Towards an adaptive QoS-driven monitoring of cloud SaaS

Mohamed Adel Serhani* and Yacine Atif

College of Information Technology,
UAE University,
P.O. Box 17551, Al-Ain, UAE
Email: serhanim@uaeu.ac.ae
Email: Yacine.atif@uaeu.ac.ae
*Corresponding author

Abdelghani Benharref

Faculty of Engineering and Information Sciences,
University of Wollongong in Dubai,
Dubai, UAE
Email: AbdelghaniBenharref@uowdubai.ac.ae

Abstract: The abundance of services offered by cloud providers raises the need for Quality of Service (QoS) monitoring to enforce Service Level Agreements (SLAs). In this paper, we propose a cloud-monitoring scheme, which offers flexible and dynamically reconfigurable QoS monitoring services to adapt to various cloud based service characteristics. We propose to grant cloud service consumers the option to decide whether to switch to another service provider, if the current provider frequently violates the corresponding SLA contract. For cloud service providers, the monitoring scheme provides dashboard-like indicators to continuously manage the performance of their SaaS platform and visualise the monitoring data. Our experimental evaluation involves a real cloud platform to illustrate the capability of our monitoring scheme in detecting and reporting violations of SLAs for both single and composite SaaS services. A particular emphasis on visualisation options is highlighted when revealing and reconfiguring monitoring data in a user-friendly display.

Keywords: utility computing; cloud; cloud monitoring; SaaS; composite SaaS; SLA; QoS.

Reference to this paper should be made as follows: Serhani, M.A., Atif, Y. and Benharref, A. (2014) 'Towards an adaptive QoS-driven monitoring of cloud SaaS', *Int. J. Grid and Utility Computing*, Vol. 5, No. 4, pp.263–277.

Biographical notes: Mohamed Adel Serhani is an Associate Professor at the College of Information Technology at UAE University. He received a PhD in Computer Engineering from Concordia University, Canada, in 2006. His main research interests include web services architecture, management and QoS monitoring; web services in MANET; management of communities of web services; web services applications and security; and cloud data centres for intensive e-health data, applications, and services; SLA enforcement in cloud data centres, architectures for e-health monitoring and prevention-based SOA and cloud. He published a number of research articles, some of which were published in prestigious journals and conferences.

Yacine Atif received the PhD degree in Computer Science from Hong Kong University of Science and Technology (HKUST) in 1996. After graduation, he worked at Purdue University in the USA as a Post-Doc and then joined a faculty position at Nanyang Technological University (NTU) in Singapore. Since 1999, he is with the UAE University as a faculty, then Program Chair at the College of Information Technology. He has made a number of research contributions particularly in the area of semantic web and internet computing. He is also involved in the technical programmes of several research forums.

Abdelghani Benharref is currently working at the University of Wollongong in Dubai. He received a PhD in Computer Engineering from Concordia University, Canada in 2007. He has been working before for SAP Montreal, Concordia University, Abu Dhabi University, and the United Arab Emirates University. His interest domains include but not limited to: e-health systems, mobile applications, web services, web services composition, management of web service, QoS of web services, cloud computing, software testing, protocol design and validation.

1 Introduction

Cloud computing extends high performance computing approaches, mainly cluster-based and grid computing, to reach services anytime, anywhere and running on any device as long as there is an adequate access to the internet. Cloud computing (Buyya et al., 2010; Rimal et al., 2009; Armbrust et al., 2010; Buyya et al., 2008) has become an effective solution to scale up the potential of hardware and software resources to an extent that was not thought of before. Cloud computing has emerged as a cost-effective trend for the provision of infrastructure and services, while freeing users from managing and maintaining them. The fundamental and driving idea behind the cloud paradigm is that users should be able to easily use various services, applications or data storage facility, wherever and whenever they are, as long as they have an internet connection and some basic client software. The latter can be a standard browser or any simple standalone application that can generate cloud requests and process returning responses. This idea waives the complexity of managing remotely hosted data and applications, leading to high performance, and high availability. The cloud paradigm is based on three different service-oriented models: software as a Service (SaaS), Platform as a Service (PaaS) and Infrastructure as a Service (IaaS) (Armbrust et al., 2010). Organisations are increasingly offering and/or contracting cloud service providers to increase profitability, harness complexity, expand scalability, elevate robustness and efficiently deal with load variations.

