetermined fixed number (50) of iterations unless it terminates while the training error converges) for the training stopping criterion, and then obtains the parameters for the selected output member ship function.

Step 4: Forecast testing datasets. The FIS parameters of the forecasting models are determined when the stopping criterion is reached from Step 3; then the training forecasting model is used to forecast x_t, x_{t-1} …, for the target training and testing datasets, respectively.

Step 5: Refine forecasts by the different statistical evaluation criteria: From Step 4, the forecast testing datasets are obtained. Then we use the different statistical evaluation criteria to adapt the forecasts with minimal error in the training dataset.

4.4.3. Planning phase

In the planning phase, the workload prediction, average response time and cost of each tier received from the analysis phase is used to plan for efficient resource provisioning. Scaling decision-making (scaling type and amount) is described below, and the fuzzy decision tree algorithm is employed to use the predicted results to determine optimal actions.

Fuzzy decision tree (Zhai et al., 2018) is an extension of the decision tree that can be applied to many fields, such as decision-making. It is an appropriate tool to deal with uncertain, imprecise or qualitative decision-making problems. Fuzzy classification rules can be extracted by a fuzzy decision tree from fuzzy decision tables (or constructed by capturing stakeholder experience) with fuzzy conditional attributes and fuzzy decision attributes. In this paper, capturing stakeholder experience is used to constructing fuzzy rules in the planning phase of each tier of the MMOG application. In fuzzy logic, the member ship of x in fuzzy set F has a fuzzy value in a continuous interval between 0 and 1 which represents the extent to which x belongs to F. Each fuzzy set is associated with a linguistic term such as “low” or “high”. The member ship function denoted by $\mu y(x)$ quantifies the degree of member ship of input signal x to the fuzzy set y.

The workload type, response time violation and cost violation were used as input variables in the present study. Workload type is the linguistic variable representing the level of workload as being low, medium or high. Value of the response-time violation ($SLAV_{RT}$) is a linguistic variable representing the violation as being good, bad or very bad. The linguistic variable represents the level of cost violation ($SLAV_{Cost}$) as being good, bad or very bad. The basic idea is to determine future resource needs using a fuzzy algorithm having the inputs of *Workload Type*, $SLAV_{RT}$ and $SLAV_{Cost}$ that are calculated in the analyze phase. The algorithm output is scaling type (*Scale in*, *Scale out*, $No-op$), where the output value determines the resource needs (the number of deployed VMs) at future times as an integer {−2, −1, 0, +1, +2}. The member ship functions are trapezoidal, as depicted in Fig. 9. Three fuzzy sets are defined for each input to achieve reasonable granularity in the input space while keeping the number of states small.

Fuzzy logic is used to design a fuzzy rule-based system that uses IF-THEN fuzzy rules to represent the knowledge or control strategies of system. A collection of fuzzy rules is called a rule base. There are many approaches to constructing fuzzy rules, such as capturing stakeholder experience or system control actions. In this paper, capturing stakeholder experience is used to constructing fuzzy rules. The fuzzy inference system processes the fuzzy rules to forecast future resource demands based on the predicted workload type, current $SLAV_{RT}$ and current $SLAV_{Cost}$. The stakeholders were asked to determine a scaling value. The stakeholders are different human experts (e.g., architects or administrators). For instance, administrators employ subconsciously a set of if-then rules to manage the amount of resources a system needs to maintain acceptable level of user experience. The following question is used to extract knowledge and stakeholder responses: IF (the workload is … & $SLAV_{RT}$ is … & $SLAV_{Cost}$ is …), THEN (scaling value (SV) is …).

In order to reduce the threat of ordering effects, we reordered the questions. To design the fuzzy rules, we collected the required data by performing a data collection among 10 experts in cloud computing. These experts were asked to determine a consequent using an integer from [−2, 2]. As we expected, different experts chose a different number of node instances for the same questions. We then calculated the mean of the three linguistic labels for workload, SLAV_ Response Time, and SLAV_-Cost. We provide an example to illustrate the generation of the fuzzy decision tree in Fig. 10. Moreover, the questions and responses are summarized in Table 3.

Algorithm 2 describes the pseudo-code of the planning phase. In this phase, two operations are carried out for each tier (line 1). In the first, the scaling value is calculated using fuzzy decision tree (lines 1–9). The planner carries out the following actions: (1) the inputs comprising the predicted workload, the value of response time violation and the cost violation value are fuzzified. In other words, the crisp data projects fuzzy information using member ship functions (line 4). (2) The fuzzy engine uses the set of fuzzy rules to extract fuzzy actions (line 5–8); (3) decisions made by fuzzy inference are in fuzzy form, which cannot be directly used; thus, the defuzzifier changes the results back to the crisp mode and activates an adaptation action (line 9).

The state space in the proposed system is finite (27 states, as the full combination of $3 \times 3 \times 3$ member ship functions) and the planner calculates the amount of scaling as a weighted average:

$$y(x) = \sum_{i=1}^{N} \mu_i(x) \times a_i \tag{12}$$

Where N is the number of rules, $\mu_i(x)$ is the firing degree of rule i for input signal x and a_i is the consequent function for the same rule. Because of the discrete nature of scaling actions, the output is rounded to the

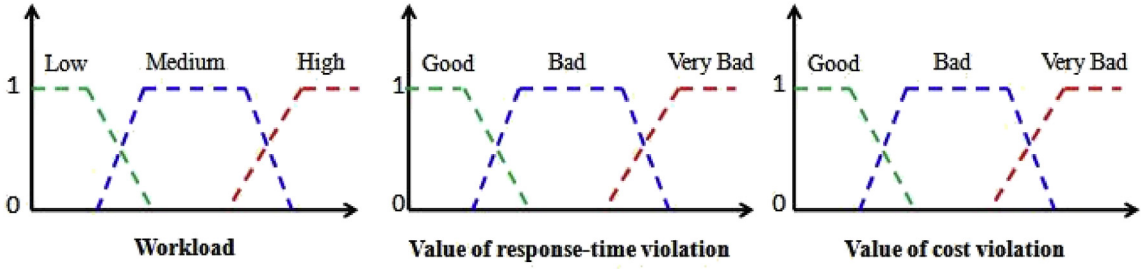


Fig. 9. Fuzzy member ship functions for scaling decision-maker algorithm inputs.

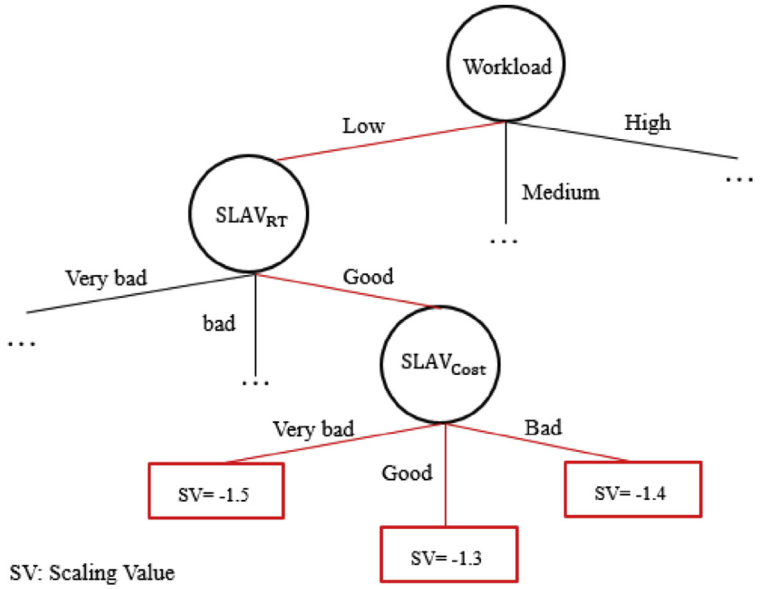


Fig. 10. An example to illustrate the generation of a fuzzy decision tree.

nearest integer (line 10).

Algorithm 2: Pseudo code for Planning Phase

Input: Amounts of the predicted workload, $SLAV_{Cost}$, $SLAV_{RT}$
Output: Scaling type and Scaling value
//Initialization
1: Define the Workload Type, $SLAV_{RT}$, and $SLAV_{Cost}$as linguistic variables and terms
2: Construct the member ship functions into three levels
3: Set of rules to determine output based on input values using capturing stakeholder experience
//Fuzzification
4: Convert crisp input data (the predicted workload, the value of response-time violation and the value of cost violation) to fuzzy values using the member ship functions
//Inference (Evaluate the rules in the rule base and calculate the amount of the scaling as a weighted average)>Eq (12)
5: for i← 1 to N
6: Summation = Sum μ_i (x)
7: y(x) = Summation* a_i
8:end for
9: Defuzzification (convert the output data to non-fuzzy values)
10: Scaling value = round the y(x) to the nearest integer
11: return (Scaling value)

4.4.4. Execution phase

In the execution phase, the decisions and changes that were adopted are applied to the infrastructure cloud according to the type and amount of scaling in order to adjust the resource configuration. For each tier, the command for the type and amount of scaling is sent to the cloud provider, which executes the command using theVM manager component in order to realize efficient resource provisioning by adding or releasing VMs. Finally, the load balancer dispatches requests to suitable VMs according to the best-fit strategy by sorting requests based on deadline in ascending order and VMs based on CPU utilization (U_i) in descending order.

5. Performance evaluation

In this section, the results of the simulation of the proposed approach are evaluated. The aim of the experiments was to show the efficiency of the proposed autonomic resource provisioning approach for MMOG applications in cloud environments. In the following, experimental setup and performance metrics are explained, and the experimental results are discussed.

