Journal of Engineering Science and Technology
Special Issue on ICCSIT 2018, July (2018) 58 - 77

A FUZZY MODEL FOR DETECTING AND PREDICTING CLOUD QUALITY OF SERVICE VIOLATION

HASSAN MAHMOOD KHAN, GAIK-YEE CHAN*, FANG-FANG CHUA

Faculty of Computing and Informatics, Multimedia University,
Persiaran Multimedia, 63100, Cyberjaya, Selangor, Malaysia
*Corresponding Author: gychan@mmu.edu.my

Abstract

Cloud computing offers cost-effective, on-demand and pay as you use IT services to service consumers based on Service Level Agreements (SLAs). The cloud services offered by cloud providers raise the need for Quality of Service (QoS) monitoring to ensure that SLAs are maintained for accountability. As a result, service performance, which includes availability, response time and throughput, is one of the core concerns of the cloud users. This paper proposes a fuzzy-based model for QoS violation detection and prediction of service response time and duration for ensuring continuous service availability. Our experimental results show small Root Mean Squared Error (RMSE) of less than 0.50, thus confirming the greater than 99% detection and prediction accuracy for QoS violation. This fuzzy model with 16 fuzzy rules of detecting and predicting QoS violation for Software as a Service (SaaS) allows the cloud administrator to make accurate decisions for appropriate remedial action. Consequently, accountability is taken care of with a win-win benefit for both the cloud users and service providers. This contributes towards a different dimension in the measurement of quality of cloud services where accountability is taken into consideration.

Keywords: Accountability, Cloud monitoring, Fuzzy logic, Quality of service, Service level agreement.

1. Introduction

Cloud computing provides convenient and real-time access to a network of computing resources which are easily configurable. The resources involved include network, servers, data storage, and software as a service. These resources can be provisioned and released rapidly with little interaction and effort from the cloud service provider [1]. A service resembles a functionality performed by a "service provider" to facilitate a "service requester" or a consumer [2]. This interaction between both entities, which performs transferring value from the provider to the consumer, is known as service provisioning. Cloud consumers can deploy their applications over a pool of network resources provided by the cloud providers with practically no capital investment cost. Additionally, cloud users pay only a low operating cost proportional to the actual use of the resources

For example, many commercial cloud service providers such as Amazon Web Services, Microsoft Azure, Salesforce.com and Google App Engine, have their resources running on millions of physical hosts. Each of these hosts could be hosting many Virtual Machines (VMs) whereby they can be invoked or removed dynamically [3]. In addition, cloud hypervisors are used to handle resource provisioning that includes mapping and scheduling the instantiated VMs residing on the cloud's physical servers [4]. These technical and economic benefits of the on-demand capability of cloud computing have given rise to possible migration of traditional enterprise computing to cloud computing in recent years.

The lack of trust by many customers toward cloud computing represents a big challenge that could hinder the migration of traditional enterprise computing to cloud computing. Although research in preventive tools and techniques of cloud privacy and security is active there is still lacking in detective techniques related to auditability and accountability [5]. Trust is defined as the "level of confidence in something or someone". Therefore, trust in cloud computing is to establish users' confidence technically and psychologically in cloud service. The components that effects trust in cloud computing are Security, Privacy, Accountability, and Auditability. Each of these components has preventive and detective practices in general but for a cloud environment, they have complexities, which include virtual-to-physical mapping, logging of operating and file systems' perspectives, log size and scale, and the dynamicity of cloud environments.

Another study by Theoharidou et al. [6] referred to the migration of data and application over to the cloud environment would expose enterprises to new threats and vulnerabilities that need to be properly dealt with. Privacy risks and the trustworthy environment is a complex challenge that needs accountability, rules enforcement and penalties from the auditors and independent authorities. Security and privacy is a core risk concern of cloud service consumers that need a comprehensive risk assessment for trust building. Multi-tenancy, data duplication and dynamic nature of the cloud make it complex for consumers to accept it readily. Hence, according to Theoharidou et al. [6], there is a need for a comprehensive risk assessment in accordance with privacy threats to enable cloud accountability and privacy compliance.

Cloud services are needed to describe and realize in terms of functional capabilities as well as service quality. Cloud service quality consists of several related attributes that include availability, security and response time of the service. These quality attributes are an important factor to grade cloud service providers and

are the key factors in selecting the best service provider among similar functionality service providers. Quality attributes are the base to define a contract between two parties to ensure that their service expectations are fulfilled. Such contracts are used for service management systems to monitor and assess the service quality when a service is being executed. Enforcement of such contracts could be through triggering adequate actions, for example, upscaling the relevant resource, substitute or recompose the erroneous service or apply settlement action. It also helps in defining final costs, penalties payable by either the service provider or consumer and negotiation for termination of the contract.

QoS includes an objectively measurable quality attribute such as response time and is explicitly defined in the SLA. The service quality is dynamic in nature with respect to server functionality, so the value of a service quality attribute may change by not affecting the main functionalities of the service. To ensure QoS to the cloud users, monitoring of cloud resources, data and SLAs are needed for effective and smooth operations in the cloud environment [7]. Additionally, dynamic monitoring of quality attributes such as availability, throughput, and response time is essential [8-10] as dynamic monitoring of these attributes could provide continuous and real-time information about the cloud services and resources, not only on Platform as a Service (PaaS), Infrastructure as a Service (IaaS), but specifically also for Software as a Service (SaaS). Deviation from the agreed QoS ranges stated in the SLA may cause a system failure, which leads to customers' dissatisfaction. Therefore, for cloud services to be trustworthy, effective mechanisms that can monitor performances and detect SLA violations are required. With these mechanisms in place, the trust level between the end users and the cloud service providers shall be increased.

This research focuses on cloud QoS violation detection, prevention, and prediction using machine learning technique to better provide a firm basis for remedial action. In this paper, we propose a fuzzy logic-based model for SaaS's QoS violation detection, prevention, and prediction mainly on two attributes namely response time and throughput assuming continuous service availability This paper is organized as follows, Section 2 describes related works, Section 3 presents the methodology and introduces the proposed model, Section 4 describes the experimentation for performance evaluation and analysis of results and Section 5 concludes and discusses future work.

2. Background Study and Related Work

This section provides a literature review of related works on cloud resource monitoring, quality of service for SaaS and detection of QoS violations.

2.1. Trustworthy cloud services based on service level agreement

Service Level Agreement (SLA) serves as a contract for specifying cloud services for both the service providers and service consumers. SLA lists the QoS attributes, guarantees, and obligations involving both parties. The QoS attributes and guarantees are Service-Level Objectives (SLOs) which are criteria for QoS metrics that are measured through quantitative values. A set of SLOs based on specific QoS metrics represents a specific Service Level (SL). The cloud service provider provides various SLs to be defined in an SLA. This SLA then represents different modes of service and can be executed in different time periods. If the agreed SL is violated or surpassed, auto-scaling or downgrading of service may occur.

Consequently, the service provider shall have the obligation to execute the agreed SLA and remedial action is thus guaranteed as well.

