Computers, Materials & Continua
DOI:10.32604/cmc.2022.020852
Article

Tech Science Press

Adaptive Runtime Monitoring of Service Level Agreement Violations in Cloud Computing

Sami Ullah Khan[1], Babar Nazir[1], Muhammad Hanif[2,*], Akhtar Ali[3], Sardar Alam[1] and Usman Habib[4]

[1]Department of Computer Science, COMSATS University, Abbottabad Campus, Pakistan
[2]Faculty of Computer Science and Engineering, Ghulam Ishaq Khan (GIK) Institute of Engineering Sciences and Technology Topi, Pakistan
[3]IFahja Pvt Limited, Peshawar, Pakistan
[4]Department of Computer Science, National University of Computer and Emerging Sciences Islamabad, Chiniot-Faisalabad Campus, Pakistan
*Corresponding Author: Muhammad Hanif. Email: muhammad.hanif@giki.edu.pk
Received: 28 May 2021; Accepted: 17 August 2021

Abstract: The cloud service level agreement (SLA) manage the relationship between service providers and consumers in cloud computing. SLA is an integral and critical part of modern era IT vendors and communication contracts. Due to low cost and flexibility more and more consumers delegate their tasks to cloud providers, the SLA emerges as a key aspect between the consumers and providers. Continuous monitoring of Quality of Service (QoS) attributes is required to implement SLAs because of the complex nature of cloud communication. Many other factors, such as user reliability, satisfaction, and penalty on violations are also taken into account. Currently, there is no such policy of cloud SLA monitoring to minimize SLA violations. In this work, we have proposed a cloud SLA monitoring policy by dividing a monitoring session into two parts, for critical and non-critical parameters. The critical and non-critical parameters will be decided on the interest of the consumer during SLA negotiation. This will help to shape a new comprehensive SLA based Proactive Resource Allocation Approach (RPAA) which will monitor SLA at runtime, analyze the SLA parameters and try to find the possibility of SLA violations. We also have implemented an adaptive system for allocating cloud IT resources based on SLA violations and detection. We have defined two main components of SLA-PRAA i.e., (a) Handler and (b) Accounting and Billing Manager. We have also described the function of both components through algorithms. The experimental results validate the performance of our proposed method in comparison with state-of-the-art cloud SLA policies.

Keywords: Energy consumption; penalty calculation; proactive resource allocation; service level agreement (SLA) monitoring; SLA violation detection

1 Introduction

In recent years, cloud computing revolutionized information technology and introduce new protocols for effective communication. Cloud computing is a new computing paradigm in which customers can use highly scalable IT resources or services according to their needs in a basic pay-per-use [1]. The provision of cloud services, such as platforms, infrastructures, (processing capacity, network bandwidth, and storage) and applications have been carrying out by cloud service providers based on a predefined set of roles called quality of service properties (QoS) [2,3]. The negotiation of QoS is made by an agreement called the service level agreement (SLA) [4]. SLA is considered a legal contract for cloud service providers and customers to ensure QoS for cloud services. In case of breach of service quality, the defaulter has to pay a fine as specified in the SLA [5]. SLA plays an important role in defining the expectations of parties, by describing the nature and type of services [6,7]. Mostly web services agreements are used for the specification of main factors and structure of SLA [8].

The reliable monitoring and management of SLA are also important for cloud consumers and providers, as this is the dominant mechanism for maintaining strong business between the two parties. SLA monitoring can also help to ensure QoS applications in the cloud. The SLA monitoring system effectively detects SLA objectives and helps to provide resources and services in an adaptive way [9,10]. The recent proposals, adoptions are very much influenced to detect the SLA violations either at runtime through early analysis [11] or at a testing time by pre-fix analysis [12,13]. There is a problem with the above-mentioned methods because in early analysis too many messages will be sent due to which network traffic will increase and pre-fix time analysis violations will be over headed due to fix analysis time. Such parameters are ignored in most literature. Some machine learning based run-time SLA monitering methods are suggested in literature to somehow overcome or reduce the effects of the above mentioned shortcomings. In [14] the authors presents a comparision of time series (TS) and machine learning (ML) -based methods for SLA violation predictions. They concluded that ML based methods out-performed the TS based methods in predicting the service quality. A neighborhood-based approach is suggested in [15] for quality of service estimation. Comparetively, their proposed method produce optimal results. A run-time SLA violation detection method using linear regression technique is presented in [16]. They used check-points for execution of composite service using recorded historical data for the prediction regression models. In [17], the authors compared two ML models, namely Naïve Bayes and Random Forest Classifier for SLA violation prediction. They concluded that Randome Forest Classifier can efficiently predict SLA violations.

To the best of our knowledge, there are no such models that achieve the above problem with both methods as well most of the methods in the literature are either completely coupled with SLA, or use a single component for monitoring and analysis, while others only detect violation without any explanation, and some of them provide very basic information about the violation. The main contribution of our work is highlighted below:

- We have design architecture for the prediction of SLA violations at runtime and provide reserved resources proactively in case of any SLA violation prediction.
- Design and implemention of SLA based Proactive Resource Allocation Approach (SLA-PRAA). This will help in legitimate user identification according to the services and will monitor the authentication of service level agreement on both sides.
- The proposed monitoring scheme is completely decoupled from SLA, which means no changes are required to monitor if SLA specification is changed. Which supports the independent core component of SLA monitoring and analysis.

- SLA-PRAA does early analysis for critical SLA metrics, due to which SLA violations are detected early and pre-fix time analysis for non-critical SLA metrics, which reduce network traffic. SLA-PRAA also calculates penalties after any violation.
- We proposed an adaptive resource allocation system to allocate IT resources for cloud applications and efforts to reduce the occurrence of SLA violations by allocating additional resources to detect the likelihood of a violation.
- If an SLA violation is detected using threshold value then SLA-PRAA provides a resource proactively to the requester. Otherwise, if a violation occurs due to insufficient resources then SLA-PRAA will handle it and calculate the penalty automatically.

The rest of the paper is organized as follows: Section 2 describes related work. Tthe problem statement is presented in Section 3. The proposed solution is explained in Section 4. Similarly, in Section 5 we have discussed the simulations and results, and finally, Section 6 concludes the paper.

2 Related Work

In the current literature, we have focused on six features related to services and monitoring policy. A brief description is presented below.

2.1 Monitor Configuration

In some models monitor is configured automatically from SLA [18–20]. However, in this mechanism, the monitor works well only with concrete SLA specification, but incase of any changes in the SLA specification then the monitor modification is necessary according to the coupling. Generally, SLA is decoupled from monitor. In some models SLA is decoupled from the monitoring component by translating SLA to another document that contains the required monitoring information [3,21,22].

2.2 Monitoring Policies

In most cases, pre-fix analysis is used for SLA monitoring [10,11]. In contrast, the SALMonADA [9] used an early analysis policy for SLA monitoring but they have issue in terms of energy efficiency. In [23] the author has used the monitoring policy as a time based. They have separate monitoring policies for up and downtime.

2.3 Monitoring Result Response

In several articles, the authors do not mention that how to make the report of monitoring result [3,20]. In some cases, the monitoring reports are presented in a log file [24] or in the form of Application Programming Interface (API) [10,22,23]. There is no standardized method because APIs and log files are two completely different representations. Due to the absence of a standard form, we cannot change monitor easily, and a problem of analyzer compatibility with monitor logs and APIs raised. The beneficial solution is that author uses a query language [25] for the reporting of monitoring results. But according to this solution monitor must deal with that query language.