5.1. Experimental setup

Existing simulators like GridSim (Buyya and Murshed, 2002), and CloudSim (Calheiros et al., 2011) can only be used to simulate batch processing and infrastructure utilization workloads and are not suitable for interactive 3-tier applications. There is a need to bring the benefits of modeling and simulation to the area of 3-tier application provisioning in Cloud. Rice University Bidding System (RUBiS) consists of an e-commerce website similar to eBay.com and a client application, which generates requests to the website. Grozev and Buyya (2013) employed RUBiS in the construction of a performance model of a 3-tier application in cloud environments and they provided an extension of the CloudSim toolkit. To validate our work, we used this CloudSim extensions, allowing modeling, simulation and performance evaluation of 3-tier applications in Cloud environments. We are interested in how to resource provision

Table 3
Fuzzy rules to extract knowledge from stakeholders and responses in Fuzzy decision tree.

Num	Rule
1	IF (the workload is Low & $SLAV_{RT}$ is Good & $SLAV_{Cost}$ is Good), THEN (SV is −1.3).
2	IF (the workload is Medium & $SLAV_{RT}$ is Good & $SLAV_{Cost}$ is Good), THEN (SV is −1.6).
3	IF (the workload is High & $SLAV_{RT}$ is Good & $SLAV_{Cost}$ is Good), THEN (SV is −1.8).
4	IF (the workload is Low & $SLAV_{RT}$is Bad & $SLAV_{Cost}$ is Good), THEN (SV is +0.4).
5	IF (the workload is Medium & $SLAV_{RT}$is Bad & $SLAV_{Cost}$ is Good), THEN (SV is +0.6).
6	IF (the workload is High & $SLAV_{RT}$is Bad & $SLAV_{Cost}$ is Good), THEN (SV is +0.4).
7	IF (the workload is Low & $SLAV_{RT}$is Very Bad & $SLAV_{Cost}$ is Good), THEN (SV is +1.8.
8	IF (the workload is Medium & $SLAV_{RT}$is Very Bad & $SLAV_{Cost}$ is Good), THEN (SV is +1.6).
9	IF (the workload is High & $SLAV_{RT}$is Very Bad & $SLAV_{Cost}$ is Good), THEN (SV is +1.5).
10	IF (the workload is Low & $SLAV_{RT}$ is Good & $SLAV_{Cost}$is Bad), THEN (SV is −1.4).
11	IF (the workload is Low & $SLAV_{RT}$is Bad & $SLAV_{Cost}$is Bad), THEN (SV is +0.4).
12	IF (the workload is Low & $SLAV_{RT}$is Very Bad & $SLAV_{Cost}$is Bad), THEN (SV is +1.4).
13	IF (the workload is Medium & $SLAV_{RT}$ is Good & $SLAV_{Cost}$is Bad), THEN (SV is −1.7).
14	IF (the workload is Medium & $SLAV_{RT}$is Bad & $SLAV_{Cost}$is Bad), THEN (SV is +0.6).
15	IF (the workload is Medium & $SLAV_{RT}$is Very Bad & $SLAV_{Cost}$is Bad), THEN (SV is +1.3).
16	IF (the workload is High & $SLAV_{RT}$ is Good & $SLAV_{Cost}$is Bad), THEN (SV is −1.9).
17	IF (the workload is High & $SLAV_{RT}$is Bad & $SLAV_{Cost}$is Bad), THEN (SV is +0.4).
18	IF (the workload is High & $SLAV_{RT}$is Very Bad & $SLAV_{Cost}$is Bad), THEN (SV is +1.2).
19	IF (the workload is Low & $SLAV_{RT}$ is Good & $SLAV_{Cost}$is Very Bad), THEN (SV is −1.5).
20	IF (the workload is Low & $SLAV_{RT}$is Bad & $SLAV_{Cost}$is Very Bad), THEN (SV is +0.4).
21	IF (the workload is Low & $SLAV_{RT}$is Very Bad & $SLAV_{Cost}$is Very Bad), THEN (SV is +1.1).
22	IF (the workload is Medium & $SLAV_{RT}$is Good & $SLAV_{Cost}$is Very Bad), THEN (SV is −1.8).
23	IF (the workload is Medium & $SLAV_{RT}$is Bad & $SLAV_{Cost}$is Very Bad), THEN (SV is +0.6).
24	IF (the workload is Medium & $SLAV_{RT}$is Very Bad & $SLAV_{Cost}$is Very Bad), THEN (SV is +1.0).
25	IF (the workload is High & $SLAV_{RT}$is Good & $SLAV_{Cost}$is Very Bad), THEN (SV is −2).
26	IF (the workload is High & $SLAV_{RT}$is Bad & $SLAV_{Cost}$is Very Bad), THEN (SV is +0.4).
27	IF (the workload is High & $SLAV_{RT}$is Very Ba d & $SLAV_{Cost}$I is Very Bad), THEN (SV is +0.8).

and load balance the incoming server-side requests. The RUBiS workload consists of sessions, each of which consists of a string of player requests.

5.1.1. Workload

In this paper, the real workload (RunEScape) and synthetic workload were used to evaluate the proposed approach. To simulate a real workload within the capacity of our testbed, we vary the number of players proportionally according to the RunEScape workload (Jagex, 2017). To create real data during one week, a sample was taken every 1 h approximately (Fig. 11). Incoming players increases during the nighttime and weekend more than the daytime and weekdays. The start time of workload is selected, randomly. Thus, the precise time of measurement points is unspecified. Nevertheless, we suppose that incoming requests will have regular patterns in every interval. Fig. 11 represents that the first 4 h and the last 10 h have high peaks. Therefore, the beginning of nighttime is considered the tenth hours from the measurement because the number of requests is making more from the tenth hour and end at the fourth hour in Fig. 11. Similarly, we can consider the weekend begin on the fourth day from the measurement by supposing incoming players increases during the weekend rather than the weekdays.

In this paper, for instance, we set the SW to one week. Indeed, one weak data is sampled as a training dataset for the prediction method and the following week data is used to test the accuracy of the prediction method. Moreover, we performed a time-stepped simulation of PI = 30 min. The workload analyzer predicts distribution parameters in every 30 min. Fig. 12(a) represents RunEScape real workload pattern in every 30 min during of a week. To create a synthetic workload, two periodic workloads with different strategies were used (Fig. 12(b) and 12(c)). The number of incoming players in the synthetic workload ranged from 500 to 20,000.

We have deployed in a local virtualized environment the PHP version of RUBiS with a standard non-clustered MySQL database in the backend. In addition, the MySQL database can be used as a Knowledge Base. It maintains data of the managed system and environment, adaptation goals, and relevant states that are shared by MAPE components. During the test, the monitor component monitors how the performance utilizations (in terms of cost and response time) of the servers change over time as a result of the executed workload. Based on that, we continue other phases of the MAPE-K control loop for auto-scaling decision-making.

Moreover, the specifications of the datacenter used for the simulation of the cloud environment are shown in Table 4. As a cloud provider, the data center takes on the role of MMOG hosting.

5.2. Performance metrics

Both accuracy and performance metrics were used to evaluate the proposed approach. The accuracy metrics for evaluating the prediction accuracy of the analyzer component were mean square error (MSE) (Wang and Bovik, 2009), root mean square error (RMSE) (Chai and Draxler, 2014), and R^2 prediction accuracy (Achelis, 2001).

Mean Square Error (MSE): This metric measures the average of the squares of the errors or deviations. It is always non-negative, and the values closer to zero are better. It is expressed as:

$$MSE = \frac{\sum_{i=1}^{n} \left(y_p(i) - y_m(i)\right)^2}{n} \tag{13}$$

where y_i is the actual value, $y'_i$is the predicted value and$n$ is the total number of observations from the beginning of the time series.

Root Mean Square Error (RMSE): This metric is used to identify the error value of the solution. Here, $y_m(i)$ shows the outcome that is obtained by ANFIS in time i and $y_p(i)$ shows the actual outcome of the system. n refers to the sample size that is used in the application and is expressed as:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} \left(y_p(i) - y_m(i)\right)^2}{n}} \tag{14}$$

Coefficient of determination (R^2): R^2 prediction accuracy metric is a measure of the goodness-of-fit of the prediction method. It is a statistical measure of how close the predicted values are to the actual values. It is expressed as:

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - y'_i)^2}{\sum_{i=1}^{n} (y_i - \overline{y_i})^2} \tag{15}$$

Performance metrics were applied to compare the proposed approach with other strategies, including the number of allocated VMs, average response time and cost as follows:

Allocated VMs: In the MMOG multi-tier application, an important

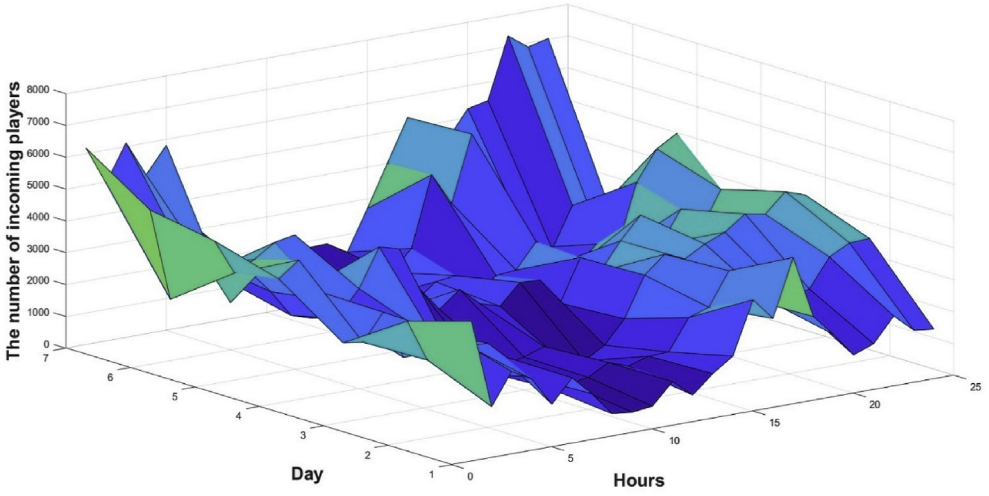


Fig. 11. Real data during a week.

metric in the evaluation of autonomous mechanisms is the number of VMs allocated to each tier (*VM_Allocate_Tire*) and their sum (*Total VMs*). This is expressed in Eq. (16), where $VM\_Allocate\_Tire_i$ denotes the number of VMs allocated to the *i*th tier.

$$Total\ VMs = \sum_{i=1}^{3} VM\_Allocate\_Tire_i \tag{16}$$

Response time: Players require a response time from autonomous mechanisms used by game providers that do not exceed the limit. If it does so, it will result in an SLA violation. The response time of a player request is defined as the amount of time taken to process the request from tier 1 to tier 3. More information about the desired response time of applications hosted in the cloud environment has been used by several authors (Aslanpour et al., 2017b; Safari and Khorsand, 2018a,2018b; Ghobaei-Arani et al., 2018a, 2018b).