Therefore, effective mechanisms for monitoring cloud service performances and detection of SLA violations are urgently in need. Several types of research have been conducted in this domain. For example, research in [11] has presented a Business Process Execution Language (BPEL) based monitoring framework, which monitors Cloud-based Web services. This framework gathers Cloud's information, analyses them and provides corrective measures whenever violations in SLA are detected. This monitoring during run-time is based on workflow patterns generated by BPEL.

A study by Qazi et al. [12] states that SLA violations are a trust or accountability problem between the service provider and service consumer. If this accountability problem is not properly dealt with, it might lead to financial or reputation loss of the parties involved. To claim for a service violation, the consumers must show proof of violation, which involves overhead, thus is time-consuming as well as expensive if done manually. Their study proposed a method to analyze and apply enforcement of SLA using fair exchange for the three parties who are involved, the service provider, service subscriber and Trusted Third Party (TTP). Through this method, the protocols and monitoring data parameters are defined, and data flow is managed through TTP periodically with least overheads. Their approach provides dispute resolution protocol and calculates penalties for SLA violations, thus maintaining accountability.

Another study by Michlmayr et al. [13] presented a framework that monitors both client- and server-side QoS. This framework makes use of event processing that could detect current QoS values and possible violations of SLAs. If a possible violation occurs, an adaptive action such as hosting new service instances to avoid undesired QoS is triggered. Furthermore, the cost is another parameter for the economic feasibility of the cloud services as mentioned in [14]. Typically, cloud provider charges VMs usage as whole VM-hours and user must feed the bill for the entire hour even if a VM is not used for the entire hour. Additionally, prioritization of quality attributes in SLAs is very important. As a result, Chong et al. [15] used a fuzzy Analytic Hierarchy Process (AHP) to identify and prioritize quality attributes for pre-negotiation of SLA regarding to the service consumers' needs. A set of quality attributes is defined, and the service stakeholders rank the attributes according to their own use and priority. This method of incorporating all stakeholders' involvement surely helps in providing a better SLA.

2.2. Cloud resource monitoring and violation detection

For ensuring that the software and hardware deployed by the cloud service providers are running at the satisfactory quality level as agreed in the SLA, effective cloud resource monitoring mechanisms are in need. This mechanism should be able to gather QoS parameter values and help to detect QoS violations. Typically, if a system behaves in an abnormal manner, corrective respond, and remedial action should be taken. Cloud resource monitoring is also the key to maintain trust among the cloud parties such as Cloud facilitators (CF), Cloud service provider (CSP) and Cloud service consumer (CSC). Efficient and effective services must be provided by the CF to the CSP and likewise, efficient and effective services must be provided by CSP to the CSC. In this way, all the three parties are accountable for each other. This accountability could be achieved through SLA.

Many diverse IT infrastructure (network, computer nodes) monitoring tools are available long before cloud computing evolves. Some tools are specialized for a particular domain, for example, clusters and grid computing monitoring. Many of these resource monitoring tools are now used for cloud resource monitoring at some abstraction level, for example, general-purpose infrastructure monitoring applications. These infrastructure monitoring applications measure metric values gathered from component and server under surveillance. The server collects stores and then analyses data. Some monitoring tools even generate graphs, trending reports and SLA reports based on recorded data [16]. Nagios has created an industry benchmark level monitoring tool. This tool monitors cloud infrastructure. It could be initiated by executing the Nagios Remote Plugin Executor (NRPE) at regular intervals. For passive monitoring, Nagios Service Check Acceptor (NSCA) could be used [17].

NSCA is initiated through external applications. Many cloud service providers also provide diverse monitoring as a service for better management of applications. In addition, many cloud service providers provide their own monitoring facility to their customers. For example, Amazon Cloud Watch or ACW [18] is used to monitor Amazon Web Services (AWS). Azure Fabric Controller or AFC is used to monitor Azure-based cloud resources, hosted websites, web services, and Window Azure storage facility [19]. Both ACW and AFC enable user-defined metrics to be monitored from a set of available metrics. Additionally, there are some clouds monitoring services that provide cloud monitoring tools for the diverse cloud platform. For example, Meddeb [20] is used for monitoring Amazon EC2, S3 Web Services, Rackspace Cloud Microsoft Azure, Salesforce CRM, GoogleAppEngine, and GoogleApps. Monitis [21] mentioned that Nimsoft is used for monitoring Rackspace Cloud and Amazon EC2/AWS. CloudKick [22] is used for monitoring GoGrid, Amazon EC2, and Rackspace Cloud.

2.3. SaaS's quality of service

ISO 9126 model presents software quality attributes that are applicable to standalone software. However, the cloud service composition and exposure to the consumer is different from that of the standalone software [23]. Therefore, to make use of the ISO 9126 model, there is a need to adapt the model for distinctive attributes relevant to cloud services. This is due to cloud services only exposes their external attributes to the consumers in advertisement and utilization phases. In this case, external quality attributes such as security, privacy, performance, and reliability are applicable, but some other internal quality attributes, for example, changeability is not applicable as the internal but hidden code is required in service-oriented programming. Consequently, the ISO 9126 [23] quality model does not adequately represent SaaS's service quality [24].

Monitoring resources for quality compliances serve as an important procedure in cloud computing, especially for SaaS. Resource monitoring could determine whether users' expectations have been met and become an important factor in deciding whether to continue or discontinue paying for the service. For example, [8] used QoS attributes such as response time, availability for estimation of cloud cost effectiveness. In another study by Zheng et al. [9] they discussed response time, throughput, and failure probability for cloud services QoS ranking prediction. For cloud modelling, Chen and Bahsoon [10] used availability, response time and throughput. In the research by Khan et al. [25], a framework is proposed to deal with accountability and trust among cloud parties based on availability and

response time whereby SLA's QoS metrics and their acceptable values are defined. Consequently, through the monitoring of these metrics and values, the degree of QoS violation or no-violation could be identified. The ability to detect QoS violation helps in developing more trustworthy cloud environment.

Mabrouk et al. [26] presented a model that could determine QoS by focusing on service dynamic, for example, context awareness and user mobility. It is a semantic model, which has four ontology layers for a network resource, hardware resource, application service and end user. In this model, quality items such as response time and throughput, which determine service performance, are considered. A point to note is that this model provides only QoS knowledge representation as a general framework with specific QoS attributes represented in each ontology layer. Nevertheless, this model could provide another direction towards QoS service engineering. Additionally, performance evaluation and validation of the model is yet to be determined.

Gill et al. [27] presented QoS-based cloud resource management technique for ensuring reliable and cost-efficient cloud environment. This intelligent approach focuses on the preventions of the SLA violations whereby when there are sudden failures, self-healing function takes over to optimize resources. In this approach, availability and performance attributes are considered for violation evaluation as normal, good and critical. By managing these violations data, penalties could be calculated, thus accountability is being measured.

Kritikos and Plexousakis [28] discussed QoS importance based on both functional and non-functional aspects and mentioned that web services description models and semantic web technologies help in the selection of services and consider QoS attributes. The authors presented an analysis of requirements for a semantically rich and effective QoS-based web service selection. In the classification of the QoS attributes, response time and throughput are represented as domain independent performance attributes. Meanwhile, the service availability is classified under the dependability category because it is based on security and reliability of the service that can prevent a specific failure from happening. However, this attribute has nothing to do with the internal design and built. The paper presented a comprehensive classification of QoS attributes that can help in the management of service-oriented applications. This model also provides support to QoS brokers for multiple service-oriented applications such as SaaS that can map to the cloud-based services.