2.4 The SLA Violations

Based on related work, the SLA violation can be categorise as given below:

- *Detect only:* Detection of SLA violation without the reason for violation [3,19,22,24].
- *Incomplete explanation:* in some proposals, i.e., [21], the reason for violation is incomplete.
- *Hard for human undertanding:* In some proposals, i.e., [20] the authors have provided the reason for violating the terms. But Event-Calculus is used to express monitoring results and conditions, which are not easily nderstandable by human.
- *Easy and detailed*: a correct explanation and easy to understand reasons for violation is presented in some proposals [7,8].

Tab. 1 presents a summary of related work. We can say that there are two types of components, functional and architectural. In the functional part, few proposals cover several or none of the five issues which were defined earlier. In our proposed solution model, all these features are considered, either cover the issue as it or make some improvement in it with new strategy. A detailed description is given in the next section.

Table 1: Analysis of related work

Techniques	Monitoring Policy			Monitoring Results				Explanation of Violations				Limitation
	Pre–Fix Analysis	Early analysis	Time-based analysis (up and down time)	Through API	Through log file	Through formal document	Through query language	No explanation (only detect violations)	H-U. violation Not	A partial explanation of violation H-U.	Detection and explanation of violation H-U.	
Keller et al. [18]	✓	✗✗	✗	✓	✗	✗	✗	✗	✓		✗	Partial explanation, No accounting, No violation handling
Kotsokalis et al. [22]	✓	✗	✗	✓	✗	✗	✗	✗	✗		✗	No explanation, No accounting, No violation handling
AMichlmayr et al. [19]	✓	✗	✗	✗	✗	✗	✓	✓		✗		✗ No explanation, No accounting, No violation handling

(Continued)

Table 1: Continued

Techniques	Monitoring Policy			Monitoring Results				Explanation of Violations				Limitation
	Pre–Fix Analysis	Early analysis	Time-based analysis (up and down time)	Through API	Through log file	Through formal document	Through query language	No explanation (only detect violations)	No explanation H-U.	A partial explanation of violation Not H-U.	Detection and explanation of violation H-U.	
Sahai et al. [21]	✓	✗	✗	✗	✓	✗	✗	✓		✗	✗	Partial explanation, not H-U, No accounting, No violation handling
Mahbub et al. [20]	✓	✗	✗	✗	✗	✗	✗		✗	✓	✗	Detail explanation not H-U, No accounting, No violation handling
Spanoudakis et al. [21]	✓	✗	✗	✓	✗	✗	✗	✓		✗	✗	No explanation, No accounting, No violation handling
Emeakaroha et al. [3]	✓	✗	✗	✗	✗	✗	✗	✓		✗	✗	No explanation, No accounting, No violation handling
Müller, Oriol et al. [7]	✓	✗	✗	✗	✗	✓	✗	✗		✗	✓	Detail explanation H-U, No accounting, No violation handling
Muller et al. [8]	✓	✗	✗	✗	✗	✓	✗	✗		✗	✓	Detail explanation H-U, No accounting, No violation handling

(Continued)

Table 1: Continued

Techniques	Monitoring Policy				Monitoring Results				Explanation of Violations				Limitation
	Pre–Fix Analysis	Early analysis	Time-based analysis (up and down time)	Through API	Through log file	Through formal document	Through query language	No explanation (only detect violations)	H-U. No explanation Not violation	A partial explanation of violation H-U.	Detection and explanation of violation H-U.		
Emeakaroha [25]	✗	✗	✓	✗	✗	✓	✗	✗	✗	✓			Detail explanation H-U, No accounting, No violation handling
Müller et al. [9,26]			✗	✗	✗	✓	✗	✗	✗	✓			Detail explanation H-U, No accounting, No violation handling

3 Problem Formulation

The consumers and providers use techniques such as runtime monitoring to avoid the penalties of SLA violations. Traditional systems have the following drawbacks and require to be rectified:

- Let in Eq. (1), SLA_{v1}, SLA_{v2}, $SLA_{v3}, \ldots, SLA_{vn}$ are the number of total SLA violations and P_{TA} is prefixed time analysis then

$$\sum_{i=1}^{n} SLA_{vn} \propto P_{TA} \tag{1}$$

The Eq. (1) shows that the total number of SLA violations is directly proportional to prefix time analysis. Where prefix analysis is the pre-specified time after which the monitor start monitoring of non-critical parameter and send the monitoring result for analysis. While critical and non-critical parameters will decide both SLA consumer and provider in SLA agreement.

- Similarly, Let in Eq. (2) P_{TA} is a prefixed time analysis technique and T_{NE} represent *not-efficient time* to analyze SLA violations then

$$T_{NE} \propto P_{TA} \tag{2}$$

Eq. (2) show that prefix time analysis of SLA violation is not efficient due to the occurrence of mix violation in maximum time.

- Let AN_E represent *early analysis notification* and T_{IN} represent *increase in network traffic* then

$$AN_E \propto T_{IN} \tag{3}$$

Eq. (3) shows that early analysis of SLA violation is directly proportional to the increase in network traffic. Where early analysis is the monitoring and analysis policy in which the monitor will do monitoring on regular basis and send a notification to the analyzer for analysis. Due to network traffic, energy consumption is increased. In our observation, the current literature have the following short comings:

- No such policy of SLA monitoring to minimize the SLA violations
- No corrective method for the detected violations.
- No account or metering system of miss-detection.

The above mentioned problems are diagrammatically shown in Fig. 1 i.e., increase SLAs violation due to prefix time analysis, increase network traffic due to early analysis notification, no corrective methods for detecting violations, no account or metering system of miss-detection. In Fig. 1, when the SLA is agreed upon, the mapping rules are then created by the service provider. The customer requests for provisioning of an agreed service, once it was provisioned, the run-time monitor from the agreed SLA repository load the services. Provisioning of services is based on infrastructure resources that represents the network resources and host in a data center for hosting cloud services. The metrics measure the resources by the metric agent's, accessible by the host monitor. The pairs of visitor control values extract raw metric value indicators and periodically transmit the performance monitor and knowledge component. Upon receipt of measured steps, the execution monitor is assigned a low-level metric based on predefined target allocation rules allowed by SLA. The resulting allocation is stored in the affected pool mapping repository, which also contains predefined allocation rules. The performance monitor uses the assigned values to monitor the implementation of state services. In case of future threats of SLA violation, the parties will be notified by the knowledge component.

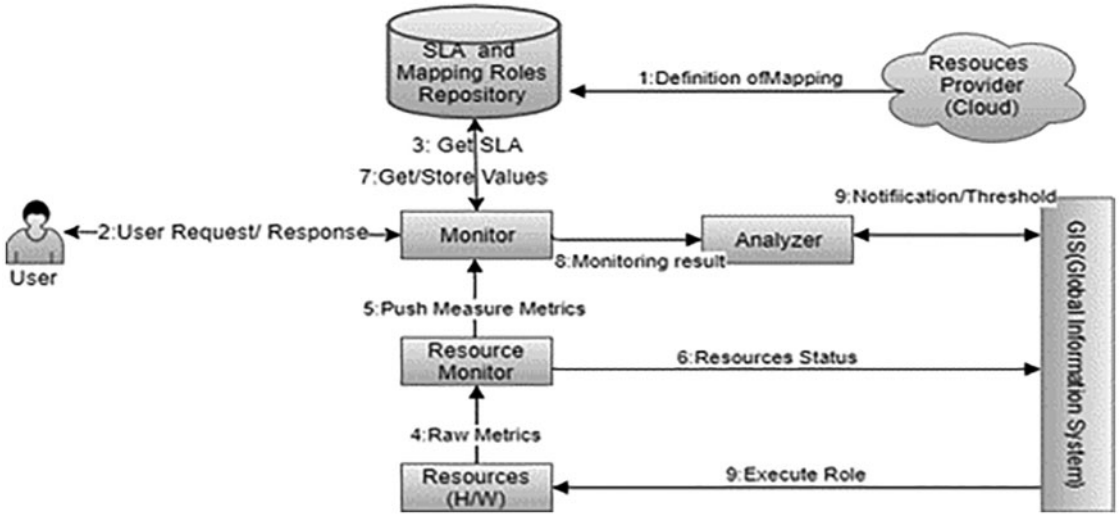

Figure 1: Monitoring and analysis of SLA violation in a cloud environment

This article intends to propose an efficient model for service level agreement monitoring, analysis over the cloud environment, which is not only suitable for cloud consumers but also very important for a cloud provider with hard and soft deadlines of service level agreement in cloud computing.