Cost: The sum of the cost of allocating a VM and the cost of penalties for SLA violation is calculated as follows (Patel et al., 2016):

$$Total\ Cost = VM.Cost + PenaltyCost \tag{17}$$

$$VM.Cost = \sum_{i=1}^{M} VM\ Price_i \times (VM\_Time_i + VM\_Init\_Cost_i) \tag{18}$$

$$Penalty\_Cost = \sum_{i=1}^{M} T_i.\ \alpha Penalty \times T_i.\ Penalty \times Req.SLAV_{RT_i} \tag{19}$$

$$SLAV_{RT_i} = Response_i - Deadline_i \tag{20}$$

For the calculation of the cost related to the allocated VMs (as *VM.Cost*), $VMPrice_i$ is the cost of the VM for the *i*th request. VM_Time_i is the length of time each VM has been rented from the cloud provider to service for the player request. $VM\_Init\_Cost_i$ is the cost of the initial configuration of the VM for the *i*th request and *Penalty_Cost* is the penalty the provider pays to the player for a SLA violation. When calculating the *Penalty_Cost*, the player requests (*M*), $T_i.\ Penalty$ is the penalty rate, $T_i.\ \alpha Penalty$ is the penalty increase coefficient per repeated SLA violation and $Req.SLAV_i$ is the SLA violation. $Req.SLAV_{RT_i}$ is the difference in the time required to respond to the *i*th request and the time expected by the player (i.e., its deadline). *VM.Cost* and *Penalty_Cost* have been calculated for all VMs and player requests and the result is denoted as *Total Cost*.

5.3. Experimental analysis

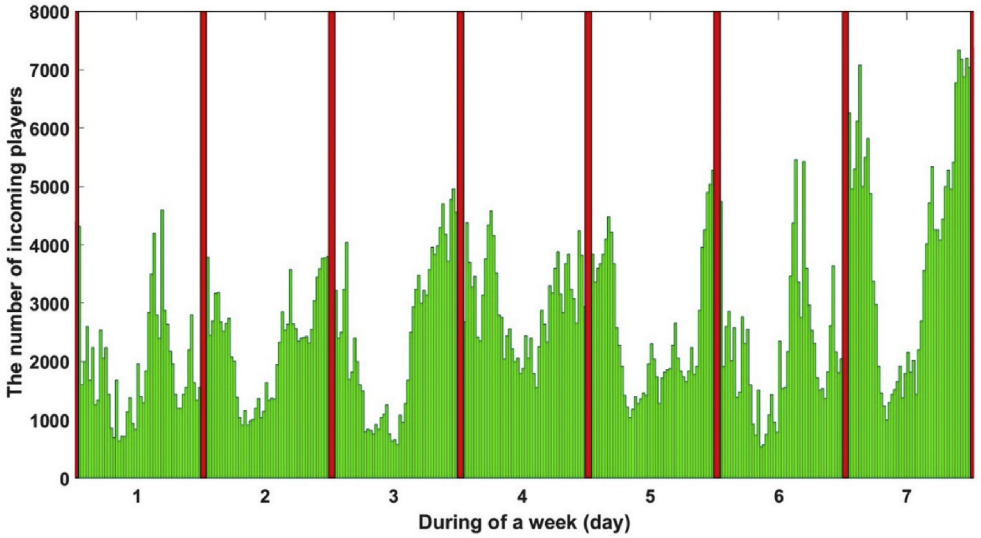
This section investigates the effectiveness of the proposed approach in compared with two other approaches; 1) System I, a dynamic resource provisioning for cloud-based gaming infrastructures (Marzolla et al., 2012) that uses a QN performance model to estimate the system response time for different configurations and a greedy algorithm in the analysis and planning phases. 2) System II, a machine-learning-based proactive approach for resource provisioning that uses a neural network-based predictor (for the client) in the analysis and planning phases, respectively (Nae et al., 2011). Evaluation and comparison are carried out from the two general aspects of prediction performance in the analysis phase (first scenario) and the overall performance of the autonomic resource provisioning approach (second scenario). To confirm the validity of the evaluations, the experiment was carried out with both real and synthetic workloads. The scenarios are shown in Table 5.

Evaluation and comparison are carried out from the two general aspects of prediction performance in the analyzer component (first scenario) and the overall performance of the autonomic resource provisioning approach (second scenario), as shown in Table 5. To confirm the validity of the evaluations, the experiment was carried out with both real and synthetic workloads.

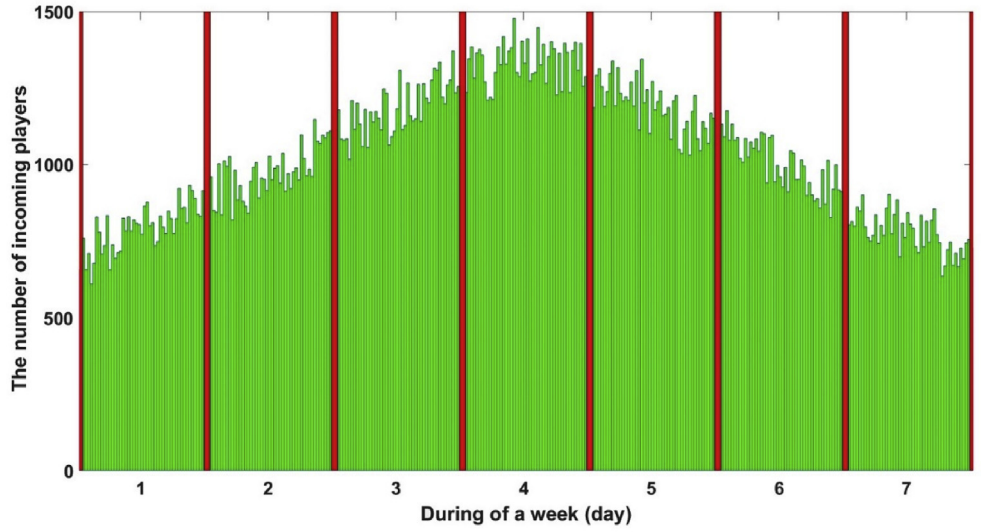
5.3.1. First scenario: evaluation of prediction accuracy

The performance of the analyzer component in the proposed framework was compared with support vector regression (SVR) and regression tree (RT). An important feature of SVR is that the solution is based only on the data points at the margin; the support vectors (SVs). The linear SVR can be extended to handle nonlinear problems when the data is first transformed into high dimensional feature space, reproducing kernel Hilbert space using a set of nonlinear kernel functions. Its goal is to find a function in which most error does not exceed a predefined threshold. In SVR, a real value is predicted using a tube bounded by the ± threshold. Bankole et al. (Bankole and Ajila, 2013) used SVR to predict response time given a specified load for individual workloads co-hosted on a shared storage system. Regression tree (RT) method proposed by Ref (Mori and Kosemura, 2001). is a way of clustering. It classifies the load range into several classes and decides which class the forecasted load belongs to according to the classification rules, then multi-layer perceptron is used to train the samples in every class.

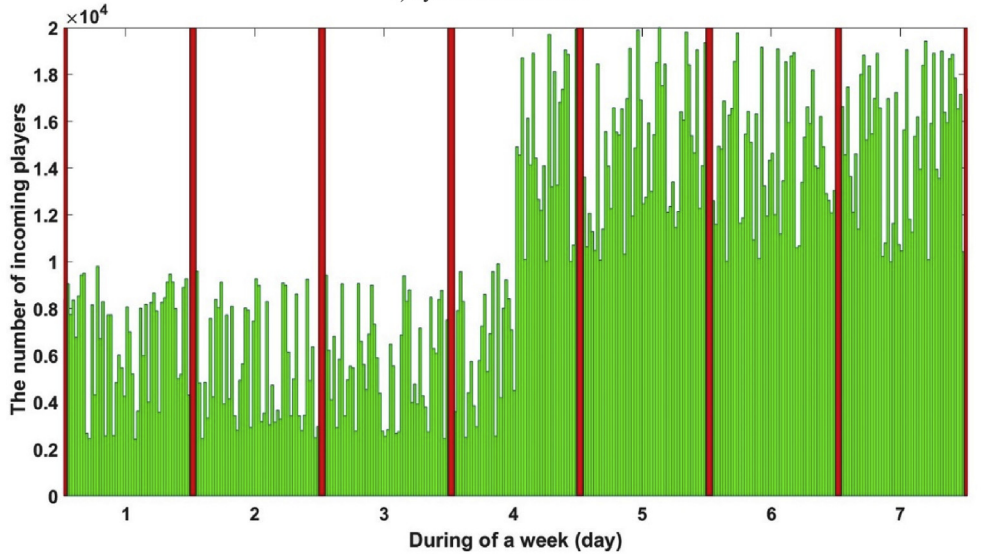
In this paper, in the analyzer component, the ANFIS workload predictor is designed by request numbers as the inputs and the future request number as the output. The first stage of the experiment is to select the input parameters and prepare the dataset for ANFIS input layer. The ANFIS usages hybrid-learning algorithm combined with back-propagation gradient descent and a least-square method to identify the parameters needed for ANFIS in order to reduce the network error. We prepared three datasets (real and two synthetic workloads) for ANFIS



a) RunEScape real workload pattern at time interval a week



b) Synthetic workload 1



c) Synthetic workload 2

Fig. 12. Real and synthetic workloads: a) RunEScape real workload b) Synthetic workload 1, c) Synthetic workload 2.