Cappiello et al. [29] presented a model for QoS monitoring and its adaptation. While presenting the QoS models, service quality categorization is presented and the first and foremost category is service performance. Service performance category attributes characterize how well a service performs. Response time and throughput are the most common among all service performance attributes under discussion. As response time is a composite attribute, it can be computed based on network delays and latency, where latency is based on queue delay time and execution time. This quality model shows the importance of performance and its attributes; response time and throughput as a fundamental parameter for QoS evaluation. Based on these attributes, QoS prediction can be performed.

Brandic et al. [30] focused on grid computing, the most relevant predecessor of cloud computing. The author presented an approach that focuses on the comprehensive set of QoS (performance, economic, legal and security) for high-

level workflow specification. For this purpose, a subset of BPEL and UML based QoS aware grid Workflow Language (QoWL) is presented. This approach is evaluated through experimentation and focused on the QoS, specifically performance only. Nevertheless, QoS requirements, negotiations and fulfilment through BPEL and UML do provide a better solution for service-oriented applications such as SaaS.

Truong et al. [31] focused on the importance of QoS monitoring and its role in the selection of service resources. This framework presents ways to monitor and analyze QoS metrics. Using a QoS Classification Tree, response time and throughput are placed in performance category and considered as most important in QoS evaluation. Subsequently, for our research, we emphasize these three important attributes, namely availability, response time and throughput. Availability is the ability of cloud service to provide desired customer services for an agreed period.

A point to note is that service availability is affected by simultaneous user load and during system maintenance. Response time is the total amount of time a system or an application takes to respond to a request for service. Response time is calculated for all aggregated SaaS services and maximum acceptable response time is agreed. In synchronous execution, a sum of response times of all single SaaS service is measured. Whereas for asynchronous execution, a maximum response time among all SaaS services is measured. Throughput is the ability of cloud service to process units of information in each amount of time. Throughput is calculated for services and the minimum acceptable throughput is agreed. Network capacity, especially from the service user side, also affects throughput. However, SaaS hosted infrastructure should have high throughput network.

3. Methodology and Proposed Model

This section provides an overview of the methodology applied and describes the proposed fuzzy model.

3.1. Fuzzy logic

SaaS applications QoS evaluation parameters (attributes) defined in SLAs could change over the duration of the service. QoS parameter values also vary according to service, its deployment configuration and agreed terms. To take runtime decision about the level of QoS based on the attribute is challenging due to different attributes values and their relationship or dependency on each other. Moreover, most research works for the binary classification are still unable to consider various environmental uncertainties [32]. Consequently, a fuzzy model such as the Adaptive Neural Fuzzy Inference System (ANFIS) and Fuzzy Least-Squares Regression (FLSR) is considered as a valid alternative among other available classifiers. However, fuzzy logic might not be the only solution for problems with uncertainty. There are many approaches and frameworks found in the literature that tackle the problem of classification in uncertainty. For example, Pernici and Siadat [32] proposed to use Bayesian networks in making decisions about non-functional properties while Kwiatkowska et al. [33] used a probabilistic method for classification in uncertainty. Additionally, fuzzy logic has been used in research on other domains such as domains of computing.

For example, intrusion detection, prediction and prevention system for web services using ANFIS [34]; medical, for example, using fuzzy models to classify

handgrip strength between normal person and a patient [35]; real estate, for example, using two non-linear fuzzy models, namely the ANFIS and FLSR model in prediction of real estate values [36]. Subsequently, fuzzy logic in dealing with QoS for SaaS is yet to be among the first being applied in our current research. The fuzzy logic being multi-valued logic deals with vague concepts. Deriving from fuzzy set theory, fuzzy logic deals with reasoning to determine approximate rather than precise values.

Thus, for fuzzy logic, the truth value of an item ranges from 1 to 0 and the resultant or decision is not 'true' or 'false' only. Fuzzy logic could deal with the real-world problem in imprecise terms as the human brain does and then respond with precise actions. A fuzzy set is the extension of the classical notion of a set, represents the elements with the degree of membership. For example, $(A, \mu A)$ represents a pair of fuzzy set with A as a set, $\mu A(x)$ as degree of membership for x, where $\mu A: A \rightarrow [0, 1]$ for all $x \in A$. Therefore, if $\mu A(x)$ is equal to 1, this means x is in $(A, \mu A)$, whereas if $\mu A(x)$ is equal to 0, then x is excluded from $(A, \mu A)$ [37]. In our study, raw response time and throughput, which are quantitative values, are then fuzzified to become fuzzy sets and measurable based on their membership functions.

3.2. An overview of our proposed method

The raw response time and throughput data captured through the simulation of a private cloud service are quantitative values, which do not represent any meaningful terms. However, through observation, it is seen that response time and throughput do exhibit an inverse relationship. This means response time is inversely related to throughput. In other words, the more the throughput, the slower is the response time. To establish a meaningful relationship between these two attributes, the quantitative data collected from simulation need to go through the process of fuzzification and defuzzification to obtain an accurate inference or decision to determine whether any cloud service violation based on these two attributes has occurred (refer to Section 2.3). Figure 1 provides an overview of the process and methodology used for determining QoS violation. First, a set of agreed attribute values (in our case, throughput and response time) for quality service between the CSCs and the CSPs are defined in SLAs (Fig. 1 Sequence #1). These attribute values shall then be constantly monitored in real time for further analysis to determine whether a violation of quality service has occurred (Fig. 1, Sequence #2). These real-time captured interactive attribute values are then matched with fuzzy if-then rules derived from our proposed fuzzy model to detect or predict whether violation to the QoS in terms of throughput and response time has occurred or not (Fig. 1, Sequences #3-#4).

3.3. Experimental setup and data simulation (response time and throughput)

An open source e-commerce SaaS is hosted in a private cloud using Windows Server 2012R2 and System Center 2012 Virtual Machine Manager [38] serving as IaaS. On this private cloud is where Windows virtual machine is hosted and Magneto [39], an open source e-commerce SaaS application is installed. Over on the e-commerce client website, simulation of real-time transactions is carried out and data (response time and throughput) are monitored and captured using an open source cloud monitoring

tool such as Apache Jmeter [40]. This tool tests and monitors the performance of static and dynamic resources with a variety of configurations catering consistent results and providing sufficient offline technical support.

Subsequently, through observation of these captured data and referring to Table 1, a high percentage (80%) of the transactions for response time and throughput are in the region from 0.5-3.5s (Columns 1-3, Row 5) and 16-28 kb (Columns 4-6, Row 4) respectively. These two ranges are then categorized as 'normal' range of values. Thus, for the rest of the ranges of values for response time are categorized as very long time (6.5-11.011s), long time (3.5-6.5s) and short time (0.0-0.5s). Similarly, for throughput, the range of values 28-71.428 kb is categorized as high range, 6-16 kb as low range and 0-6 kb as a very low range of throughput values.