4 The Proposed Model (SLA-Proactive Resource Allocation Approach (PRAA))

In this section, we have described the SLA-PRAA platform in detail. The main component of SLA-PRAA is a controller, which plays the role of the external interface and controls the flow of internal execution of the system keeping decoupled the monitoring and analysis components from each other. Furthermore, the SLA-PRAA provides the monitoring document manager (MDM) service, which is used for generating independent monitoring document (MD) from the documents underlying structure or we say it is generated from SLA which is constructed between user and provider. This platform is capable to monitor, analyze, and handle expressive SLA violation detection by allocating resources proactively.

The SLA-PRAA has a decouple architecture which is shown in Fig. 2. The detection of SLA violation is fully dependent on the selection of the right threshold. In our approach, SLA threshold targets are defined based on experiment and historical information with types of explicit applications in terms of resource consumption [25,27]. The source model elements is mapped into a target domain that has formally defined semantics. Our target domain is constraint satisfaction problem [28] which is used and explain below.

In the following subsection we describe the components of SLA-PRAA in detail. We focus on the internal function of the main components of SLA-PRAA architecture and its responsibilities.

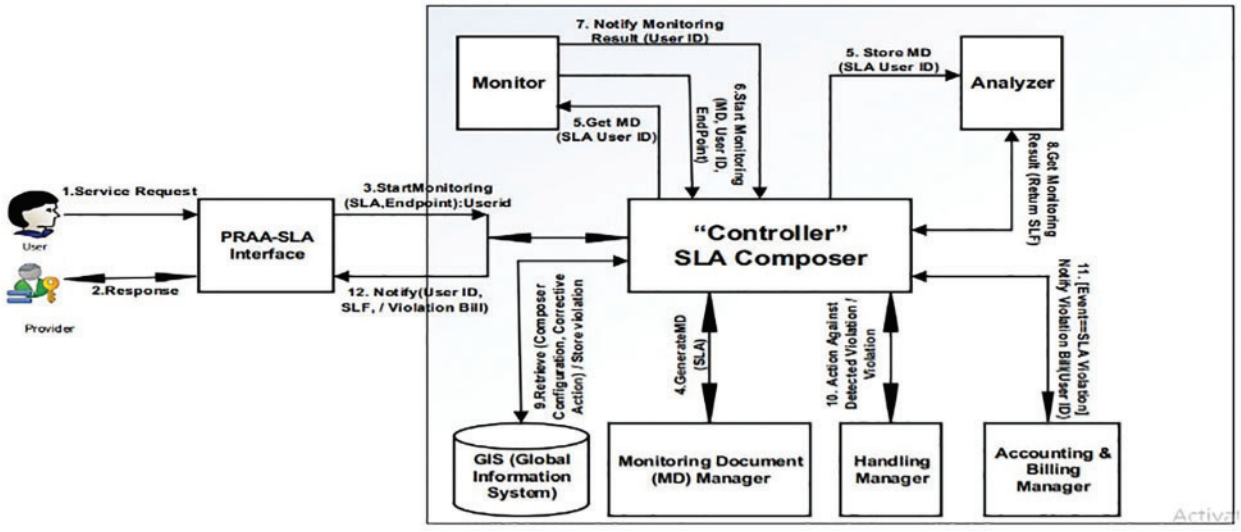

Figure 2: Architecture model of SLA-PRAA

4.1 SLA Monitor

SLA monitor starts monitoring after resource allocation, under the role defined in SLA between user and provider. When the resource is allocated to the user then the SLA template expand and the MDM generates an independent separate monitoring document from SLA for critical and non-critical SLA parameters and assigned it for monitoring. Then monitor starts two separate monitoring

sessions; for the critical and non-critical parameters based on that monitoring documents. Then monitor regularly monitor critical parameters and the monitoring result is directly sent to update MDM. The other hand store monitoring results of non-critical SLA parameters are updated in the QoS repository. After a pre-fixed time (decided by the provider), the MDM is updated according to the monitoring results. The document and client id is then returned to the Controller. The next step is to analyze the monitoring result and compare it with the signed SLA. Algorithm 1 shows the overall process of SLA monitor.

4.2 SLA Analyzer

SLA analysis is to analyze the monitoring result and compare it with the given parametric value which is defined in SLA. SLA analyzer gets SLA id, monitoring document and also retrieves the monitoring document from MDM. Then compare the monitoring results with signed SLA using the following formula.

$$\textit{Un fulfillment}\ (\Delta_{op},\ \mathrm{M}_{op}) \leftrightarrow\ \mathrm{Solve}\ (\mathrm{V} - \mathrm{V}'\Delta,\ \mathrm{D} -\ \mathrm{DDolv} -\ \mathrm{CD}) =\ \mathrm{Dolve} \tag{4}$$

where Δ_{op} defines some guarantees for a set of service properties $\mathrm{M_{op}}$ represent monitored service operation. V is a set of variables, D is a set of domains, and C is a set of constraints.

Algorithm 1: SLA Monitor

Input: γ: Accepted SLA Template (username, provider name, date and time, parameter name, price, start and end time)
Output: ζ: MD (parameter name and type, start and end time) and client SLA id

```
1: Procedure SLA Monitoring
2:        Queue ← SLA
3:        while (Queue is not empty) do
4:              δ ← Delete first SLA from Queue
5:              MDM ← δ.
6:              Expand δ as Memory, MIPS, BW
7:              MDM generate MD for critical & Noncritical Parameter from δ
9:              Monitor ← MD
10:                 Initiate Monitoring sessions in Monitor
11:                 Start session of Noncritical parameter
12:                 Monitor. get monitoring result
13:                QoS Repository ← Monitoring Result
14:                 MDM ← Updated Monitoring Result after pre-fix time
15:                 Start session of critical parameter
16:                 Monitor. get monitoring result
17:                 MDM ← Updated Monitoring Result
18:                  ζ = Monitoring Result
19:                  SLA Analyzer (). get σ
20:       end While
21:          return ζ;
22:   end procedure
```

The (V, D, and C) represents the values before monitoring, and (V', D', and C') represents values after monitoring. If$((\mathrm{V} - \mathrm{V}',\ \mathrm{D} - \mathrm{D}',\ \mathrm{C} - \mathrm{C}') = \emptyset)$ then it means that SLA is fulfilled otherwise

SLA is unfulfilled and then the reason for violations using the following formula.

$$Un\ fulfillment_{exp}\ (\Delta_{op},\ M_{op}) = \text{explain}\ (V - V',\ D -\ DDxpl -\ CD); \tag{5}$$

The handler is then called to handle the violations, store the handler output in a repository and return the service level fulfillment to the controller. Algorithm 2 shows the overall process of SLA analyser.