While such a paradigm is very appealing, managing dynamic and distributed services in a cloud environment is a very challenging mission. In fact, this management requires identifying dependencies between cloud services as well as keeping track of all services' state changes. Furthermore, guaranteeing Quality of Service (QoS) in provisioning cloud services raises many challenges, including the unpredictability of the underlying network connecting cloud clients and providers, the lack of control over client's resources, and the dynamic nature of QoS monitoring. In fact, maintaining a pre-agreed level of performance of cloud services should be ensured at every single point of the underlying cloud infrastructure, platform and application services. This is crucial, particularly when cloud services are the result of a combination of different and complex workflows created by various providers to add value to composite cloud services.

When cloud customers contract a cloud service provider, they usually negotiate QoS levels for provisioning that service. Cloud providers usually define QoS levels for each service. Customers select a level of service from a predefined range of levels, or can even negotiate and personalise one on demand. During service provisioning, cloud clients and/or providers might be interested in checking to what extent the pre-agreed QoS level is being accomplished. For cloud clients, this means they will be able to select a provider that best matches their service needs at given QoS requirements. For example, clients of a storage cloud service would like to make sure that their data are securely stored as claimed by the provider, and backup copies are snapshotted on pre-agreed time intervals. For cloud providers, having well-defined Service Level Agreements (SLAs) can be used as a competitive advantage to preserve their customers and even to grab a better market share. In fact, in an environment where many providers are offering similar cloud services, a client might need a pragmatic assessment tool to discriminate between providers.

Monitoring cloud service provision is of prime importance for clients and providers alike. For clients, this is useful to check if they are getting the service, they are paying for at the pre-agreed QoS levels. For cloud providers, identifying variations of QoS provisioning is the first step in a series of corrective measures to improve their QoS levels. Checking the validity of a provisioned QoS level necessitates an appropriate monitoring approach. All exchanged messages between cloud services and their consuming clients along with messages' timestamps are expected to be analysed in real time to detect violations as soon as they occur. Such a thorough monitoring of cloud services is difficult to implement since it has to fulfil SLA requirements across all heterogeneous cloud providers.

From a technical perspective, cloud computing makes QoS monitoring extremely challenging, for a number of reasons, which we address in this paper. First, the dynamic, shared and virtualised SaaS services make monitoring a very complex and challenging process. This process has to cope with the dynamic composition of services, to be able to track service changes and provide continuous monitoring, even while changes are in progress. Most of existing monitoring solutions for cloud SaaS are unable to monitor the performance of SaaS services that are being pulled from a range of providers, and do not support hybrid/composite SaaS-based applications.

The main objective of this paper is to develop a cloud-monitoring service approach, which is flexible, autonomic, self-adjusting, platform-independent, reliable and available to all cloud roles (i.e. clients, providers and third parties). The approach is based on the cloud paradigm itself since the core component of the approach, called the monitor, is designed and implemented as a cloud service. The main factor behind such a design is to take maximum benefit from a cloud environment for monitoring purposes. This monitor is a platform and role-independent in such a way that it can be used by all cloud roles and platforms.

The paper starts by concisely defining some of the main attributes of an SLA agreement (e.g. response time, availability, efficiency and expected cost) between cloud providers and contractors. In such SLAs, each of these attributes might define a minimum, an average, or a ceiling threshold QoS level that cloud providers must adhere to. To maintain these QoS levels, an overseeing entity is expected to monitor the activity of the delivered services, while abiding by SLA parameters. This same monitoring entity could also be used as a source for accounting and billing purposes.

Clients and providers of cloud services are the main users of our monitor entity; they invoke the monitor entity to undertake monitoring activities of SaaS cloud service. Whoever initiates the monitoring has to make sure the monitor gets information as the input: (a) cloud SLA

description and (b) exchanged traces between clients and providers. The traces include the messages themselves and their timestamps. The monitor then analyses, on the fly, these traces and generates appropriate alarms whenever it detects a violation. The monitoring scheme that we propose in this paper has the following features:

- *Flexible*: may involve monitoring services used by the same customer but deployed on different clouds.
- *Re-configurable*: *self-adjusting* to adapt to the type of cloud service (single or composite service). The monitoring scheme will be automatically triggered based on the type of cloud service being monitored.
- *Reliable*: the entity responsible for monitoring is itself a cloud service that might benefit from the cloud key features including scalability and elasticity.
- *Available*: should be available to all cloud roles (users, providers and third parties).
- *Role-independent*: the monitoring architecture is not linked to any client or cloud provider.
- *Platform-independent*: the monitoring solution should not rely on underlying technologies being used by clients and/or cloud providers.
- *Lightweight*: should not require or present a high communication overhead (light monitoring messages).
- *Accurate*: should detect QoS violation as soon as it occurs and reports it to concerned parties.
- *Dynamic*: supports on-the-fly monitoring of SLAs while clients are using cloud services.
- *Pre-assessment*: should allow clients and providers to assess the validity of any SLA even before clients start using the associated cloud service.