A point to note is that for capturing the response time and throughput during simulation, the private cloud service and the SaaS application are constantly in an available state. No data is being captured when the cloud service and SaaS are not up and running.

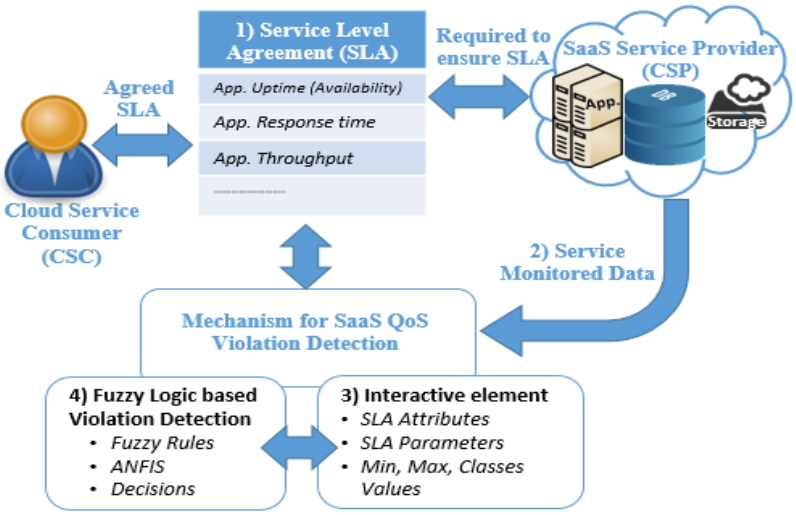

Fig. 1. An overview of the process for determining QoS violation.

Table 1. The range of values of response time and throughput.

Response Time			Throughput		
Range of Values (in seconds, s)		**Linguistic Terms**	**Range of Values (in kilobytes, kb)**		**Linguistic Terms**
6.5 - 11.011	(1%)	Very Long Time	28-71.428	(5%)	High
3.5 - 6.5	(14%)	Long Time	**16-28 kb**	**(80%)**	**Normal**
0.5-3.5	**(80%)**	**Normal**	6-16 kb	(14%)	Low
0.00-0.5	(5%)	Short Time	0-6 kb	(1%)	Very Low

3.4. Fuzzification and defuzzification

The process of transforming crisp values to fuzzy values is known as Fuzzification. Defuzzification, on the other hand, is the process of obtaining non-fuzzy values that best represent the possible distributions of inferred fuzzy decisions. Fuzzification

could be achieved through Membership Functions (MFs) which approximate the fuzziness in a fuzzy set, the elements in the set are either discrete or continuous. For our proposed fuzzy model, fuzzification of two inputs such as response time and throughput and defuzzification of one output such as cloud service violation decision carried out as shown in Fig. 2. Each fuzzified pair of inputs is mapped through a fuzzy inference process consisting of fuzzy if-then rules to derive the defuzzified decisions. The defuzzified crisp values for decision in linguistic terms are 'definitely no violation', 'normal', 'probably violation', and 'definitely violation'.

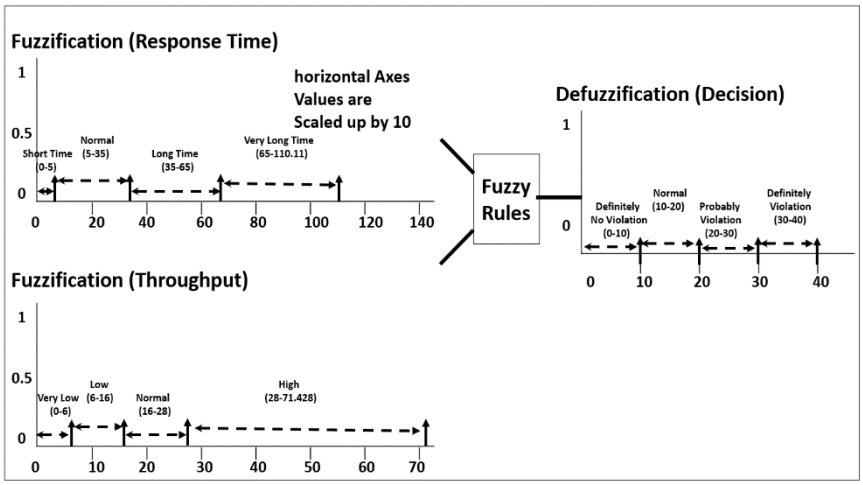

Fig. 2. Fuzzification of two inputs and defuzzification of one output.

Referring to Fig. 3, membership function for response time, through fuzzification, the values that fall under the 'short time' range shall exhibit no violation. Those under the normal range would be treated as agreed values under the SLAs. As for the 'long time' range of values, closer to the 'very long time' range of values, would be considered as probably causing a violation. This shall alert the administrator to take appropriate remedial action as needed, thus enabling prediction of definite violation, although the chance of it happening is only 1% (Table 1: Columns 1-3, Row 3).

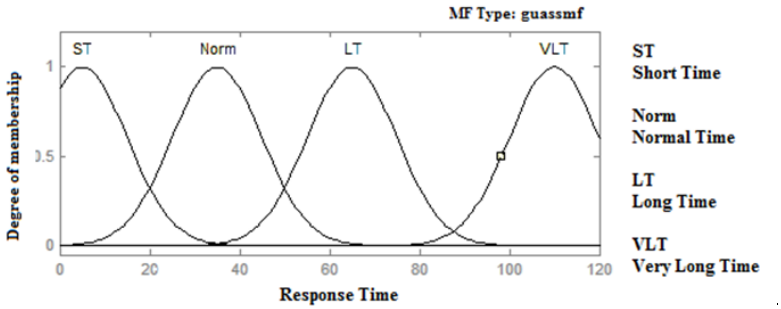

Fig. 3. Membership function for response time.

Similarly, for throughput, referring to Fig. 4, membership function for throughput, through fuzzification, the values that fall under the 'high' range shall exhibit no violation. Those under the normal range would be treated as agreed values under the SLAs. As for the 'low' range of values, closer to the 'very low' range of values, would be considered as probably causing a violation. Again, this shall alert the administrator to take appropriate remedial action as required. Thus, leading to the prediction of definite violation, although the chance for it to happen is just 1% only (Table 1: Columns 4-6, Row 6).

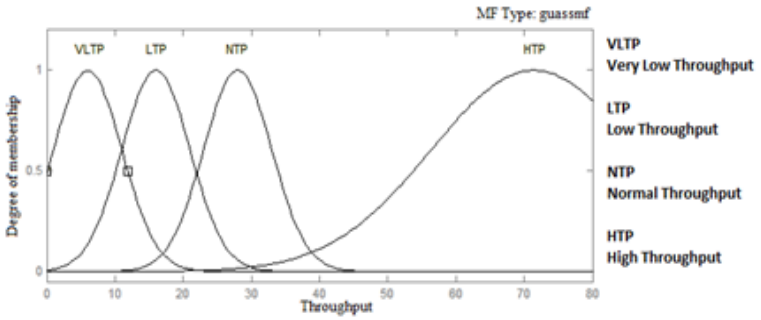

Fig. 4. Membership function for throughput.