Algorithm 2: SLA Analyze

Input: ζ: SLA ID, MD (Parameter Name & Type, Start Time, End Time)
Output: σ: Service Level Fulfillment/Unfulfillment.

```
1:  procedure SLA Analysis
2:      Queue← SLA ID, MD. MD: Monitoring Document
3:      while (Queue is not empty) do
4:          Analyzer ← Retrieve M D
5:          Unfulfillment (Δop, Mop) ←→ solve (V − V0, D − D0, C − C0) = ∅
6:          Δop: Define Service Properties, Mop: Monitored Service Operation, V: Variable, V0:
Variable after monitoring, D: Domain, D0: Domain after monitoring, C: Constraint, C0: Constraint
after monitoring
11:         if ((V − V 0, D − D0, C − C0) = ∅) then
12:             Return No violation to a controller;
13              σ = Service Level Fulfillment
14:         else
15:             Unfulfillmentexp (Δop, Mop) = explain (V − V 0, D − D0, C − C0);
16:             SLU nf = explain (V − V 0, D − D0, C − C0) (SLUnf: Service Level Unfulfillment)
18:             SLA Violation Detector & Handler (). get θ
19:             σ = θ
20:         end if else
21:       end While
22:         return σ;
23:  end procedure
```

4.3 SLA Violation Detector and Handler

This technique is called after the analysis phase when SLA violations are detected, or SLA violation occurred than SLA violation detector and handler is called to handle it according to the SLA. Based on the analysis results, after SLA violation detection or violation, the service level un-fulfillment (SLUnf) routine is called. First store SLUnf in the repository, then check a condition if a condition fulfills, means violation detected then store it in SLA-IS (Service Level Agreement Information System). Then allocate a correlative resource to the requester. Else, SLA violation occurred, then store in SLA-IS repository. Then allocate a correlative (due to which SLA detected or violated) resource to the requester and also call accounting and billing manager (A&B M), which calculate SLA violation penalties explained in the next algorithm. After the completion of the violation handler process, the result (Ω) is assigned to output (θ). In the end, return it to the proactive notification. This process is explained in Algorithm 3.

Algorithm 3: SLA Violation Detector and Handler

Input: σ: SLUnf. SLUnf: Service Level Unfulfillment
Output: θ: Handle violations, calculate penalties

```
procedure SLA Violation Detecting & Handling
    Queue ← SLU nf
    while (Queue is not empty) do
        δ ← Delete the first SLUnf from Queue
        ExpPand δ as Memory, MIPS, BW Violations.
        if (SLUnf ≥ Req) then. Req: Requested Resource
            SLA-IS←detected Violations. SLA-IS: SLA-Information System
            Allocate Correlative Resource Where (M IP S ≥ M IP Sreq &
            M ≥ Mreq &BW ≥ BWreq)
        else
            SLA-IS ← Violations
            Allocate Correlative Resource Where (M IP S ≥ M IP Sreq &
            M ≥ Mreq &BW ≥ BWreq)
            A&B M (). get Ω. A&B M: Accounting & Billing Manager
            θ = Ω
        end if
    end While
    return θ;
end procedure
```

4.4 Accounting and Billing Manager

This technique is called when an SLA violation occurred, then accounting and billing manager start penalty calculations. Four things are required, Cost of penalty per violation (P_v) (defined in the SLA), the total time to recover violations V(t), the unit cost of violation(u), and how many times a violation occurred (NV_s) (from SLA-IS). According to the situation, the penalty is calculated using the following equations:

$$Pt = V(t) \times P_u \tag{6}$$

$$Pt = P_v \times NV_s \tag{7}$$

where Pt is the total penalties, P_v is penalty per violation, NVs is Number of violations, $V(t)$ is violation time, P_u is unit cost of violation, and AC_r is account repository. If a penalty is based on violation time, then Eq. (6) is used to calculate the penalty, else if violation depends on the number of violations then use Eq. (7) will be used.in. After this calculation, the total penalty is stored in the accounting repository. The whole process is given in *Algorithm-4*.

Algorithm 4: A and B Manager

Input: φ: SLUnf. SLUnf = Service Level Unfulfillment
Output: Ω: calculate SLA Penalties

(Continued)

Procedure SLA Penalty Calculation.
  Initiate A and B, M: Accounting and Billing Manager
  Pv ← penalty cost — *Pv: penalty per violation*
  Pu ← penalty unit cost — *Pu: Penalty Unit*
  $V(t) \leftarrow \sum_{i=1}^{n} V(t)_i$ — *Vt : Violation time*
  $NVs \leftarrow \sum_{i=1}^{n} Vi_i$ — *NVs: Number of vilations*
  if (Count V(t)) Then
    P = V(t) x Pu — *P: total penalty*
  Else
    P = Pv x NVs
  End if
  ACr ← P; — *ACr: Accounting Repository*
  Ω = Calculate SLA penalties
  Return Ω;
End Proceduere

4.5 Flow Model for SLA-PRAA

The function of SLA-PRAA is explained with the help of sequential diagram given in Fig. 3. Once the client provides a WS-agreement for monitoring process to monitor its requirments. The analyzer also stores the WS-agreement independently of this approach. This WS-agreement document is also sent for generating monitoring documents by the monitoring document manager. For monitor configuration, all information is stored in monitoring document and also stored monitoring results of agreed metric. Once done, monitor start monitoring and the results are updated in the monitoring document at the endpoint after the consumption of all services. To denote specific monitoring session, client identifier is generated for SLA. If there are multiple clients of SLA-PRAA who want to monitor a unique SLA, then a multi dissimilar monitoring session is created and with different client-ids. A newly updated monitoring document is notified to a controller and the controller sent it to the analyzer for the analysis of service level fulfillment. It will then notify the service level fulfillment to the consumer. There is a loop in the monitoring and analysis session. If the WS-agreement is not fulfilled, then store the WS-agreement violations report in global-view and send corrective action to the handler manager according to the knowledge base or past experiences. If a user over user the resources, then de-allocate the resources otherwise allocate normal resources to the job and initiate account and billing manager to calculate the penalties. After this calculation total penalty is stored in the accounting repository. Notification of handling violation and penalty calculations is generated to the controller and SLA-PRAA client.

5 Experimental Setup

A simulation-based approach is used to analyze and evaluate our proposed SLA-PRAA model. We have implemented the proposed SLA-PRAA model in the CloudSim simulation tool. CloudSim is an open-source java-based scalable simulation tool that presenting some attributes, such as sustaining modeling service broker, allocation policies of application, provisioning of resources, and cloud computing large scale simulation environment. For a detailed discussion on CloudSim interested readers are directed to [29]. We compare our result of SLA-PRAA with three existing well-known

policies, i.e., Pre-fix analysis [10,11], Early Analysis [9], and Time Based Monitoring policy (Up and Downtime) [25]. A detailed description in terms of jobs submitted by one or more users and resources provided from cloud computing and the specification of the experimental environment are presented in the following sections.