Table 4
The characteristics of Cloud Provider Data Centers.

Characteristic	Type
Architecture	X64
Operation system	Cloud Linux
VM management	XEN
Host parameters	MIPS = 6821, RAM = 256 GB, and Bandwidth = 10 Gbit/s

Table 5
Experimental scenarios.

Scenario	Goal	Workload	Metrics
First scenario	Evaluation of analysis phase	Real	MSE, RMSE, R^2
		Synthetic	MSE, RMSE, R^2
Second scenario	Evaluation of proposed approach	Real	allocated VMs, Response Time, and Cost
		Synthetic	allocated VMs, Response Time, and Cost

prediction model. We also used Gaussian member ship functions for the input set. The proposed autonomous resource-provisioning framework was first subjected to a RunEscape workload, and the analyzer component was allowed to analyze the workload and the relationship between players. We divided each input set into two sets. The first input set is used for training the network, and the other input set is used to test the ANFIS network. For training, we used 70% of the samples, and for testing, we used the remaining 30% samples. The collected data included 168 samples among which 117 samples were used for training and 51 samples for testing. The predictions results of the developed ANFIS model for the analyzer component as a function of the experimentally determined values of four different workloads were shown in Figs. (13-15), separately. As seen in Figs. (13-15 (a)), there is a very high correlation between the obtained results in this research by ANFIS modeling and the experimental data. The coefficient of determination about all data under three different workloads were 0.943, 0.988, and 0.997, respectively. In this study, to determine the accuracy rate for the all data set, we used the MSE criterion. As shown in Figs. (13-15 (b)), the MSE of ANFIS in the prediction of real RunEscape workload and two synthetic workloads are 1.6907, 8.318, and 701.4674, respectively. According to the values of the MSE, the model of ANFIS in the prediction of real RunEscape workload is selected as the best model. In Figs. (13-15 (c)), the standard deviation of error (St.D) is a measure of how widely values are dispersed from the average value. According to the obtained St.Ds, it can be concluded that the ANFIS model has a good potential to determine the future request number of an MMOG. Table 6 summarizes the best result achieved by ANFIS, RT, and SVR prediction models with their corresponding MSE, R^2 and RMSE error rate for real and synthetic workloads. The higher coefficient of determination and the lower amount of other criteria indicate better goodness of fit for the observations and strong prediction ability of the created model by ANFIS prediction model. The RT prediction model gives almost the same performance compared to ANFIS. Indeed, RT can build a tree structure with some leaf nodes to express the training experience in "if … Then" rules. By setting the upper limitation of dispersion and lower limitation of sample number in the leaf nodes, the validity of the rules is assured.

5.3.2. *Evaluation of proposed approach (second scenario)*

In this section, we evaluate the proposed algorithm described in Section 4.3. The proposed approach was evaluated in the form of three experiments for the number of allocated VMs, average response time, and cost metrics.

Experiment 1: Impact of scaling on the number of allocated VMs.

One of the most important evaluation metrics is the number of VMs allocated by the autonomic virtual machine-provisioning agent. The number of VMs allocated in a real and synthetic workload by either approach was assessed. The proposed framework begins to receive the workload of the MMOG players. Because the arrival rate of players varies, the autonomous virtual machine-provisioning agent takes action for the allocation and de-allocation of VMs so that the MMOG provider is provided with the resources it has requested for rental. Of course, a lower number of allocated machines is indicative of conscious performance without unnecessary scaling decisions. Hence, in different experiments, synthetic and real workloads were sent to the proposed framework. This study applied virtual machine provisioning in per-tier form; thus, the results obtained from the number of allocated VMs for real workload (Fig. 16(a)) and synthetic workload (Fig. 16(b) and 16(c)) at various frequencies are shown in each tier of MMOG application separately.

The gateway tier of the MMOG application required the greatest number of VMs because it has more interaction with players. The database tier required the minimum number of VMs because it used virtual memory or storage. Note that, in the database tier, storage is most important and uses a computing resource, while for the gateway and cellular tiers, the computing resources of RAM and MIPS are more important. The need for resource allocation was evident in the gateway tier, followed by the cellular and database tiers because players often open the game and do not engage in fights. This can also be understood in other web applications. An example is in an online store web application where user requests are often to browse, and no need is felt for access to the lower tiers of the application, such as the database.

The number of allocated VMs for a real workload and synthetic workload at various frequencies of system I, system II and proposed approaches are shown in Fig. 17. The increase in the number of allocated

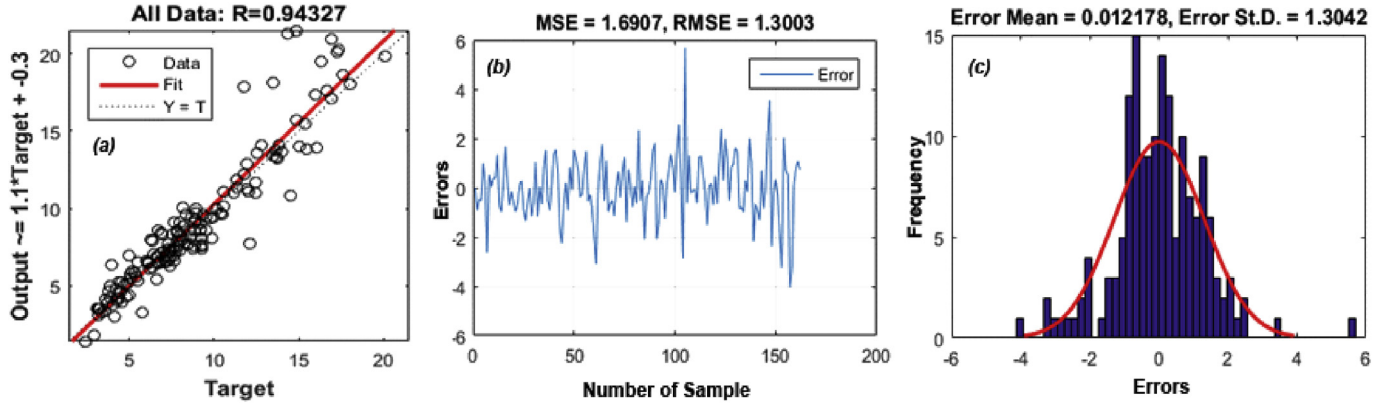


Fig. 13. The analyzer component equipped with ANFIS in the prediction of real RunEscape workload: a) Regression plot b) MSE and RMSE value of all data samples c) frequency errors of all data samples.

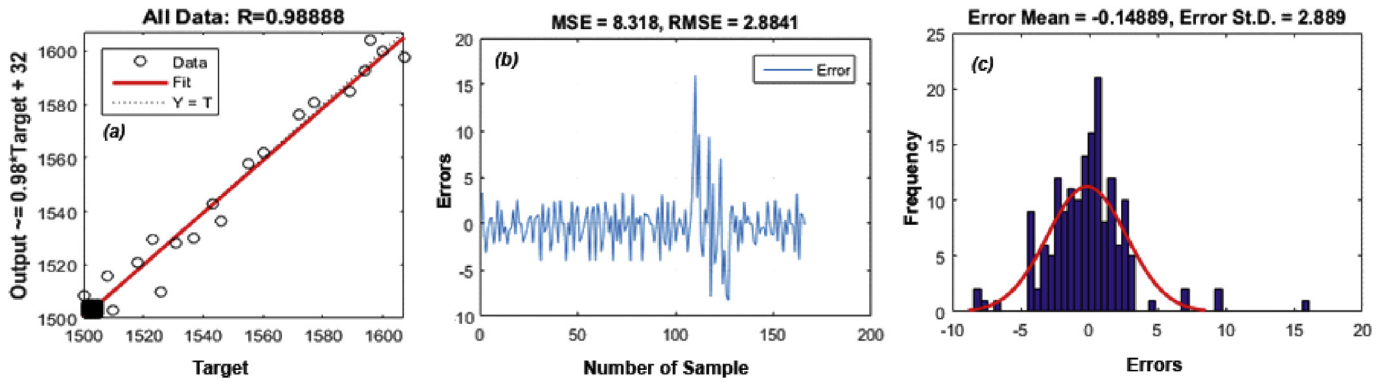


Fig. 14. The analyzer component equipped with ANFIS in the prediction of synthetic workload 1: a) Regression plot b) MSE and RMSE value of all data samples c) frequency errors of all data samples.