The fuzzified pair of inputs then goes through a fuzzy inference process (refer to Section 2.5) consisting of fuzzy if-then rules for determining the defuzzified crisp values or linguistic violation decisions. The QoS violation decisions are categorized into four categories, namely, "definitely no violation" with a defuzzified output range of 0-10; "normal", with a defuzzified output range of 10-20; "probably violation", with a defuzzified output range of 20-30; and "definitely violation" with a defuzzified output range of 30-40 as shown in Fig. 2.

3.5. The proposed fuzzy model - ANFIS

ANFIS, an adaptive network, makes use of supervised learning resembling the Takagi-Sugeno fuzzy inference system. Figure 5 shows the Takagi-Sugeno model's fuzzy reasoning mechanism and Fig. 6 shows the ANFIS architecture.

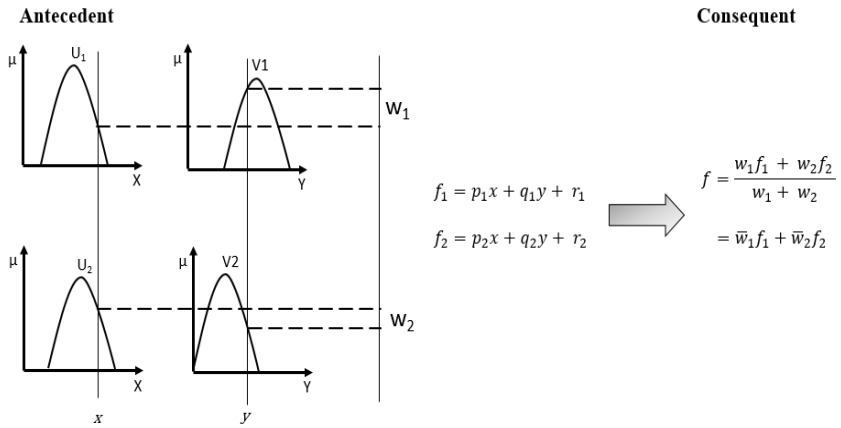

Fig. 5. Sugeno's fuzzy reasoning mechanism.

Referring to Fig. 5, the two inputs are *x* and *y* with one output *f*. For Takagi-Sugeno model, two "If-Then" rules are used as follows:

Rule 1 = if *x* is *U*1 and *y* is *V*1, then $f1 = p1x + q1y + r1$

Rule 2 = if *x* is *U*2 and *y* is *V*2, then $f2 = p2x + q2y + r2$

In Fig. 6, *A1*, *A2* are membership functions for *x* and *B1*, *B2* are the membership functions for *y*. *A1*, *A2*, *B1*, and *B2* are the antecedent (If) while *p1*, *q1*, *r1* and *p2*, *q2*, *r2* (referring to Rules 1-2) are linear parameters as resultant representing the consequent (then). The ANFIS architecture shown in Fig. 6 has five layers whereby the adaptive nodes are in the first and fourth layers, while three other layers (layer 2, layer 3 and layer 5) are fixed nodes [37, 41].

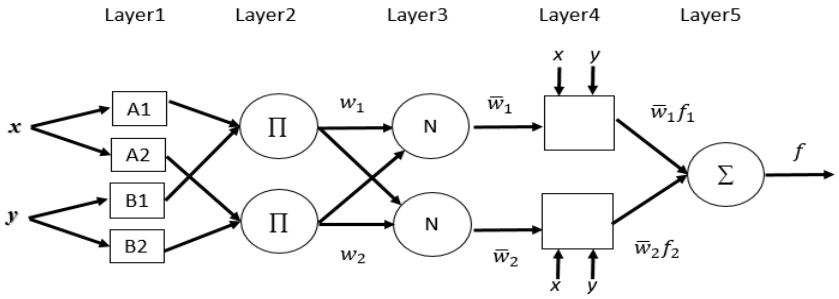

Fig. 6. ANFIS architecture.

For further explanation of our proposed ANFIS model, refer to Fig. 7. As in Fig. 7, Layer 1 is the fuzzification layer. Each node in Layer 1 is mapped to a function parameter. Each node's output shows a degree of membership value. Each output value is determined by inputs of membership functions. It transforms the crisp input *x*, in this study, response time, and y, the throughput, to membership functions represented by μ. It is a curve that defines how each point in the input space is mapped to a membership value i.e. degree of membership between 0 and 1, where *Ui* is the membership function of x, the response time and *Vi* is the membership function for *y*, the throughput and $i = \{1, 2, 3, 4\}$. Consequently, the four fuzzy sets associated with *x* are, short time *(U1)*, normal *(U2)*, long time *(U3)* and very long time *(U4)*. Similarly, the four fuzzy sets associated with *y* are very low *(V1)*, low *(V2)*, normal *(V3)* and high *(V4)*. The membership function can be a generalized bell membership function represented by Eq. (1), a Gaussian membership function represented by Eq. (2), or other types of membership functions such as triangular membership function, sigmoidal membership function and so on.

$$\mu_{Ui}(x) = \frac{1}{1+\left[\frac{x-c_i}{a_i}\right]^{2b}} \quad (1)$$

$$\mu_{Ui}(x) = exp\left[-\frac{x-c_i}{2a_i}\right]^2 \quad (2)$$

Note c_i and a_i are the center and width of the *ith* fuzzy set U_i. In our study, the Gaussian membership function is used. Thus, Eq. (3) is derived based on Eq. (2)

for the membership function of the second input y where a, and c are antecedent parameters with j= {1, 2, 3, 4}.

$$\mu_{Vi}(y) = exp\left[-\frac{y-c_j}{2a_j}\right]^2 \quad (3)$$

Each node of Layer 2 is non-adaptive or known as a fixed node. As shown in Fig. 6, the node represented as the circle is tagged as $\prod$ (the T-norm operator). The result for the output node could be obtained by multiplying signals coming to second layer node and going to a subsequent node. There are two inputs with each of the four fuzzy sets whereby the firing strength of every rule is represented by W_k where k = {1, 2, ...,16}. Subsequently, the T-norm operator with general performance, AND operation, is applied for both inputs to obtain the output for layer 3 according to Eq. (4).

$$w_k = \mu_{Ui}(x)\mu_{Vi}(y) \quad (4)$$

For our ANFIS, 16 pairs of x (response time) and y (throughput) outputs from Layer 2 with equal firing strength (weight for each rule is 1) are computed using Eq. (4). A point to note is that if input premises are partially true according to membership functions, then, based on Sugeno implication method, the output fuzzy set shall be truncated. In our case, since the weight applied to every rule is 1, then no effect is placed upon the implication process.