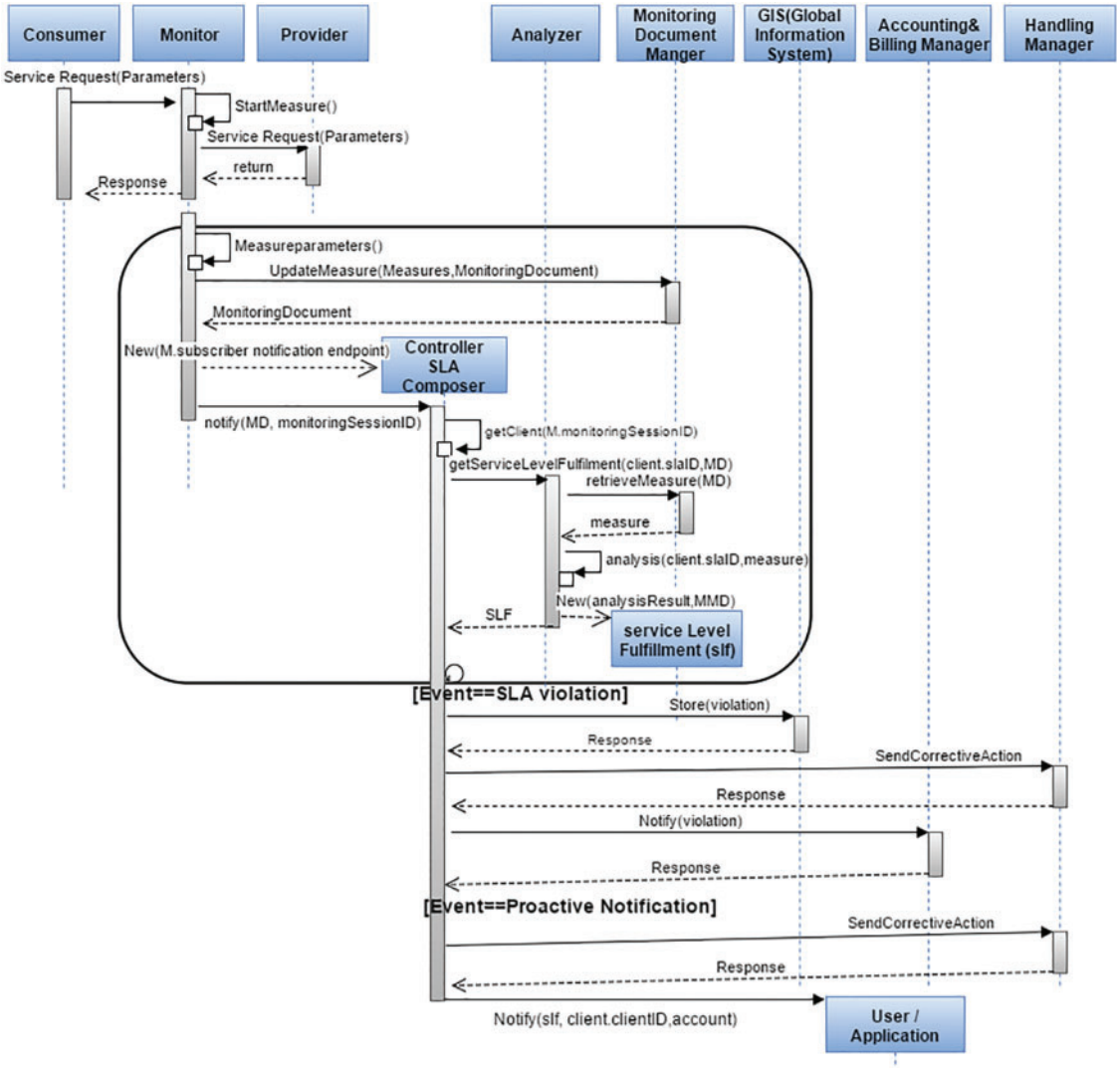

Figure 3: Sequential diagram of SLA-PRAA

5.1 Resource Modeling

Since our target system is a generic cloud computing environment, therefore, it is essential to analyze a scale virtualized data center infrastructure. However, the implementation and evaluation of the algorithms proposed in such type of infrastructure are very costly and time-consuming. In addition, the execution of large-scale repetitive experiments to analyze and compare the results of proposed algorithms is also difficult. Thus, simulation was used for the performance evaluation. We have used an extension of the Cloudsim toolset [30] and all its infrastructure provided as a platform

for simulation. The adoption of Cloudsim Toolkit allows us to perform repeatable experiments for large-scale virtualized data centers.

For our infrastructure configurations, we simulate a cloud infrastructure that includes a data center with 40-100 heterogeneous physical machines [31] are installed with (10-25)G4 HP ProLiant ML110, (10-25) HP ProLiant ML110 G5, (10-25) IBM x3250 and (10-25) IBM x3550 server. The characteristics of these machines are shown in the below Tab. 2. Virtual machines are assumed to correspond to four types of Amazon EC2 virtual machines, as shown in Tab. 3.

5.2 Application Modeling

For our proposed policy SLA-PRAA, we have created a simulated platform for resources in the cloud, such as computing, storage, and bandwidth. To measure the performance of our proposed method, we have considered one user in our experiment, who submitted 200, 400, 600, and 800 jobs separately like (Olio Search Operations) [31] to 100 homogeneous virtual machines. We have presented a virtual machine allocation method where more correlated information among different virtual machines are considered thus, energy consumption is minimized. The price of each virtual machine instance is the same as that used by Amazon EC2 VM for different sizes. Although only four types of virtual machines are considered, our model can easily be extended to other types of virtual machine instances.

Table 2: Shows the resource specification [32]

Server	CPU model	Cores	CPU (MIPS)	Frequency (MHZ)	RAM(GB)
(10-25)G4 HP ProLiant ML110	Intel Xeon 3040	2	6000	1860	4
(10-25)HP ProLiant ML110 G5	Intel Xeon 3075	2	6000	2660	8
(10-25)IBM x3250 Server	Intel Xeon 3470	4	7000	2933	8
(10-25) x3550 IBM 200 Server	2∗ Intel Xeon5675	2∗6	8000	3067	16

Table 3: Shows the VMs types [33,34]

VM type (Xen)	CPU (MIPS)	RAM(GB)
VM-1	1000	0.85
VM-2	2000	3.75
VM-3	2500	1.7
VM-4	3500	0.613

5.3 Performance Evaluation Parameters

To evaluate the performance of our proposed algorithm with an existing algorithm, we use the following QoS parameters.

5.3.1 Energy Consumption

Our first performance evaluation parameter is the energy consumption of physical resources. Greater the utilization of physical resources more energy will be consumed, but if there are fewer active physical resources, smaller energy will be consumed [35–37]. We have developed and used the following equation to calculate the energy consumption of physical resources in the data center [38,39].

$$E(c) = \mathrm{r(i)} * E(max) + (\mathrm{n} - \mathrm{r}) * \mathrm{C(u)} \tag{8}$$

where $E(c)$ is total energy consumption, $r(i)$ is the number of idle resources, $E(max)$ represents the maximum energy consumed by an active idle resource, $(n\text{-}r)$ represents the number of currently used resources and current energy consumption and CPU usage is represented by $C(u)$. For our simulations, $E(max)$ is assigned to 250 W, which is a normal value for each active resource.

5.3.2 SLA Detections

When SLA is violated, it will be detected first and then may or may not be violated. The phenomenon is called SLA detections. For this purpose, we have defined a suitable threshold in terms of resource provision [7].

$$SLA_{\mathrm{D}} = D(V_{Th} \geq V_{va}) \tag{9}$$

where SLA_D represents SLA detection, V_{Th} is a threshold value and Vva is a variable value for each resource.

5.3.3 SLA Violation Miss-Detection Cost

SLA violation Miss-detection Cost is an economic factor, which is directly dependent on the agreed cost of SLA penalty for precise application. SLA Violation Miss-detection Cost is the total cost/penalty for each miss-detection due to which SLA is violated [36]. This cost is calculated according to Eqs. (4) and (5).