Given the above features, we hereby summarise the key contributions of our proposed work in this paper:

1 An independent cloud service monitoring scheme, which offers flexible and dynamically reconfigurable monitoring services that adapt to the cloud service characteristics and inherently observe QoS performance of both single and composite SaaS services.

2 A monitoring scheme that is able to identify the source of violation to manage its propagation effect, and eventually isolate the service responsible of the violation for a possible remedial action (e.g. service substitution).

3 Support for synchronous and asynchronous SaaS composition.

4 Dashboard-like indicators to continuously manage the performance and visualise the monitoring data of their services at different levels.

5 A visual evaluation of monitoring scheme capabilities used to report and to adjust monitoring data displays, in a user-friendly and realistic way.

6 Certification facility of monitored SaaS services to improve trust towards service providers.

7 Evaluation of the proposed monitoring scheme on a real cloud SaaS platform.

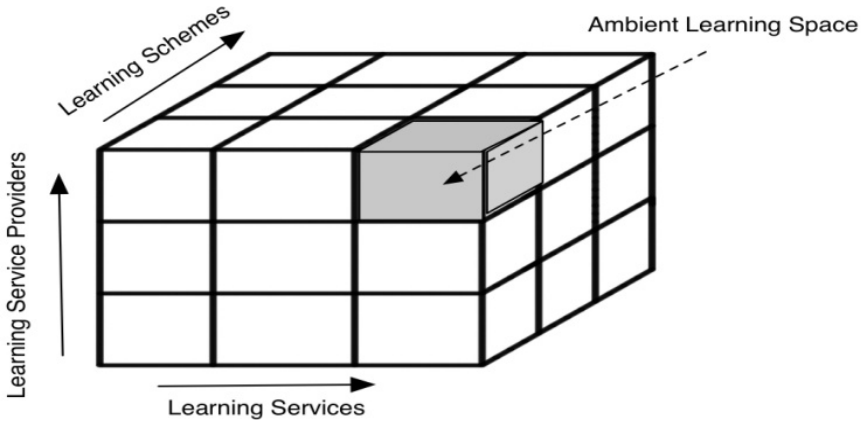
This work describes a novel monitoring approach that integrates the above key contributions. Essentially, we consider the dynamic characteristics of cloud SaaS services and allow monitoring of hybrid/composite SaaS services. The proposed approach also copes with the propagation effects of QoS violation (i.e. cascading effect) due to monitoring composite SaaS to identify/detect the services that violate contracted QoS terms. We implement monitoring processes for both the synchronous and asynchronous composition of SaaS services through analysing traces of events that are executed, and react to QoS violation through a proposed service substitution scheme. Finally, we illustrate a monitoring portal, which includes a dashboard to allow service providers to monitor multi-level SLA achievements, when a composite SaaS is monitored. This portal facilitates the process of troubleshooting and resolving problems with service performance regardless the providers.

This paper is organised as follows: first, we motivate further the problem through an illustrative scenario in Section 2. Then we reveal our monitoring architecture in Section 3. We then specify SLA specification of QoS levels associated with SaaS services in Section 4. An evaluation and an application are shown in Section 5 to evaluate and validate our proposed approach. Finally, Section 6 concludes the paper with a summary of results and some suggested future works.

2 Motivational scenario of monitoring SaaS services

In the context of Ubiquitous Learning (u-learning), instruction may be delivered across learning services following different quality schemes to match various learning contexts or personalised learner preferences. Typically, u-learning processes occur in an envisioned smart campus, which integrates three-dimensional ambient learning spaces together, as shown in Figure 1. Learning services, their cloud-based providers and the quality schemes along which these services are delivered control each Ambient Learning Space (ALS). Examples of ALS include technology-enhanced classrooms, labs and showcase areas. Our hypothetical smart campus features dedicated Cloud-based Learning Service Providers (CLSPs), which deliver their proprietary learning services through Learning Management Systems (LMSs). A CLSP uses an LMS to organise the distribution of its learning services, which could be in the form of virtual experiments, discussion forums, augmented reality illustrations, multimedia presentations, etc. Each CLSP registers its LMS into an institution-wide portal, which provides clients with a wider storehouse of learning services. A metaphoric description of this smart campus concept is a university digital-library portal, which provides site-wide access to various publishers' services (including e-books' readers, online labs, tutorials and articles).