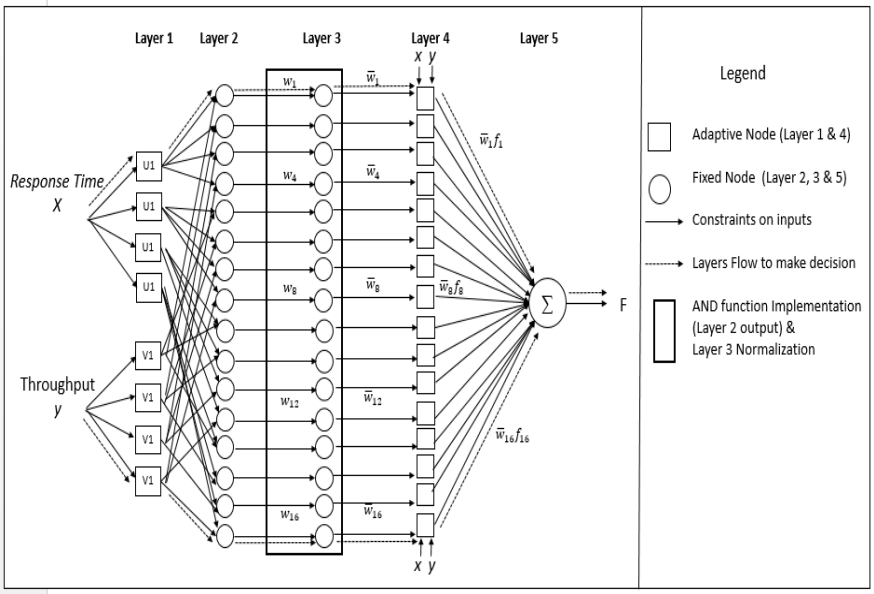

Fig. 7. The proposed ANFIS network structure for QoS violation decision.

Layer 3 represents the Normalized layer. Every node in Layer 3 is fixed or non-adaptive. It is symbolized as a circle node and labeled as N graphically. In Layer 3, each node is the result of computation of the ratio between the kth rule's firing strength and the sum of all rules' firing strengths, where k = {1, 2, 3, …, 16}. The result, which is the normalized firing strength, is computed using Eq. (5).

$$\bar{w}_k = \frac{w_k}{\sum_k w_k} \quad (5)$$

Layer 4, the defuzzification layer, has adaptive nodes that generate outputs based on the fuzzy if-then rule, Eq. (6)

$$\bar{w}_k f_k = \bar{w}_k \, (p_k x + \, q_k x + \, r_k) \tag{6}$$

where $\bar{w}_k$ is the previous layer normalized firing strength, k = {1, 2, 3, …,16} and $(p_k x + \, q_k x + \, r_k)$ are the consequent parameters in fourth layer node. Layer 5, the summation layer, has a single fixed non-adaptive output. This node provides an overall output of the ANFIS network. This overall output is the total weighted outputs of all if-then rules. While inputs are all 16 weighed output from the previous layer, the output is a single decision symbolized as a circle node that is labeled with the ∑ sign where computation is performed using Eq.(7) with k = {1, 2, 3, …,16}.

$$\sum_k \bar{w}_k f_k = \frac{\sum_k w_k f_k}{\sum_k w_k} \tag{7}$$

Based on these formulae and computation, 40 fine-tuned parameters whereby 16 are linear parameters (parameters *a*, and *c*: 2×8 =16) and *24* are nonlinear parameters (parameters *p*, *q*, and *r:* 3×8 = 24) are obtained. For further details on ANFIS with its respective formulae [42]. As can be seen from Fig. 7, our proposed network structure for QoS violation decision based on ANFIS, the raw inputs of response time *(x)* and throughput *(y)* shall go through five layers to derive a decision output. The 16 fuzzy rules thus derived from the fuzzy inference process are tabulated as shown in Table 2.

Table 2. The 16 fuzzy if-then rules.

Rule #		Response Time is		Throughput is		Decision
1		Short Time		High		Definitely No Violation
2		Short Time		Normal		Definitely No Violation
3		Normal		High		Definitely No Violation
4		Normal		Normal		Normal
5		Short Time		Low		Probably Violation
6		Normal		Low		Probably Violation
7		Long Time		High		Probably Violation
8		Long Time		Normal		Probably Violation
9	**IF**	Long Time	**AND**	Low	**THEN**	Probably Violation
10		Short Time		Very Low		Definitely Violation
11		Normal		Very Low		Definitely Violation
12		Long Time		Very Low		Definitely Violation
13		Very Long Time		High		Definitely Violation
14		Very Long Time		Normal		Definitely Violation
15		Very Long Time		Low		Definitely Violation
16		Very Long Time		Very Low		Definitely Violation

4. Performance Evaluation of Proposed ANFIS and Results Analysis

This section presents the experiment or testing performed to evaluate the performance of our proposed ANFIS model. Results obtained are analyzed for performance evaluation.

4.1. Test data and experimental results

Through simulation using the e-commerce SaaS, about 1500 instances of response time and throughput data are captured and used as test data for evaluation of the model's performance. The range of values captured for response time and

throughput are from 0.171 to 11.011 seconds (s) and from 0.433 to 71.428 kilobytes (kb) respectively. These data are then fed through the ANFIS model using MATLAB 2014a fuzzy logic toolbox.

The 1500 instances of response time and throughput pairs of data are randomly divided into five subsets of equal size with 300 instances; namely, *TQ1*, *TQ2*, *TQ3*, *TQ4* and *TQ5* (Table 3: Columns 1-2, Rows 2-6). For each data set *TQ3*, *TQ4*, and *TQ5*, it is further randomly divided into two subsets of 100 and 200 input-output pairs, namely, *TQ31*, *TQ32*, *TQ41*, *TQ42*, *TQ51* and *TQ52* (Table 3: Columns 3-4, Rows 4-9). The number of instances of these data sets divided in such a way so that the training to the testing ratio for *TQ1*: *TQ31* would be 3:1 and for *TQ1*: *TQ32* would be 3:2 and so on for the other training and testing pairs of data sets.

In our experiments, for the training data sets, *TQ1* and *TQ2*, the following parameters are used: the hybrid optimization method, grid partition with guassmf and linear as input and output membership type respectively. Epoch 40 and error tolerance is zero. By trying different parameters combination, it is found that this set of parameters provide the best combination for optimum results with low measures of RMSE..

Table 3. Training and testing data sets.

Dataset	No. of Instances	Sub Datasets	No. of Instances
TQ1	300	-	-
TQ2	300	-	-
TQ3	300	*TQ31*	100
		TQ32	200
TQ4	300	*TQ41*	100
		TQ42	200
TQ5	300	*TQ51*	100
		TQ52	200

Table 4 shows the training errors for datasets *TQ1* and *TQ2* with RMSE 0.3811 and 0.4105 respectively with an average training error (RMSE) of less than 0.4. Table 5 shows the results of testing with all the same size (300 instances) data sets, i.e. *TQ1*-*TQ5*. The range of testing errors (RMSE) is from 0.4127 to 0.6432 with an average testing error (RMSE) of less than 0.5.

Table 4. Training errors for data sets with 300 instances.

Item No.	Data Sets	Training Error (RMSE)
1	***TQ1***	**0.3811**
2	*TQ2*	0.4105
Average Training Error (RMSE)		**0.3958**

Table 5. Testing errors with data sets with 300 instances.