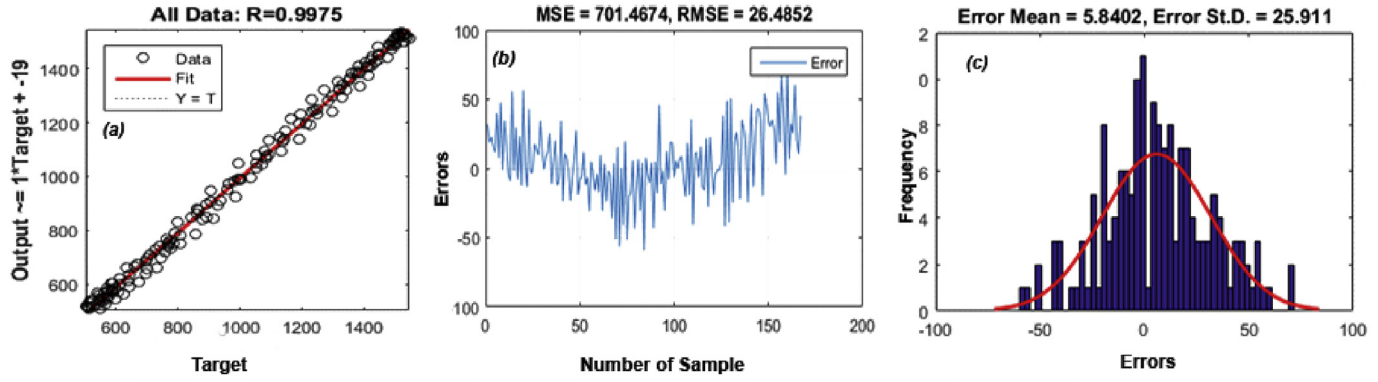


Fig. 15. The analyzer component equipped with ANFIS in the prediction of synthetic workload 2: a) Regression plot b) MSE and RMSE value of all data samples c) frequency errors of all data samples.

Table 6
The results of the performance of analyzer component equipped with SVR, regression tree and ANFIS in the prediction of real RunEscape workload and synthetic workload at different strategies.

Workload Type	Statistical Metrics	Prediction models		
		SVR	RT	ANFIS
RunEscape real workload	MSE	19.424	1.777	1.690
	RMSE	4.407	1.333	1.300
	R^2	0.668	0.940	0.943
Synthetic workload 1	MSE	24.31	19.45	8.31
	RMSE	4.930	4.411	2.884
	R^2	0.968	0.973	0.988
synthetic workload 2	MSE	412.844	341.062	199.2016
	RMSE	20.318	18.467	14.113
	R^2	0.9994	0.9995	0.9997

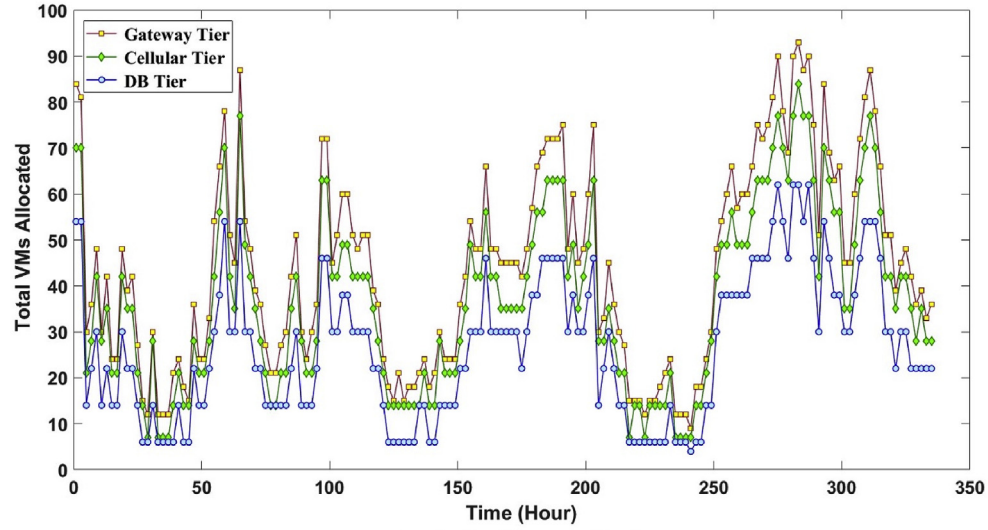
VMs in all three systems seems to be directly proportional to the workload arrival rate. At higher workload arrival rates, the system receives a higher number of requests per unit time, increasing the potential for allocating a higher number of VMs/hour. System II only considers the cost and system I considers a greedy policy (time or cost). The proposed approach uses the qualitative specification of elasticity rules to take various aspects of QoS (time and cost) into account. Because the MAPE-k loop is equipped with a fuzzy auto-scaling decision-maker, the proposed approach is more effective for auto-scaling MMOG applications to meet user QoS requirements while reducing infrastructure provider costs. As the number of concurrent users increases, so does the system response time, which in turn triggers reconfigurations resulting in more hosts being added to the appropriate tiers. When the number of concurrent users decreases, servers are de-allocated from the tiers.

Experiment 2: Impact of scaling on response time.

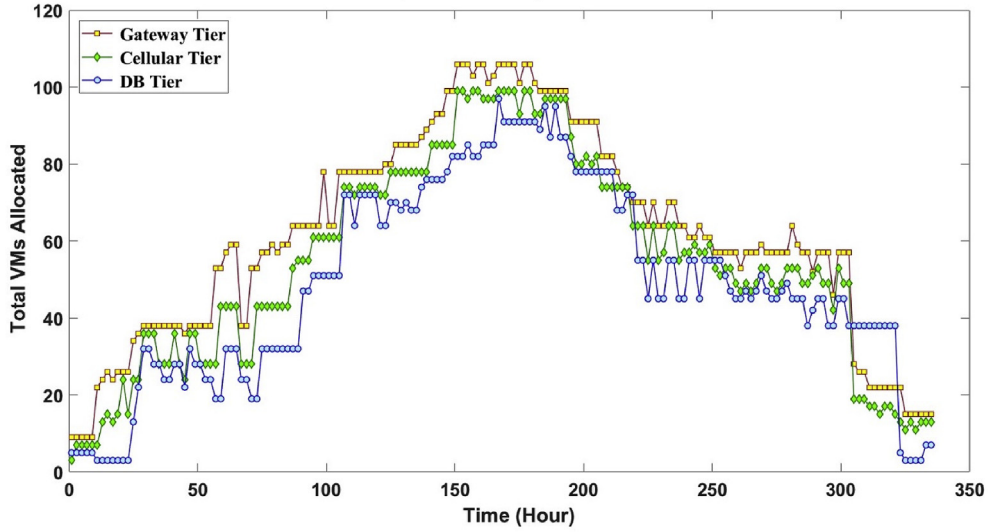
One of the most important metrics in virtual machine provisioning is response time. This criterion should be maintained within the limit of the SLA at a more optimal level to fairly reduce costs and consider MMOG player satisfaction. The results of the average response time of resources rented from the cloud provider were assessed as follows: The proposed framework was fed with the real workload and then the synthetic workload at different frequencies in order to measure and evaluate the response time of the proposed, system I and system II approaches. Comparison of results for the real and synthetic workload 1, 2 are shown in Fig. 18. For each change in input workload (concurrent input requests submitted by individual users), the corresponding response time varies between upper or lower bounds depending on the current load balance and number of available VMs. The proposed approach upgraded the number of VMs more accurately by detecting these fluctuations in response time and the average response time was less than the other approaches. These results indicate that fuzzy auto-scaling decision-making considers MMOG player satisfaction while avoiding insufficient virtual machine provisioning.

Experiment 3: Impact of scaling on cost.

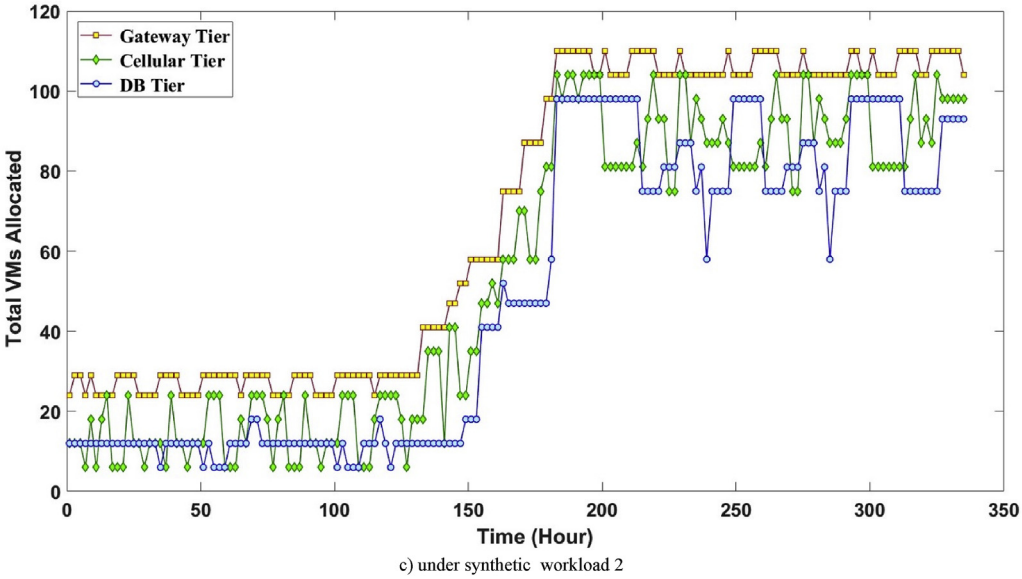
The average cost of the proposed approach was investigated and compared with system I and system II approaches. The results for the experiment for real and synthetic workloads 1, 2 are shown in Fig. 19. The figure shows the effect of varying the workload on the cost average parameter. The number of allocated VMs is proportional to the cost of the MMOG infrastructure and is paid by the MMOG operator to the Cloud