5.3.4 Number of SLA Violation

It is the total number of SLA violations either from the user side or from the service provider. SLA is violated from the service provider side if it fails to provide the requested resources to the user on time. Similarly, SLA is violated from the user side if a user either over-uses the requested resources or does not release the resources on time [40].

$$\mathrm{SLA}_{T.V} = \{(R_A < R_{\mathrm{R}})1 + (R_A < R_{\mathrm{R}})2 + (R_A < R_{\mathrm{R}})3 + \cdots\cdots + (R_A < R_{\mathrm{R}})n\} \tag{10}$$

$$\mathrm{SLA}_{T.V} = \sum_{i=1}^{n}((R_A < R_{\mathrm{R}})n) \tag{11}$$

where $SLA_{T.V}$ is total SLA violations, R_A is the available resources, and R_{R} is the requested resources.

5.3.5 User Satisfaction

User satisfaction is the percentage upon which a user can be satisfied with the performance of a system in terms of resource requests and resource allocations. Low number of SLA violations indicates high user satisfaction. Based on user satisfaction, cloud providers are catogeried interms of trustworthness [41,42]. The user satisfication is calculated as below

$$U(s) = \frac{N(a)}{N(r)} * 100 \tag{12}$$

where, $U(s)$ represents user satisfaction percentage, $N(a)$ represents the number of resources allocated and $N(r)$ is the number of resources requested.

6 Results and Discussion

To evaluate the proposed method, we have defined and simulated the proposed work through three different scenarios. In each experiment, we have calculated the Energy Consumption, SLA detection, Cost of Miss-detection, Number of SLA violations, and User Satisfaction. We compare our result of SLA-PRAA with three existing well-known policies Pre-fix analysis [13], Early analysis [11], and Time Based Monitoring policy (Up and Downtime) [25].

6.1 Scenario 1

In scenario 1, we have executed 200, 400, 600, and 800 jobs in our simulated environment with 100 numbers of VMs and 50 hosts. This setup will help us to examine each parameter for the proposed and existing policies (SLA-PRAA, Pre-fix, and Early Time-base). The existing policies also evaluated their performance with the same number of jobs. The main objective of this scenario is to evaluate the performance parameters in terms of variations of job number, while the number of hosts and VMs remain constant. We have evaluated each performance parameter as below:

6.1.1 Energy Consumption

The objective of this assessment is to evaluate the performance parameter i.e., Energy Consumption in terms of job variations, while the number of hosts and VMs remain constant. The average results for the proposed policy SLA-PRAA compared with existing policies in terms of energy consumption are plotted in Fig. 4 which reveal that the proposed SLA-PRAA consumed 52.15% less energy from the existing policies. The reason for less energy consumption for the proposed policy SLA-PRAA is that we have used an early analysis mechanism to critical SLA metrics, due to which network traffic and the energy consumption decreased. While the existing policies use early analysis for both critical and non-critical metrics, due to which energy consumption increased. We have also used available resources by introducing the concept of reserved resources. Whereas, the reserved resources are in off condition and whenever user request does not fulfill then it makes active the reserved resources.

6.1.2 SLA Violation Detection

In this section the objective is to evaluate the performance parameters i.e., SLA detection in terms of job variations, while the number of hosts and VMs remain constant. In Fig. 5, the average detected SLA violations are plotted on Y-axis, while the number of jobs are given on the X-axis for all compared policies. This comparison reveals that SLA-PRAA detects SLA violation on time, with 29.26% more SLA violations detected from pre-fix and time-based analysis and 20.14% more SLA

violations detection from exiting time based analysis policy. We evaluate the monitoring result of SLA-PRAA using the mechanism of early analysis for critical metrics only, due to which SLA violations are detected early. While the existing policies use early analysis for both critical and non-critical metrics, which increased network traffic.

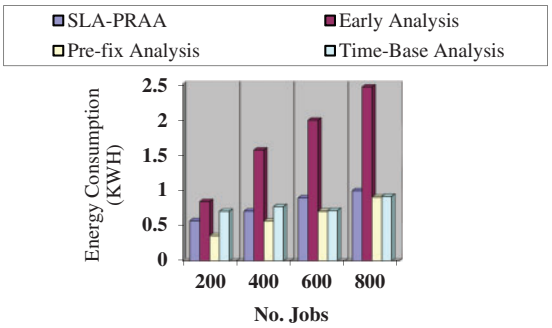

Figure 4: Energy consumption of monitoring systems on different numbers of Jobs

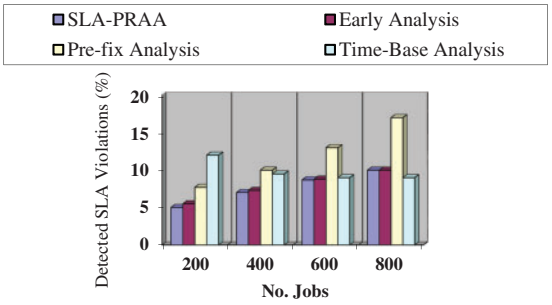

Figure 5: SLA violation detection of SLA violation monitoring systems on different numbers of jobs

A trade-off between SLA violations and energy consumption is reported in [35], therefore the ratio of SLA detections increased for the proposed policy. The pre-fix and time based analysis policies sent message after a specific time duration due to which they detect less SLA violations.

6.1.3 SLA Violation

The objective of this experiment is to evaluate the performance parameter i.e., SLA violation in terms of job variations, while the number of hosts and VMs remain constant. In Fig. 6 we have considered SLA violations on *Y-axis*, while the number of jobs on the *X-axis* for the compared policies. The proposed SLA-PRAA provides 46.46% fewer violations from existing policy Pre-fix analysis and also 39.18% fewer violations from Time Based analysis. The reason for a smaller number of SLA violations for the proposed policy is mainly due to the concept of reserved resources. When a system is going to violate SLA then we provide reserved resources proactively so that the system avoid SLA violation, due to which the ratio of handling the SLA increased and the ratio of SLA violation decreased.

6.1.4 SLA Violation Miss-Detection Cost

The objective here is to evaluate the performance parameter i.e., SLA Violation Miss-detection Cost in terms of job variations, while the number of hosts and VMs remain constant. In Fig. 7 we

have considered SLA Violation Miss-detection Cost on *Y-axis*, while the number of jobs on *X-axis* for the compared policies. We can see that the proposed SLA-PRAA provides 46.88% less SLA Violation Miss-detection Cost from the existing policy of Pre-fix analysis and 39.69% less from Time Based analysis. The SLA Violation Miss-detection Cost is directly dependent on SLA violations. In SLA-PRAA we have used a proactive resource allocation approach in case of violation, due to which the ratio of SLA violation decreased. Therefore, total SLA Violation Miss-detection costs will be reduced by decreasing SLA violations as compared to the existing policies.