Figure 1 Ubiquitous learning services



The proliferation of these services and their schemes induces local (i.e. within the ambient space) and global (i.e. within the smart campus) resources, which are contracted to meet predefined SLAs. To regulate these costs and manage QoS delivery of learning services locally and globally, a monitoring service is solicited via an institution-wide portal facility to autonomously optimise u-learning experiences in ALSs.

The networking infrastructure where CLSPs operate is subjected to high demands with increasing instances of accesses (given the smart campus large student population) to remote instruction in such a pervasive learning environment. However, each CLSP is expected to deliver learning services to all client learners or other CLSPs, involving possibly composite services, without causing any system overload or congestion (which may result in longer delays, beyond contracted SLAs). A learning scheme is a QoS-related concept, which we introduce in our illustrative smart campus model to classify the delivery options of learning services as follows:

- Continuous Access (CA),
- Delayed Continuous Access (DCA),
- Intermittent Guaranteed Access (IGA), and
- Intermittent Unguaranteed Access (IUA).

In a competitive and bandwidth-limited pervasive learning environment, CA and DCA would have higher priority over IGA and IUA schemes. In CA, an end user application can continuously consume a learning service until the learning objective is fully achieved, or the end user stops the learning process. Likewise, DCA also provides continuous learning service to an end user after an initial delay, provided that there is enough network resources to guarantee the needs for all end users requiring CA learning services within a given time period. In IGA scheme, the activity prescribed by a learning service is intermittent but an average supplied network resource within a time period is expected. Finally, in IUA scheme, no guarantees are required on the amount of networking resources to be supplied for the provided learning service. In this learning scheme, the end users are supplied with the learning service as the required networking resources become available, and may be pre-empted to advantage the other learners using the other schemes.

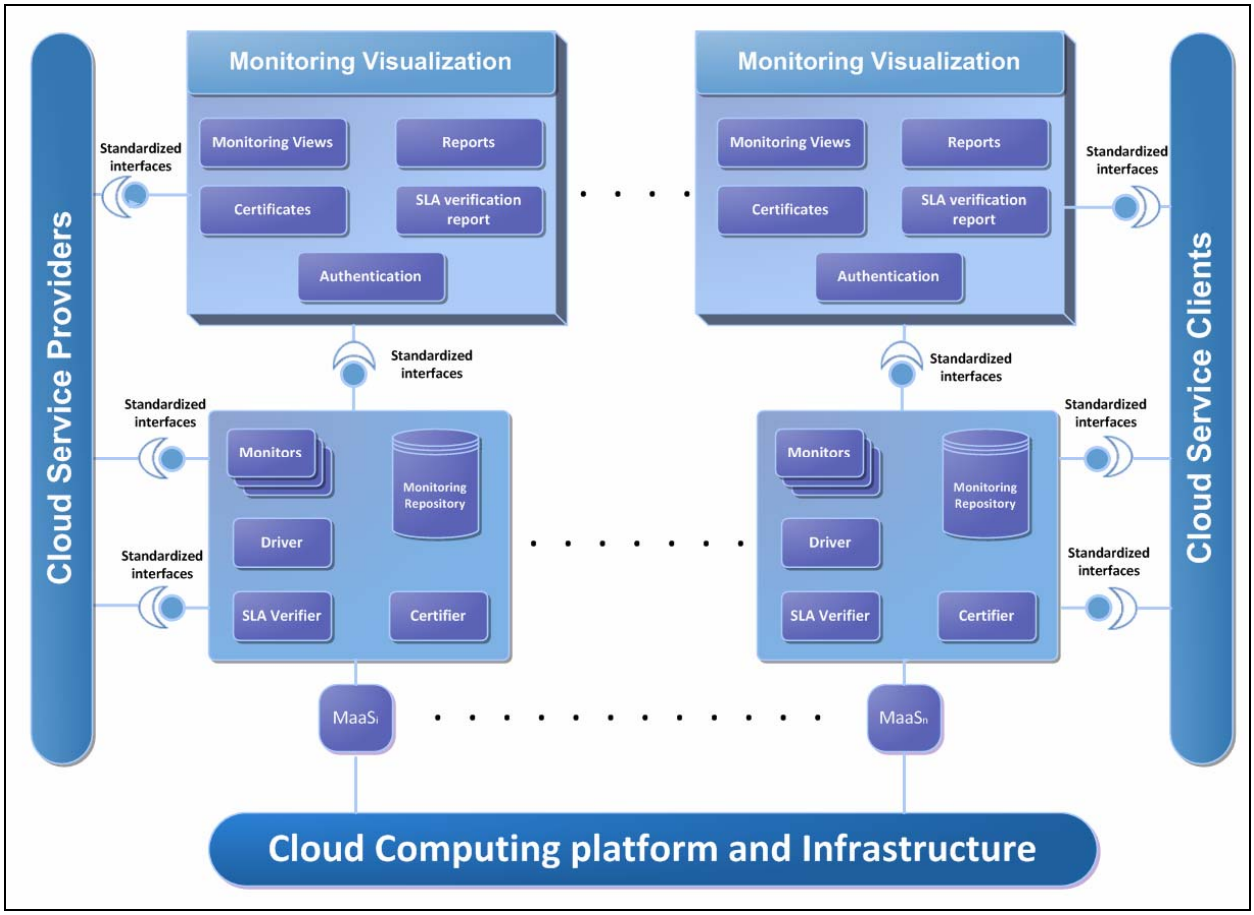
Learning schemes are advocated to match typical u-learning patterns' QoS requirements. For example, Continuous Access (CA) is matched with atomic learning, where corresponding learning services require uninterrupted exposure to learning objects (or learning atoms). This learning service type integrates a continuous learning process, which could involve a continuous media, played at an intended speed such as a video recording or a virtual-reality scenario, which may include some 3D modelling and visualisation. In this hypothetical scenario, the role of a monitoring entity, such as the one proposed in this paper, would be to ensure that the QoS levels associated with each of the above learning schemes are satisfied through appropriately provisioned learning services.