a) under RunEscape workload

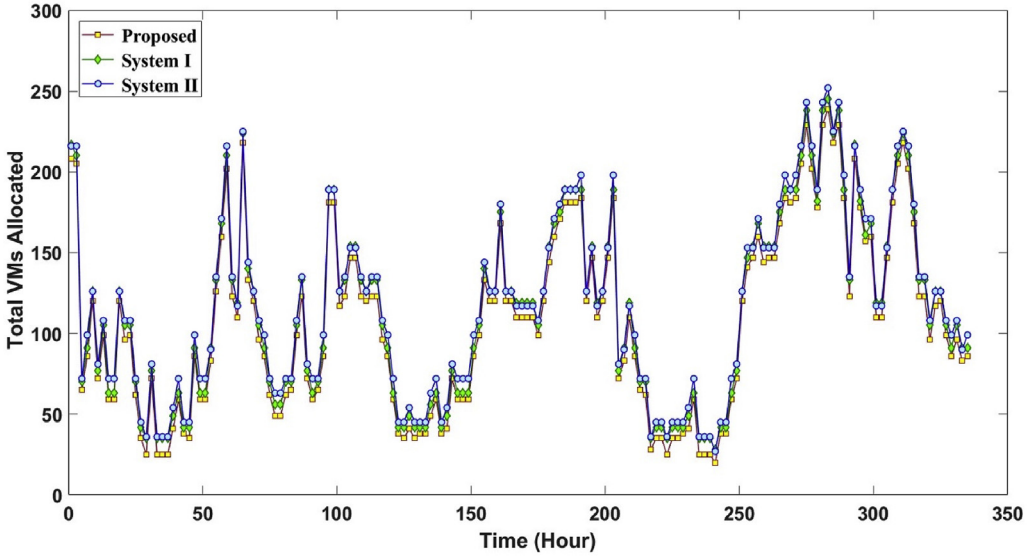


b) under synthetic workload 1

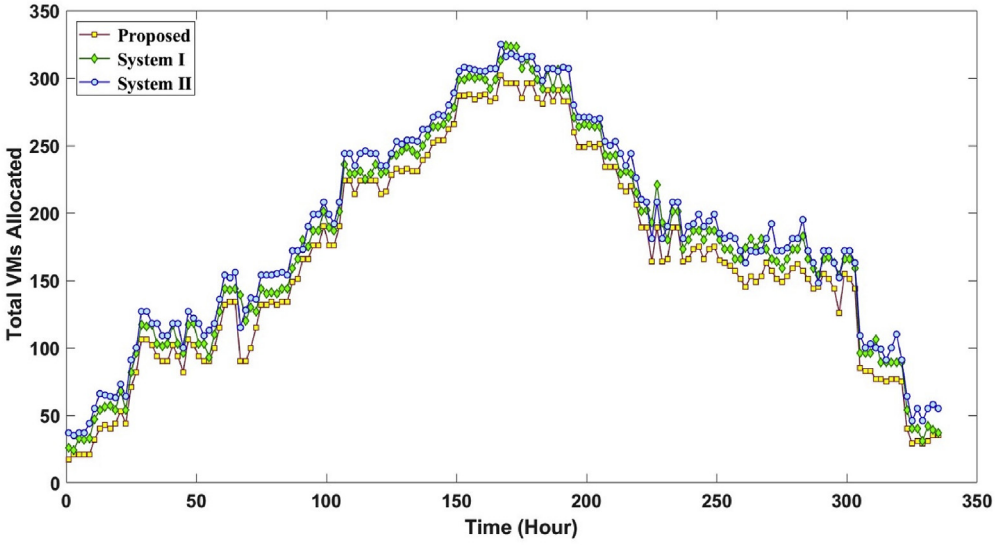


c) under synthetic workload 2

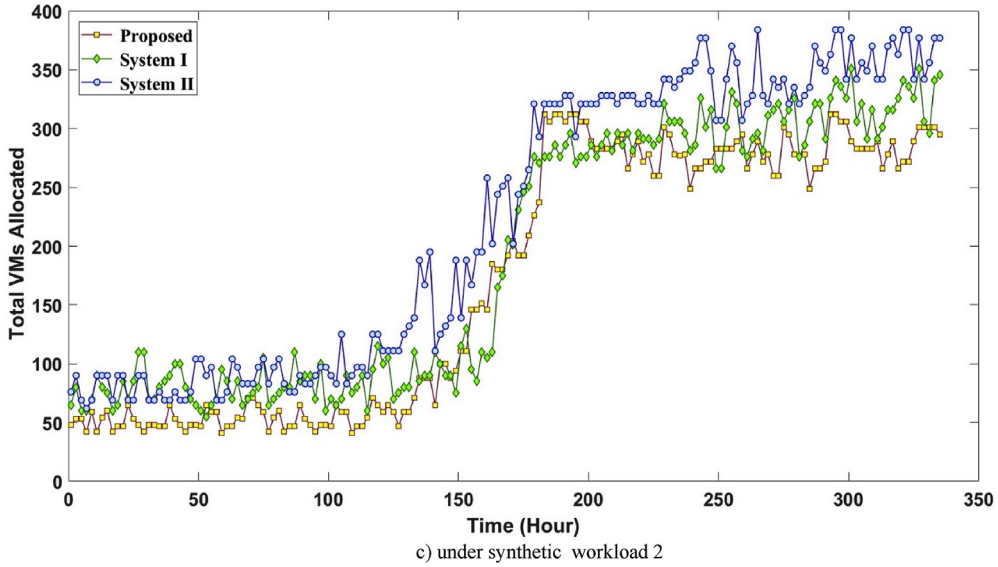
Fig. 16. Comparison between the numbers of allocated VMs in each MMOG tier under different workloads in the proposed approach.



a) under RunEscape workload



b) under synthetic workload 1



c) under synthetic workload 2

Fig. 17. Comparison between the numbers of allocated VMs in three approaches under different workload.