Item No.	Data Sets	Testing Error (RMSE)
1	*TQ1 : TQ2*	0.4560
2	*TQ1 : TQ3*	0.4576
3	*TQ1 : TQ4*	0.5207
4	*TQ1 : TQ5*	0.4127
5	*TQ2 : TQ1*	0.4835
6	*TQ2 : TQ3*	0.6432
7	*TQ2 : TQ4*	0.4887
8	*TQ2 : TQ5*	0.5224
Average Testing Error (RMSE)		**0.4981**

Table 6 shows training and testing errors for data sets, which are subsets of *TQ3*, *TQ4*, and *TQ5* with data size in 3:1 and 3:2 ratios. The testing error in RMSE ranges from 0.3684 to 0.6698 with an average of 0.4887. These average training and testing errors with RMSE of less than 0.5 (refer to Tables 4, 5 and 6) suggest that the proposed fuzzy model exhibits detection and prediction accuracy of greater than 99% for QoS violation, probably violation or no violation in terms of response time and throughput by using the 16 fuzzy rules. Thus, allowing the administrator to make an accurate decision as to when to take remedial action as most appropriate.

Table 6. Testing errors for data sets with instances in 3:1 and 3:2 ratios.

Item No.	Data Sets	Testing Error (RMSE)
1	*TQ1 : TQ31*	0.4351
2	*TQ2 : TQ31*	0.5863
3	*TQ2 : TQ41*	0.4350
4	*TQ1 : TQ51*	0.3684
5	*TQ1 : TQ32*	0.4684
6	*TQ2 : TQ32*	0.6698
7	*TQ2 : TQ42*	0.5134
8	*TQ1 : TQ52*	0.4332
Average Testing Error (RMSE)		**0.4887**

4.2. Performance comparison of our ANFIS with other classifiers

To further confirming our experimental results, the test dataset, i.e. *TQ1* dataset, having the lowest RMSE, is used in open source machine learning tool called WEKA [42] for experimentation with other non-fuzzy classifiers such as linear regression, decision Table, and zero R to obtain error measures (RMSEs) for comparison with our ANFIS model. A comparison of RMSEs of these classifiers and ANFIS are tabulated as shown in Table 7, our ANFIS model used in detecting and predicting the QoS violation obtains the lowest error measure with RMSE of less than 0.4 as compared to the other three non-fuzzy classifiers with a range of RMSEs from 0.54 to 0.77.

Table 7. Performance of ANFIS with other classifiers.

Classifiers	Linear Regression	Decision Table	Zero R	ANFIS
RMSE	0.5429	0.5699	0.7725	**0.3811**

5. Conclusion and Future Work

Cloud computing is a new paradigm towards computing with flexible use due to its flexibility in providing resources and reduction in cost factors. Assurance towards providing QoS agreed in SLA through monitoring cloud resources is, therefore, a new challenge. Hence, service performance that includes the availability of the service, service response time and size of desired data transferred from the service is one of the core concerns of the cloud users. This study proposed a fuzzy-based model for QoS violation detection and prediction of service response time and duration to ensure continuous service availability with agreed quality. Our experimental results based on ANFIS show that the proposed fuzzy model achieves detection and prediction accuracy of greater than 99% with an average small RMSE

of just less than 0.50. Hence, using our fuzzy model with 16 fuzzy if-then rules of detecting and predicting QoS violation for SaaS surely allows the administrator to make the accurate decision for appropriate remedial action. Consequently, accountability is taken care of with win-win benefits for both the cloud users and cloud service providers. Our proposed fuzzy model, thus, contributes towards a different dimension in the measurement of quality of cloud services where accountability is taken into consideration.

Our future work aims to extend the violation decisions obtained from this fuzzy model to derive an efficient mechanism for remedial action to resolve SaaS's QoS violations.

Acknowledgment

This research work is supported by the Fundamental Research Grant Scheme provided by the Ministry of Higher Education (MOHE), Malaysia with grant number FRGS/1/2016/ICT01/MMU/ 02/1.

Nomenclatures

A	fuzzy set
a_i	width of the i^{th} fuzzy set
c_i	center of the i^{th} fuzzy set
$\overline{w}_k$	previous layer normalized firing strength, k= {1, 2, 3, …,16}
W_k	firing strength, where k = {1, 2, ... ,16}
x, y	inputs for Takagi-Sugeno model
$\mu A(x)$	degree of membership for x

Greek Symbols

$\mu_{Ui}(x)$	Membership function, fuzzy sets associated with x are, short time ($U1$), normal ($U2$), long time ($U3$) and very long time ($U4$)
$\mu_{Vi}(y)$	Membership function, fuzzy sets associated with y are very low (V_1), low (V_2), normal (V_3) and high (V_4)

Abbreviations

ANFIS	Adaptive Neural Fuzzy Inference System
AWS	Amazon Web Service
AHP	Analytic Hierarchy Process
BPEL	Business Process Execution Language
CSC	Cloud Service Consumer
CSP	Cloud Service Provider
FLSR	Fuzzy Least-Squares Regression
IaaS	Infrastructure as a Service
QoS	Quality of Service
PaaS	Platform as a Service
RMSE	Root Mean Square Error
SaaS	Software as a Service
SLA	Service Level Agreement
SLO	Service-Level Objectives
VMs	Virtual Machines

References

1. Mell, P.; and Grance, T. (2011). The NIST definition of cloud computing. Recommendations of the national institute of standards and technology. *NIST Special Publication 800-145, Computer Security Division, Information Technology Laboratory*. NIST Gaithersburg, MD 20899-8930. doi: 10.1136/emj.2010.096966.
2. Sullivan, O.J.; Edmond, D.; Ter Hofstede; A., Benatallah, B.; and Casati, F. (2002). What's in a service? Towards accurate description of non-functional service properties. *Distributed and Parallel Databases*, 12, 117-133.
3. Shin, S.; and Gu, G. (2012). CloudWatcher: network security monitoring using OpenFlow in dynamic cloud networks (or: how to provide security monitoring as a service in clouds?). *20th IEEE International Conference on Network Protocols (ICNP), 2012*, 1-6.
4. Alhamazani, K.; Ranjan, R.; Mitra, K.; Rabhi, P.; Jayaraman, P.; Khan, S.U.; Guabtni, A.; and Bhatnagar V. (2015). An overview of the commercial cloud monitoring tools: research dimensions, design issues, and state-of-the-art. *Computing*, 97(4), 357-377.
5. Ko, R.K.L.; Jagadpramane, P.; Mowbray, M.; Perrson, S.; Kirchberg, M.; Liang, Q.; and Lee, B.S. (2011). TrustCloud: A framework for accountability and trust in cloud computing. *2011 IEEE World Congress on Services*, 584-588.
6. Theoharidou, M.; Papanikolaou, N.; Pearson, S.; and Gritzalis, D. (2013) Privacy risk, security, accountability in the cloud. *2013 IEEE 5th International Conference on Cloud Computing Technology and Science*, 177-184.
7. Aceto, G.; Botta, A.; De Donato, W.; and Pescape, A. (2013). Cloud monitoring: a survey. *Computer Networks*. Elsevier B.V., 2093-2115.
8. Qu, C.; Calheiros, R.N.; and Buyya, R. (2016). A reliable and cost-efficient auto-scaling system for web applications using heterogeneous spot instances. *Journal of Network and Computer Applications*, 65, 167-180.
9. Zheng, Z.; Wu, X.; Zhang, Y.; Lyu, M.R.; and Wang, J. (2013). QoS ranking prediction for cloud services. *IEEE Transactions on Parallel and Distributed Systems*, 24(6), 1213-1222.
10. Chen, T.; and Bahsoon, R. (2013). Self-adaptive and sensitivity-aware QoS modelling for the cloud. *ICSE Workshop on Software Engineering for Adaptive and Self-Managing Systems*, 43-52.
11. Grati, R.; Boukadi, K.; and Ben-Abdallah, H. (2012). A QoS monitoring framework for composite web services in the cloud. *The Sixth International Conference on Advanced Engineering Computing and Applications in Sciences (ADVCOMP 2012)*, (c), 65-70.
12. Qazi, F.; Jhumka, A.; and Ezhilchelvan, P. (2017). Towards automated enforcement of cloud SLA. *UCC Companion'17, December 5-8, 2017, Austin, Texas, USA*, 151-156.
13. Michlmayr, A.; Rosenberg, F.; Leitner, P.; and Dustdar, S. (2009) Comprehensive QoS monitoring of web services and event-based SlA violation detection. *Proceedings of the 4th international workshop on middleware for service-oriented computing*, 1-6.