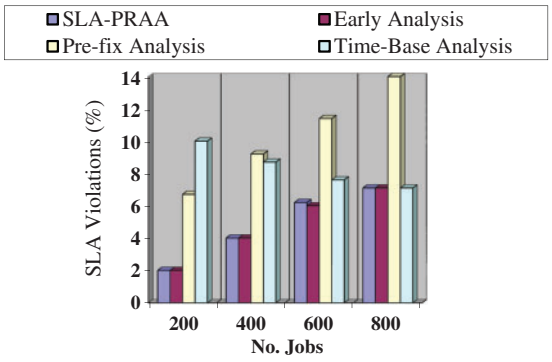

Figure 6: SLA violation of SLA violation monitoring systems on different numbers of Jobs

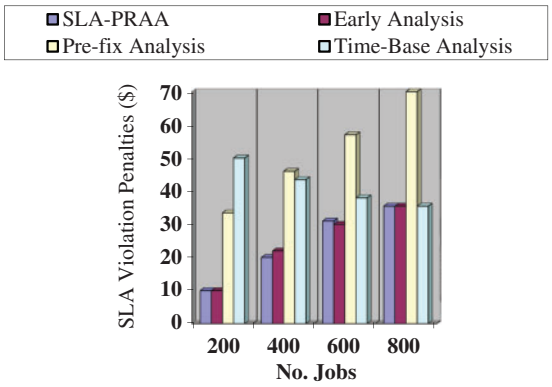

Figure 7: SLA violation penalties of SLA violation monitoring systems on different numbers of jobs

6.2 Scenario 2

In scenario 2, we have provided resources to 10, 20, 30, and 40 users in our simulated environment with 100 numbers of VMs, and 50 hosts. This parmeter setup will help to consider reasonable number of active users. Similarly, the other policies also evaluated their performance with the same setup.

The objective of this assessment is to evaluate the performance parameter i.e., user satisfaction in terms of job variations, while the number of hosts and VMs remain constant. The average results for the proposed policy SLA-PRAA PRAA compared with existing policies in terms of user satisfaction are plotted in Fig. 8 which reveal that the proposed SLAP-RAA provides 24.72% more user satisfaction from the pre-fix analysis and 18.29% more user satisfaction from time based analysis. The reason for more user satisfaction for the proposed policy as compared with existing policies is maily because we have used runtime monitoring mechanism with a suitable threshold. In SLA-PRAA we employed the

mechanism of early analysis for critical metrics, due to which help to detect SLA violations early and used pre-fix time analysis for non-critical metrics, which help to reduce network traffic. When a system is detected to violate SLA then we provide reserved resources proactively to avoid SLA violation, due to which the ratio of handling SLA violation increased and the ratio of SLA violation decreased. Therefore, the user satisfaction increased as compared with other existing policies.

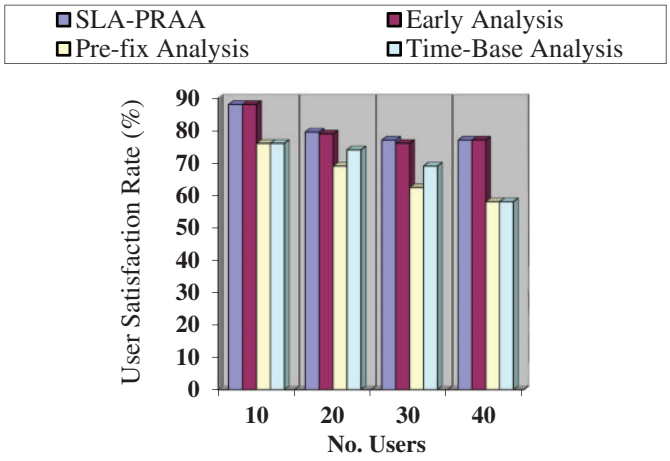

Figure 8: User Satisfaction of SLA Violation monitoring systems on the different number of users

7 Conclusion

Service level agreement is the key to ensure a service provider delivers the agreed terms of services. Cloud consumers with SLA parameters and negotiation can increase the trust level of relationship. We have presented an energy-efficient and SLA-based resource allocation policy i.e., SLA-PRAA. The performance evaluation parameters that we have considered are energy consumption, SLA detection, SLA violation misdetection cost, number of SLA violations, and user satisfaction. We compare our result of SLA-PRAA with three well-known existing policies of pre-fix analysis, early analysis, and time-based monitoring policy (up and down time). The result shows that our proposed SLA-PRAA policy outperformed and more efficient than the existing policies. Simulation results revealed that SLA-PRAA has 32% less energy consumed as compared to the othe policies. The SLA-PRAA has detected 20.15% more SLA violation than pre-fix analysis and 28.49% more than time-based analysis. Simillarly, the proposed SLA-PRAA has 45.85% less SLA violations than pre-fix analysis and 37.79% less than time-based analysis. In terms of SLA violation miss-detection cost, SLA-PRAA has 45.85% less than pre-fix analysis and 37.79% less than time-based analysis. Also, the SLA-PRAA has 24.17% more user satisfaction as compared to pre-fix analysis and 18.17% more than time-base analysis.

The main limitation of the proposed policy is the exclusion of security and fault tolerance. As future work, we are planning to include these parameters in our proposed model so that a reliable and robust system can be defined.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] L. Wu, S. K. Garg, S. Versteeg and R. Buyya, "SLA-Based resource provisioning for hosted software as a service applications in cloud computing environments," *IEEE Transactions on Services Computing,* vol. 7, no. 3, pp. 1–7, 2014.

[2] I. Breskovic, C. Haas, S. Caton and I. Brandic, "Towards self-awareness in cloud markets: A monitoring methodology," in *IEEE Int. Conf. on Dependable, Autonomic and Secure Computing*, Sydney, Australia, pp. 8, 2011.

[3] V. C. Emeakaroha, S. Paulo and C. A. F. De Rose, "CASVid: Application level monitoring for SLA violation detection in clouds," in *IEEE Annual Computer Software and Applications Conf.*, Izmir, Turkey, pp. 10, 2012.

[4] C. Redl, I. Breskovic, I. Brandic and S. Dustdar, "Automatic SLA matching and provider selection in grid and cloud computing markets," in *ACM/IEEE Conf. on Grid Comp.*, Beijing, China, 2012.

[5] L. Ye, H. Zhang and J. Shi, "Verifying cloud service level agreement,"*IEEE Global Communications Conf. (GLOBECOM)*, Anaheim, California, USA, pp. 777–782, 2012.

[6] K. Kritikos, D. Plexousakis and P. Plebani, "Semantic SLAs for services with Q-sLA," in *IEEE Int. Conf. on Web Services (ICWS)*, Madrid, Spain, pp. 24–33, 2016.

[7] C. Müller, M. Oriol, X. Franch, J. Marco, M. Resinas *et al.,* "Comprehensive explanation of SLA violations at runtime," *IEEE Transactions on Services Computing,* vol. 7, no. 2, pp. 16, 2014.

[8] C. Mu, M. Resinas and A. Ruiz-corte, "Automated analysis of conflicts in WS-agreement," *IEEE Transactions on Services Computing,* vol. 7, no. 4, pp. 530–544, 2014.

[9] Al Falasi and A. Ain, "A model for multi-levels SLA monitoring in federated cloud environment," in *IEEE Int. Conf. on Autonomic and Trusted Computing*, Vietri sul Mare, Italy, pp. 8–12, 2013.

[10] S. Anithakumari and K. Chandrasekaran, "Monitoring and management of service level agreements in cloud computing," in *Int. Conf. on Cloud and Autonomic Computing*, Boston, Massachusetts, USA, pp. 204–207, 2015.

[11] C. Muller, M. Oriol, M. Rodriguez, X. Franch, J. Marco *et al.,* "SALMonADA: A platform for monitoring and explaining violations of WS-agreement-compliant documents," in *Int. Workshop on Principles of Engineering Service-Oriented Systems*, Zurich, Switzerland, vol. 1, pp. 43–49, 2012.

[12] M. D. Penta, M. Bruno, G. Esposito, V. Mazza and G. Canfora, "Web services regression testing," in L. Baresi and E. D. Nitto (Eds.), *Test and Analysis of web Services,* Berlin, Heidelberg: Springer, 2007. https://doi.org/10.1007/978-3-540-72912-9_8.