3 Monitoring architecture

In this section, we discuss the design of our cloud service monitoring architecture along with its main components. Figure 2 describes our independent monitoring architecture that supports and provides monitoring services to involved parties. The architecture involves a cloud platform and infrastructure services (e.g. application servers, virtual machines, storage servers), and monitoring entities; in addition to a set of Applications Programming Interfaces (APIs) to allow seamless communications between various architecture's components and with external entities to streamline monitoring activities.

For the monitoring scheme to achieve a higher performance guarantee, it is desirable to be deployed on top of a cloud infrastructure and may also use cloud platform services to benefit from key cloud properties, including high availability, reliability and scalability, while being flexible and self-reconfigurable. The monitoring entities exhibit two types of monitors: namely, local monitor and global monitor. When monitoring a single cloud service, one monitor sitting between the client and the provider is enough to analyse the interactions and report on the satisfaction of the SLA. However, when monitoring a composite cloud service, there is a need for many points of observation for an accurate analysis and decision making. In this schema, there is a global monitor responsible for interactions between the client and the composite cloud service as well as local monitors, each one between the composite cloud services and one of the composing cloud services. Local monitors report to the global monitoring whenever they observe an SLA violation. Moreover, the global monitor can pull information from local monitors for further investigation purposes.

Figure 2 Monitoring architecture (see online version for colours)



A user or manager, depending on the type of the cloud service that will be monitored, can trigger the monitoring activity. If a request is for monitoring a single SaaS, a local monitor is selected to observe the service. However, if a request is for monitoring a composite SaaS, a set of local monitors is used, each of which observes a single SaaS and reports monitoring results to the global monitor. The compiled monitoring results are then returned to the user or the manager.

The local and global monitors implement a set of modules (or APIs), each of which handles specific functionalities. The *Monitor* module for instance measures performances against indicators, detects violations as soon as they occur, and reports the accumulated information to concerned parties. However, the *SLA verifier* module, as shown in Figure 2, analyses the service agreement parameters (or thresholds) and verifies whether these parameters can be guaranteed prior to start the monitoring of a service. For this purpose, sets of verification tests are conducted to validate each QoS property value claimed in SLA. For example, a stress test is executed on a SaaS services under verification to measure its availability and the response time specified in SLA under different conditions. The role of *Certifier* module is to certify that a SaaS has passed the SLA verification tests, and then grants a certificate to the verified SaaS provider, accordingly. The certificate recognises service level trustworthiness.

The verifier and certifier entities in local and global monitors are similar; except in the global monitor where the verifier handles few extra verification tests involving more than one SaaS services. These tests are required to verify and certify each single SaaS service separately, in addition, to verifying and certifying the composite SaaS service as a whole. The *Driver* is the entity that initiates/triggers the monitoring process after passing all the necessary verification tests. This entity is the same for both local and global monitors as it is responsible only for initiating the monitoring process that encompasses, for instance setting up monitoring parameters, SLA contract and the monitoring period. The latter deals with a single point of contact, which is the SaaS service to be monitored (single or composite SaaS). The *Repository* is used to store all the information resulting from the monitoring operations (e.g. verification results and collected traces). The stored information will be used for visualisation and report generation, as illustrated by the *Visualisation* modules. Service clients/providers need to enter their credentials to access the visualisation information customised according to their need. This includes varying monitoring information such as monitoring views, reports, verification reports and generated certificates. As already stated earlier, this work focuses only on SaaS since other types of cloud services (i.e. IaaS, PaaS) are different and do require different monitoring set-up. Next, we describe the monitoring models we propose in a SaaS environment.