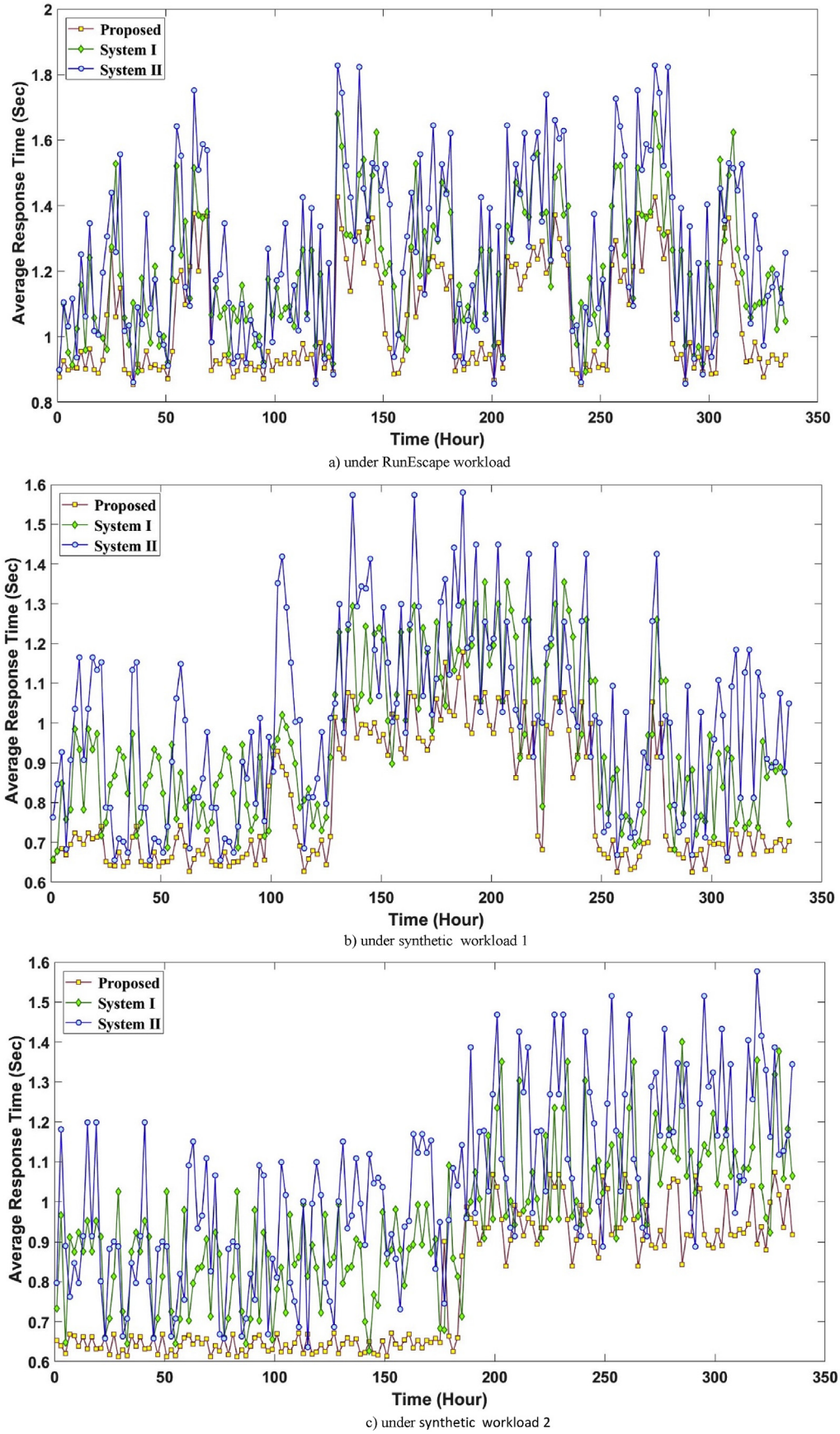
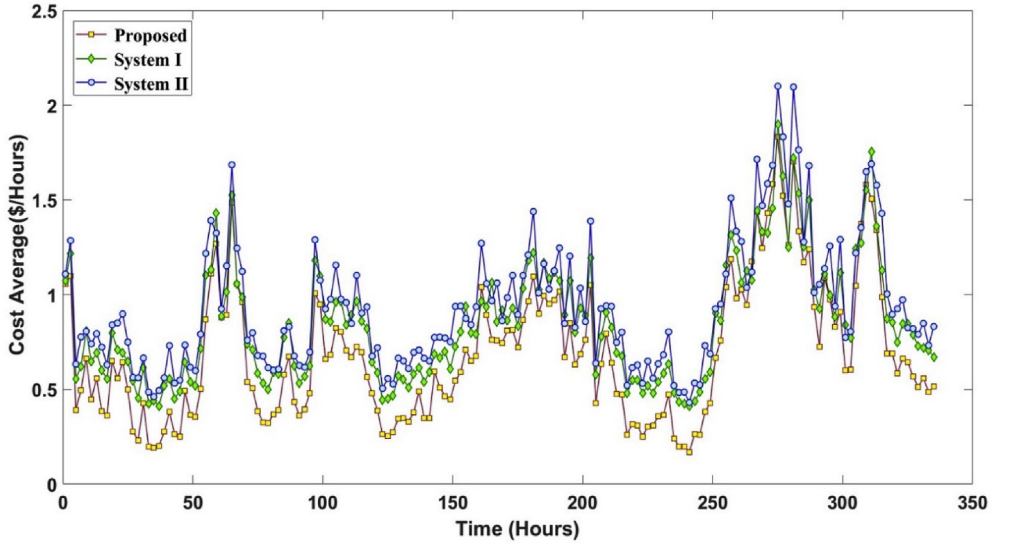
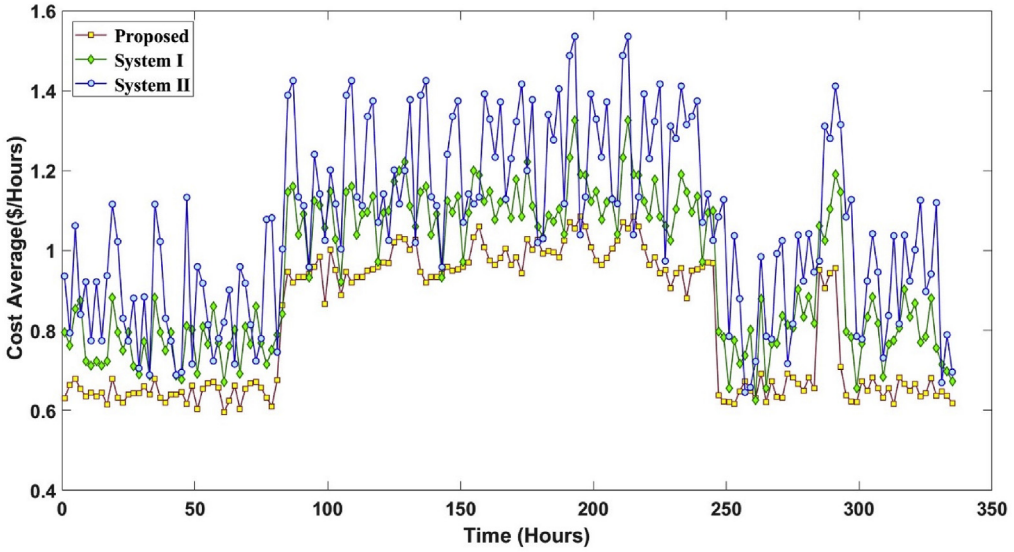


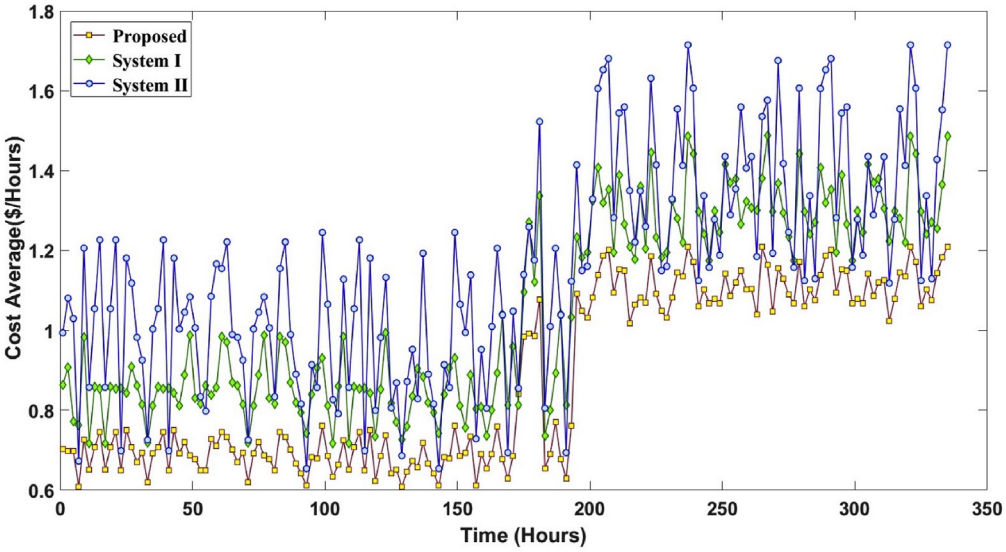
Fig. 18. Average response time during 336 extractions of different workload.



a) under RunEscape workload



b) under synthetic workload 1



c) under synthetic workload 2

Fig. 19. Average cost during 336 extractions of real workload.

provider. As seen, the proposed approach was more efficient in cost saving for the MMOG provider than the other approaches. This cost saving was attained by using parameters estimation of the probability distribution in the analysis phase and the fuzzy auto-scaling as the decision-maker, which considers the workload type, *SLAV_ Response Time*, and *SLAV_Cost* simultaneously. The proposed approach used the qualitative specification of elasticity rules and reduced the cost while providing an appropriate level of QoS to the game players.

6. Conclusion

The current MMOG ecosystem, which includes tens of millions of players across hundreds of games, forces game providers to also become game and infrastructure operators. To ensure that the user demand is satisfied at all times, the game operators resort to static resource provisioning: they build and maintain computing platforms of up to 10,000 machines located on several continents for a single MMOG. In this work, we propose an autonomous resource provisioning framework to manage data center resources. Firstly, we developed a workload prediction method that accurately estimates the future game requests from historical information using ANFIS prediction model. Our approach applies feedback from the latest observed workloads to a prediction model and updates the model parameter on the run. Then, we enhanced the performance of the planner component into the proposed framework by using a fuzzy decision tree technique as a decision maker. Evaluation and comparison are carried out from the two general aspects of prediction performance in the analyzer component (first scenario) and the overall performance of the autonomic resource provisioning approach (second scenario). To confirm the validity of the evaluations, the experiment was carried out with both real and synthetic workloads. We presented experiments, which demonstrate there is a very high correlation between the obtained results in this research by ANFIS prediction model and the experimental data. Indeed, the higher coefficient of determination and the lower amount of other criteria indicate better goodness of fit for the observations and strong prediction ability of the created model by ANFIS. In addition, it was demonstrated that the proposed approach is better able to reduce the cost of resource rental for the MMOG provider while meeting the QoS requirements of the players, especially response time. In future work, we plan to explore: integration of the proposed approach with task scheduling strategies, evaluation the proposed approach in a real cloud infrastructure such as OpenStack and extension of the planning phase of the control MAPE-k control loop by using of the self-learning fuzzy auto-scaling decision-maker and the Bayes learning techniques.

Compliance with ethical standards

Funding

There are currently no Funding Sources in the list for this paper.

Conflict of interest

We have no conflict of interest to declare.

Ethical approval

All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards.

Ethical approval

This article does not contain any studies with human participants or animals performed by any of the authors.

Informed consent

Informed consent was obtained from all individual participants included inthe study.