14. Kumbhare, A.G.; Simmhan, Y.; Frincu, M.; and Prasanna, V.K. (2015) Reactive resource provisioning heuristics for dynamic data flows on cloud infrastructure. *IEEE Transactions on Cloud Computing*, 3(2). 105-118. doi: 10.1109/TCC.2015.2394316.
15. Chong, C.Y.; Lee, S.P.; and Ling, T.C. (2014). Prioritizing and fulfilling quality attributes for virtual lab development through application of fuzzy analytic hierarchy process and software development guidelines. *Malaysian Journal of Computer Science*, 27(1), 1-19.
16. Fatema, K.; Emeakaroha, V.C.; Healy, P.D.; Morrison, J.P.; and Lynn, T. (2014). A survey of cloud monitoring tools: Taxonomy, capabilities and objectives. *Journal of Parallel and Distributed Computing*, 2918-2933.
17. Barth, W. (2008). *Nagios: System and network monitoring*. (2nd Ed.). *Time*. No Starch Press.
18. CloudWatch (2017). Amazon cloud watch developer guide. Retrieved December 7, 2017, from http://awsdocs.s3.amazonaws.com/ AmazonCloud Watch/latest/ acw-dg.pdf.
19. Azure FC (2017). Retrieved December 7, 2017, from http://snarfed.org/ windows_ azure_details#Configuration_and_APIs.
20. Meddeb, A. (2010). Internet QoS: pieces of the puzzle. *IEEE Communications Magazine*, 48(1), 86-94.
21. Monitis (2017). Retrieved October 10, 2017, from URL https://www.monitis. com.
22. Hoßbach, B.; Freisleben, B.; and Seeger, B. (2012). Reaktives Cloud Monitoring MIT Complex Event Processing. *Datenbank-Spektrum*, 12(1), 33-42.
23. ISO (2001). ISO/IEC 9126-1: Software engineering-product quality-part 2: quality model. *International Organization for Standardization*, Geneva, Switzerland.
24. Kritikos, K.; Carro, M.; Pernici, B.; Plebani, P.; Cappiello, C.; Comuzzi, M.; Benrernou, S.; Brandic, I.; Kertesz, A.; and Parkin, M. (2013). A survey on service quality description. *ACM Computing Surveys*. ACM, 46(1), 1-58.
25. Khan, H. M.; Chan, G. Y.; and Chua, F. F. (2016). An adaptive monitoring framework for ensuring accountability and quality of services in cloud computing. *International Conference on Information Networking*. IEEE, 249-253.
26. Mabrouk, N.B.; Georgantas, N.; and Issarny, V. (2009). A semantic end-to-end QoS model for dynamic service oriented environments. *Proceedings of the 2009 ICSE Workshop on Principles of Engineering Service Oriented Systems*. IEEE, 34-41.
27. Gill, S.S.; Chana, I.; Singh, M.; and Buyya, R. (2017). CHOPPER: an intelligent QoS-aware autonomic resource management approach for cloud computing. *Cluster Computing*. Springer US, 1-39.
28. Kritikos, K.; and Plexousakis, D. (2009). Requirements for QoS-based web service description and discovery. *IEEE Transactions on Services Computing*, 320-337.
29. Cappiello, C.; Kritikos, K.; Metzger, A.; Parkin, M.; Pernici, B.; Plebani, P.; and Treiber, M. (2008). A Quality model for service monitoring and adaptation. *Workshop on Service Monitoring, Adaptation and Beyond*, 29-42.
30. Brandic, I.; Pllana, S.; and Benkner, S. (2006). An approach for the high-level specification of QoS-aware grid workflows considering location affinity. *Scientific Programming*. Hindawi Publishing Corporation, 14(3-4), 231-250.

31. Truong, H. L.; Samborski, R.; and Fahringer, T. (2006). Towards a framework for monitoring and analyzing QoS metrics of grid services. *E-Science 2006 - Second IEEE International Conference on e-Science and Grid Computing*. IEEE, 65-65.
32. Pernici, B.; and Siadat, S.H. (2011). A fuzzy service adaptation based on QoS satisfaction. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Springer Berlin Heidelberg, 48-61.
33. Kwiatkowska, M.; Norman, G.; and Parker, D. (2004). Probabilistic symbolic model checking with PRISM: A hybrid approach. *International Journal on Software Tools for Technology Transfer*, 6(2), 128-142.
34. Chan, G. Y.; Lee, C.S.; and Heng, S.H. (2012). Policy-enhanced ANFIS model to counter SOAP-related attacks. *Knowledge-Based Systems*. Elsevier B.V., 35, 64-76.
35. Woo, C.; and Chitsaz, M. (2010). Handgrip strength evaluation using neuro fuzzy approach. *Malaysian Journal of Computer Science*, 23(3), 166-176.
36. Sarip, A.G.; Hafez, M.B.; and Nasir D.M. (2016). Application of fuzzy regression model for real estate price prediction. *Malaysian Journal of Computer Science*, 29(1), 15-27.
37. Viattchenin, D.A.; Tati, R.; and Damaratski, A. (2013). Designing gaussian membership functions for fuzzy classifier generated by heuristic possibilistic clustering. *Journal of Information and Organizational Sciences*, 37(2), 127-139.
38. Virtual Machine Manager (2017). Retrieved September 5, 2017, from https://technet. microsoft.com/en-us/library/gg610610(v=sc.12).aspx
39. Magento, Retrieved October 7, 2017, from https://magento.com/resources/ technical
40. Apache Jmeter (2017). Retrieved October 7, 2017, from http://jmeter.apache. org
41. Suparta, W.; and Alhasa, K.M. (2016). Adaptive Neuro-Fuzzy interference system. In *modelling of tropospheric delays using ANFIS*. Springer Briefs in Meteorology, 5-18.
42. WEKA. Workbench practical machine learning tool: weka-3-8-1jre-x64.exe, Retrieved November 5, 2017, from http://www.cs.waikato.ac.nz/ ml/weka/ downloading.html