Our proposed models do not only handle a single SaaS monitoring but also address the challenges of monitoring a composite SaaS service made of aggregated SaaS services. Thus, two monitoring configurations are proposed hereafter

to handle both the situations. Figures 3 and 4 provide some extra monitoring details associated with the single monitor-based model and a multi-monitor-based model, respectively, and further complement the component detail of Figure 2.

Single monitor-based model: Figure 3 describes a model based on a single observer where a monitor is responsible of monitoring communication flows between a single SaaS and a single instance of a tenant application. The monitoring entity might conduct its activities and collect traces in any of the monitoring points as illustrated in Figure 3 (at SaaS side, at the tenant application side, or somewhere in between). The monitoring tenancy location might dictate the monitored parameters. For instance, to monitor SaaS response time, the monitoring activities should take place at the tenant side since it measures the user experience over time when using SaaS services. However, if the monitored metric is throughput, the monitor may intercept exchanged messages anywhere between the SaaS and the tenant application.

Figure 3 Single-monitor model (see online version for colours)

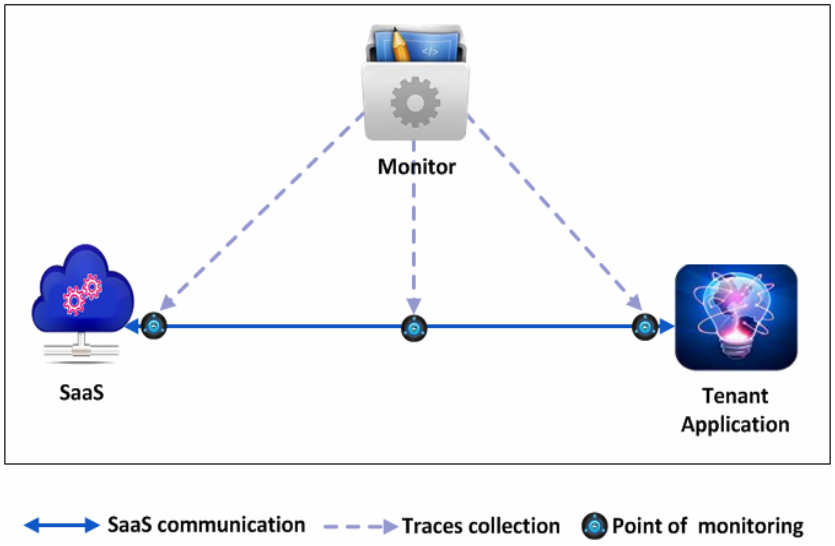
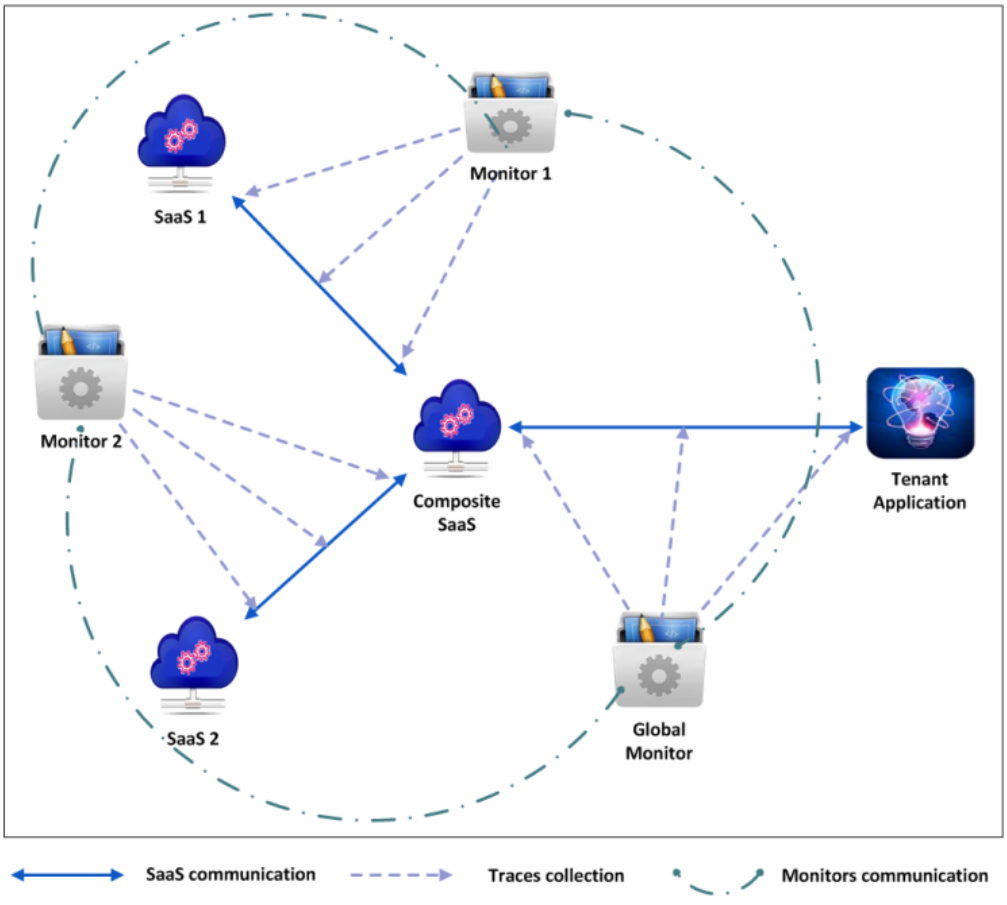