References

Achelis, S.B., 2001 Apr. Technical Analysis from A to Z. McGraw Hill, New York.
Amiri, M., Mohammad-Khanli, L., 2017 Mar 15. Survey on prediction models of applications for resources provisioning in cloud. J. Netw. Comput. Appl. 82, 93–113.
Aslanpour, M.S., Ghobaei-Arani, M., Toosi, A.N., 2017 Oct 1. Auto-scaling web applications in clouds: a cost-aware approach. J. Netw. Comput. Appl. 95, 26–41.
Aslanpour, M.S., Dashti, S.E., Ghobaei-Arani, M., Rahmanian, A.A., 2017. Resource provisioning for cloud applications: a 3-D, provident and flexible approach. J. Supercomput. 1–32.
Bankole, A.A., Ajila, S.A., 2013 May 5. Predicting cloud resource provisioning using machine learning techniques. In: Electrical and Computer Engineering (CCECE), 2013 26th Annual IEEE Canadian Conference on. IEEE, pp. 1–4.
Basiri, M., Rasoolzadegan, A., 2018 Apr. Delay-aware resource provisioning for cost-efficient cloud gaming. IEEE Trans. Circuits Syst. Video Technol. 28 (4), 972–983.
Buyya, R., Murshed, M., 2002 Nov 1. Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing. Concurrency Comput. Pract. Ex. 14 (13-15), 1175–1220.
Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A., Buyya, R., 2011 Jan. CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. Software Pract. Ex. 41 (1), 23–50.
Chabaa, S., Zeroual, A., Antari, J., 2009 Nov 15. ANFIS method for forecasting internet traffic time series. In: Mediterranean Microwave Symposium (MMS), pp. 1–4 (Tangiers).
Chai, T., Draxler, R.R., 2014 Jun 30. Root mean square error (RMSE) or mean absolute error (MAE)?–Arguments against avoiding RMSE in the literature. Geosci. Model Dev. (GMD) 7 (3), 1247–1250.
Cheng, C.H., Wei, L.Y., 2010 Jun 1. One step-ahead ANFIS time series model for forecasting electricity loads. Optim. Eng. 11 (2), 303–317.
Chiu, S.L., 1994 Jan 1. Fuzzy model identification based on cluster estimation. J. Intell. Fuzzy Syst. 2 (3), 267–278.
Dhib, E., Boussetta, K., Zangar, N., Tabbane, N., 2017 Mar 29. Cost-aware virtual machines placement problem under constraints over a distributed cloud infrastructure. In: Sixth International Conference on Communications and Networking (ComNet). IEEE, pp. 1–5.
Farlow, S., Trahan, J.L., 2018 Aug 7. Periodic load balancing heuristics in massively multiplayer online games. In: Proceedings of the 13th International Conference on the Foundations of Digital Games. ACM, p. 29.
Gao, Y., Wang, L., Xie, Z., Guo, W., Zhou, J., 2018 Nov 12. Energy-efficient and quality of experience-aware resource provisioning for massively multiplayer online games in the cloud. In: International Conference on Service-Oriented Computing. Springer, Cham, pp. 854–869.
Ghobaei-Arani, M., Jabbehdari, S., Pourmina, M.A., 2016 Sep 1. An autonomic approach for resource provisioning of cloud services. Clust. Comput. 19 (3), 1017–1036.
Ghobaei-Arani, M., Jabbehdari, S., Pourmina, M.A., 2018a. An autonomic resource provisioning approach for service-based cloud applications: a hybrid approach. Future Gener. Comput. Syst. 78, 191–210.
Ghobaei-Arani, M., Rahmanian, A.A., Shamsi, M., Rasouli-Kenari, A., 2018b. A learning-based approach for virtual machine placement in cloud data centers. Int. J. Commun. Syst. 31 (8), e3537.
Grozev, N., Buyya, R., 2013 Sep 26. Performance modelling and simulation of three-tier applications in cloud and multi-cloud environments. Comput. J. 58 (1), 1–22.
Huebscher, M.C., McCann, J.A., 2008 Aug 1. A survey of autonomic computing—degrees, models, and applications. ACM Comput. Surv. 40 (3), 7.
Jacob, B., Lanyon-Hogg, R., Nadgir, D.K., Yassin, A.F., 2004 Apr. A practical guide to the IBM autonomic computing toolkit. IBM Redbooks 4, 10.
Jagex, Ltd, February 2017. Runescape. http://www.runescape.com/.
Jang, J.S., 1991 Jul 14. Fuzzy modeling using generalized neural networks and kalman filter algorithm. AAAI 91, 762–767.
Jang, J.S., 1991. A self-learning fuzzy controller with application to automobile tracking problem. In: Proc. Of IEEE Roundtable Discussion on Fuzzy and Neural Systems, and Vehicle Application, vol. 10 page paper.
Jang, J.R., 1997. MATLAB: Fuzzy Logic Toolbox User's Guide: Version 1. Math Works.
Jang, J.S., Sun, C.T., 1993. Predicting chaotic time series with fuzzy if-then rules. In: Fuzzy Systems, 1993., Second IEEE International Conference on. IEEE, pp. 1079–1084.
Jia, M., Liang, W., 2018 Oct 25. Delay-sensitive multiplayer augmented reality game planning in mobile edge computing. In: Proceedings of the 21st ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems. ACM, pp. 147–154.
Khorsand, R., Ghobaei-Arani, M., Ramezanpour, M., 2018. FAHP approach for autonomic resource provisioning of multitier applications in cloud computing environments. Softw. Pract. Ex. 48 (12), 2147–2173.
Khorsand, R., Safi-Esfahani, F., Nematbakhsh, N., Mohsenzade, M., 2017 Jun 1. ATSDS: adaptive two-stage deadline-constrained workflow scheduling considering run-time circumstances in cloud computing environments. J. Supercomput. 73 (6), 2430–2455.

Khorsand, R., Safi-Esfahani, F., Nematbakhsh, N., Mohsenzade, M., 2017 Oct 1. Taxonomy of workflow partitioning problems and methods in distributed environments. J. Syst. Softw. 132, 253.
Lin, Y., Shen, H., 2017 Feb 1. CloudFog: leveraging fog to extend cloud gaming for thin-client MMOG with high quality of service. IEEE Trans. Parallel Distrib. Syst. 28 (2), 431–445.
Marzolla, M., Ferretti, S., D'angelo, G., 2012 Oct 1. Dynamic resource provisioning for cloud-based gaming infrastructures. Comput. Entertain. 10 (1), 4.
Masugi, M., Takuma, T., 2007 Jun 1. Using a Volterra system model to analyze nonlinear response in video-packet transmission over IP networks. Commun. Nonlinear Sci. Numer. Simul. 12 (3), 411–421.
Maurer, M., Breskovic, I., Emeakaroha, V.C., Brandic, I., 2011 Jun 28. Revealing the MAPE loop for the autonomic management of cloud infrastructures. In: Computers and Communications (ISCC), 2011 IEEE Symposium on. IEEE, pp. 147–152.
Meiländer, D., Gorlatch, S., 2018 Sep 1. Modeling the scalability of real-time online interactive applications on clouds. Future Gener. Comput. Syst. 86, 1019–1031.
Mori, H., Kosemura, N., 2001. Optimal regression tree based rule discovery for short-term load forecasting. In: Power Engineering Society Winter Meeting, 2001. IEEE, vol. 2. IEEE, pp. 421–426.
Nae, V., Iosup, A., Prodan, R., 2011 Mar. Dynamic resource provisioning in massively multiplayer online games. IEEE Trans. Parallel Distrib. Syst. 22 (3), 380–395.
Patel, M., Chaudhary, S., Garg, S., 2016. Machine learning based statistical prediction model for improving performance of live virtual machine migration. J. Eng. 2016.
Pattipati, K.R., Kostreva, M.M., Teele, J.L., 1990 Jul 1. Approximate mean value analysis algorithms for queuing networks: existence, uniqueness, and convergence results. J. Assoc. Comput. Mach. 37 (3), 643–673.
Prodan, R., Iosup, A., 2016 Nov 1. Operation analysis of massively multiplayer online games on unreliable resources. Peer-to-Peer Netw. Appl. 9 (6), 1145–1161.
Safari, M., Khorsand, R., 2018a. Energy-aware scheduling algorithm for time-constrained workflow tasks in DVFS-enabled cloud environment. Simulat. Model. Pract. Theor. 87, 311–326. Jul 20.
Safari, M., Khorsand, R., 2018b. PL-DVFS: combining Power-aware List-based scheduling algorithm with DVFS technique for real-time tasks in cloud computing. J. Supercomput. 1–23.
Takagi, T., Sugeno, M., 1985 Jan. Fuzzy identification of systems and its applications to modeling and control. IEEE Trans. Syst. Man Cyber. (1), 116–132.
Van Den Bossche, B., Verdickt, T., De Vleeschauwer, B., Desmet, S., De Mulder, S., De Turck, F., Dhoedt, B., Demeester, P., 2006 May 26. A platform for dynamic microcell redeployment in massively multiplayer online games. In: In Proceedings of the 2006 International Workshop on Network and Operating Systems Support for Digital Audio and Video. ACM, p. 3.
Wang, Z., Bovik, A.C., 2009 Jan. Mean squared error: love it or leave it? A new look at signal fidelity measures. IEEE Signal Process. Mag. 26 (1), 98–117.
Wu, H., Zhang, W., Zhang, J., Wei, J., Huang, T., 2013 Aug 1. A benefit-aware on-demand provisioning approach for multi-tier applications in cloud computing. Front. Comput. Sci. 7 (4), 459–474.
Zhai, J., Wang, X., Zhang, S., Hou, S., 2018 Oct 1. Tolerance rough fuzzy decision tree. Inf. Sci. 465, 425–438.

Mostafa Ghobaei-Arani received the B.S. and M.S. degrees in Software Engineering from Islamic Azad University (North Tehran Branch)in Iran, in 2009 and 2011, respectively. He also received the Ph.D. degree in Software Engineering from Islamic Azad University, Science and Research Branch, Tehran, Iran in 2016. His research interests include Distributed Computing, Cloud Computing, Autonomic Computing, Edge/Fog Computing, Exascale Computing, Soft Computing, and IoT.

Reihaneh Khorsand received her Ph.D. (Software engineering) in 2017 from Department of Computer Engineering, Science and Research Branch, Islamic Azad University, Tehran, Iran. She is assistant professor of Computer Engineering Department, Dolatabad Branch, Islamic Azad University, Isfahan, Iran. She has more than 7 years of teaching and working experience in distributed systems development. Her research interests are software engineering, distributed computing, scheduling and business workflow management.

Mohammadreza Ramezanpour received his B.Sc. degree in Computer Engineering from Islamic Azad University, Kashan, Iran. His M.Sc. degree from Islamic Azad University, Arak, Iran, and his Ph.D. degree in computer engineering from Science and Research Branch, Islamic Azad University, Tehran, Iran. His research interests include computer vision and image processing, pattern recognition and video coding standards