[13] G. Spanoudakis and K. Mahbub, "Non-intrusive monitoring of service-based systems,"*International Journal of Cooperative Information Systems,* vol. 15, no. 3, pp. 325–358, 2006.

[14] W. Hussain, F. K. Hussain, M. Saberi, O. K. Hussain and E. Chang, "Comparing time series with machine learning-based prediction approaches for violation management in cloud SLAs," *Future Generation Computer Systems,* vol. 89, pp. 464–477, 2018.

[15] X. H. Wu, J. He, B. Li and Y. Pei, "Personalized QoS prediction of cloud services via learning neighborhood-based model," in *Int. Conf. on Collaborative Computing: Networking, Applications and Worksharing*, Wuhan, China, 2015.

[16] P. Leitner, B. Wetzstein, F. Rosenberg, A. Michlmayr, S. Dustdar *et al.,* "Runtime prediction of service level agreement violations for composite services," in *Int. Conf. on Service-Oriented Computing*, Stockholm, Sweden, 2009.

[17] R. A. Hemmat and A. Hafid, "SLA violation prediction in cloud computing: A machine learning perspective," *ArXiv*, abs/1611.10338, 2016.

[18] A. Keller and H. Ludwig, "The WSLA framework: Specifying and monitoring service level agreements for web services *network*,", *Journal of Network and Systems Management Volume,* vol. 11, pp. 57–81, 2003.

[19] K. Mahbub and G. Spanoudakis, "Monitoring WS-agreements: An event calculus based approach," New York, NY, USA Springer-Verlag, pp. 265–306, 2007.

[20] A. Sahai, V. Machiraju, M. Sayal, A. Moorsel and F. Casati, "Automated SLA monitoring for web services," in *IEEE Int. Workshop on Distributed System Operation and Management*, Montreal, Canada, pp. 28–41, 2010.
[21] M. C. and C. Kotsokalis, "Establishing and monitoring SLAs in complex service based systems," in *IEEE Int. Conf. on Web Services*, pp. 783–790, 2009.
[22] O. Comuzzi, G. Dynamic, M. Comuzzi and G. Spanoudakis, "Dynamic set-up of monitoring infrastructures for service based systems," in *ACM Symposium on Applied Computing*, Sierre Switzerland, pp. 2414–2421, 2010.
[23] V. C. Emeakaroha, M. A. S. Netto, R. N. Calheiros, I. Brandic, R. Buyya *et al.,* "Towards autonomic detection of SLA violations in cloud infrastructures," *Future Generation Computing Systems,* vol. 28, no. 7, pp. 1017–1029, 2012.
[24] W. E. F. Raimondi and J. Skene "Efficient online monitoring of web-service SLAs," in *ACM SIGSOFT Int. Symp. on Foundations of Software Engineering*, pp. 170–180, 2008.
[25] S. D. A. Michlmayr, F. Rosenberg and P. Leitner, "Comprehensive QoS monitoring of web services and event based SLA violation detection," in *Int. Workshop on Middleware for Service Oriented Computing*, Urbana Champaign Illinois, pp. 1–6, 2009.
[26] C. Müller, M. Oriol, M. Rodríguez, X. Franch, J. Marco *et al.,* "SALMonADA: A platform for monitoring and explaining violations of WS–agreement–compliant documents," *4th Int. Workshop on Principles of Engineering Service-Oriented Systems (PESOS),* Zurich Switzerland, pp. 43–49, 2012. https://doi.org/10.1109/PESOS.2012.6225938.
[27] V. C. Emeakaroha, I. Brandic, M. Maurer and S. Dustdar, "Low level metrics to high level SLAs - lom2his framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments." in *Int. Conf. on High Performance Computing & Simulation*, Caen, France, pp. 48–54, 2010. https://doi.org/10.1109/HPCS.2010.5547150.
[28] E. Tsang, "*Foundations of constraint satisfaction*," New York, NY, USA: Academic Press, pp. 439–467, 2005.
[29] Y. Sharma, B. Javadi, W. Si and D. Sun, "Reliability and energy efficiency in cloud computing systems: Survey and taxonomy," *Journal of Network and Computer Applications,* vol. 74, pp. 66–85, 2016.
[30] R. N. Calheiros, R. Ranjan, A. Beloglazov and A. F. De Rose, "Cloudsim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *ACM Software—Practice & Experience,* vol. 41, no. 1, pp. 23–50, 2011. https://doi.org/10.1002/spe.995.
[31] S. Mehmi, H. K. Verma and A. L. Sangal, "Simulation modeling of cloud computing for smart grid using," *Journal of Electrical Systems and Information Technology,* vol. 4, no. 1, pp. 159–172, 2017.
[32] E. Arianyan, H. Taheri and S. Sharifian, "Novel energy and SLA efficient resource management heuristics for consolidation of virtual machines in cloud data centers, "*Computers and Electrial. Engineering,* vol. 47, pp. 222–240, 2015.
[33] V. Durao, F. DCarvalho, J. F. S. Fonseka and A. Garcia, "A systematic review on cloud computing," *Journal of Supercomput,* pp. 1–26, 2014.
[34] S. Kumar, A. Nadjaran and S. K. Gopalaiyengar, "Journal of network and computer qapplications SLA-based virtual machine management for heterogeneous workloads in a cloud datacenter," *Journal of Network and Computer Applications,* vol. 45, pp. 108–120, 2014.
[35] Y. Jararweh, M. Jarrah, Z. Alshara, M. Noraden and M. Al-ayyoub, "Simulation modelling practice and theory cloudExp: A comprehensive cloud computing experimental framework," *Simulation Modelling Practice and Theory,* vol. 49, pp. 180–192, 2014.
[36] E. Arianyan, H. Taheri and V. Khoshdel, "Novel fuzzy multi objective DVFS-aware consolidation heuristics for energy and SLA efficient resource management in cloud data centers," *Journal of Network and Computer Applications,* vol. 78, pp. 43–61, 2017..
[37] T. Mastelic, A. Oleksiak, H. Claussen, I. Brandic, J. -M. Pierson *et al.,* "Cloud computing: Survey on energy efficiency," HAL Id: hal-01153714," *ACM Computing Surveys*, vol. 47, no. 2, pp. 1–36, 2015.

[38] Z. Zhou, Z. -G. Hu, T. Song and J. -Y. Yu, "A novel virtual machine deployment algorithm with energy efficiency in cloud computing," *Journal of Central South University,* vol. 22, pp. 974–983, 2015.
[39] Y. Xiaoyong, T. Hongyan, L. Ying, J. Tong, L. Tiancheng *et al.,* "A competitive penalty model for availability based cloud SLA," in *IEEE Int. Conf. on Cloud Computing*, New York City, NY, USA, pp. 964–970, 2015.
[40] P. H. P. Castro, V. L. Barreto, S. L. Corrêa, L. Z. Granville and K. V. Cardoso, "A joint CPU-RAM energy efficient and SLA-compliant approach for cloud data centers," *Computer Networks,* vol. 94, pp. 1–13, 2016.
[41] S. Singh and J. Sidhu, "Compliance-based multi-dimensional trust evaluation system for determining trustworthiness of cloud service providers," *Future Generation Computer Systems,* vol. 67, pp. 109–132, 2017.
[42] P. Varalakshmi and T. Judgi, "Multifaceted trust management framework based on a trust level agreement in a collaborative cloud,"*Computers and Electrical Engineering,* vol. 59, pp. 110–125, 2017.