Figure 4 Multi-monitor mode (see online version for colours)



Multi-monitor-based model: Figure 4 describes a model where many monitors are used to monitor a composite SaaS made up of a couple of single SaaS services. A composite SaaS exhibits complex business logic and can be represented using a workflow description. GoogleDocs is an example of such a composite SaaS since it is made up of composing spreadsheets, presentation, and paint services in addition to Google Drive storage service. In this model, two types of monitors are deployed: a local monitor and a global monitor. Each local monitor is used to observe the communication flow between a composite SaaS and each of its composing SaaS module. However, a global monitor is used to observe communication flow between the tenant application and the composite SaaS module. Local monitor traces and monitoring reports are channelled to global monitors on a periodic basis. These reports and the overall monitoring activities are used to detect violations and related SaaS responsibilities. Similarly to the single monitor, a multi-monitor-based model can handle monitoring activities at the three points of monitoring locations, namely: SaaS side, tenant application side and somewhere in between. This is decided for instance based on the type of metric being monitored or the adopted monitoring implementation or even the optimisation level of the generated monitoring overhead.

Multi-monitor-based monitoring is used to observe composite cloud services that are aggregated from composing two or more single cloud services. These single cloud services might belong to the same cloud service provider or to different cloud providers. Therefore, multiple monitors are used to observe every single cloud service and report its performance to a global monitor that gather all monitoring information in order to ensure global performance of the composite cloud service.

The main difference between the Local Monitor (LM) and the Global Monitor (GM) is that the LM observes communications between single SaaS and the tenant application, whereas GL further collects traces between composite SaaS and the tenant application. GL uses traces from other LMs, corresponding to composing single SaaS services. GL performs more processing and analysis than LM, and thus requires more resources in terms of CPU, memory and storage capacities.

4 SLA specification and QoS description for SaaS

SLA is considered to be essential for any monitoring activity as it regulates the contract between parties involved in SaaS provisioning. Thus, monitoring conditions/QoS incorporated in the SLA should be respected and maintained over time. SLA should specify the guidelines used by monitors to conduct any monitoring session. Therefore, we describe hereafter the main SLA components that can be used to monitor cloud SaaS services.

SLA for cloud environment is yet to be standardised and accurately defined. Few initiatives have proposed a conceptual view of SLA for cloud services (Alhamad et al., 2010a; Torkashvan and Haghighi, 2012). Previous SLAs defined in the area of communication networks and web services are not suitable for cloud services. These suggest different requirements and challenges. In addition, cloud services are of different types (SaaS, PaaS, or IaaS) that require different QoS properties, and therefore different SLAs. As explained earlier, the focus of this paper is on SLA management of SaaS cloud services. A conceptual SLA framework proposed by Alhamad et al. (2010a) provides only a conceptual view and focuses mainly on non-functional properties of SaaS services. Torkashvan and Haghighi (2012) extended the IBM WSLA to create a new SLA specification in inter-cloud environments. They eventually proposed a scheme for SLA negotiation, and monitoring that are bound to cloud services' requirements. These initiatives do not provide the breakdown into functional properties and QoS properties. In addition, they do not consider managing cascading SLAs once a composite cloud service is monitored. In the next section, we provide a description of our SLA inspired from the previous models and adapted to consider key QoS properties only which we are planning to evaluate.

SaaS users have only access to the SaaS application and have no control over cloud resources (e.g. virtual machines, application development life cycle). Also, most of SaaS services are invoked via internet using web browsers. Therefore, QoS properties linked to this type of invocations are generally related to cloud service's performance, availability, cost and scalability. Table 1 describes QoS properties for both single SaaS user and providers, which they can agree upon.

Table 1 QoS Properties for SaaS

QoS	*Description*	*Requirements*
Response Time	Service delivery time or difference between the time of requesting a service and the time of receiving and answer.	Measured very frequently.
Scalability/ Elasticity	The ability to scale with the resources (memory, storage, processing) once the number of user increases.	Measures periodically or in demand.
Throughput	The number of requests a cloud SaaS can process per seconds.	Highly affected by the underlying network infrastructure.
Profitability	The return collected from using the services for a period of time.	Follow defined business model.
Configurability	The configuration time and complexity to use the service.	It is a subjective measure that depends on the expertise of the SaaS client.
Availability	The percentage of time during which the SaaS is up and responding, measured during a period of time.	Periodically measured.
Cost/billing	The price engaged in using the SaaS.	The cost is calculated per usage, size of data, or number of served requests.

Table 2 describes a couple of QoS schemes from Table 1 to be considered when monitoring a composite SaaS. The composite QoS is calculated by aggregating single SaaS QoS property values. The aggregation functions differ from QoS property to another. These might be a maximum, minimum, total or average of aggregated QoS values of single SaaS.

Table 2 QoS properties of composite SaaS

Composite QoS	*Description*	*Constraints*
Response Time (RT)	Maximum response time of all aggregated SaaS services.	In synchronous composition, a sum of response times of all single SaaS services is measured. If asynchronous composition, a maximum response time is measured.
Availability	Minimum availability of all aggregated SaaS services.	The availability is highly affected by the number of simultaneous client's access.
Throughput	Minimum throughput of all aggregated SaaS services.	The throughput is highly affected by the network capacity especially from client side; however, the cloud infrastructure should support a high throughput network.
Cost	Sum of costs of all aggregated SaaS services.	The cost is calculated based on a payment model, which may differ from a service to another (e.g. based on number of completed transactions, size of stored and processed data, or number of clients suing the service).

5 Evaluation and application

In this section, we describe our experimental set-up and the related cloud services, which we used to evaluate and validate our monitoring approach. The scenarios that we have developed are used to evaluate service monitoring of both single and composite cloud services. Next, we discuss the results of our experiments.

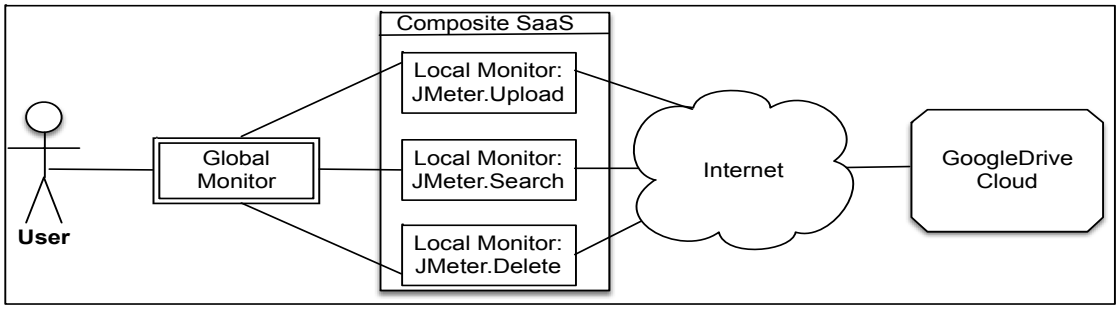
5.1 Experimental set-up

To evaluate our monitoring schemes, we used two examples of cloud services: (a) a single cloud service configuration and (b) a composite cloud service using the multi-monitor model configuration, as depicted in Figure 5. For a single cloud service, only one monitor is used; however, for a composite cloud service, both the composite SaaS and its composing SaaS are monitored using global monitor and local monitors, respectively. The set-up to monitor a single cloud service contains one monitor, which is implemented in this scenario as an instrumentation of the JMeter plug-in, as shown in Figure 5. In this scenario, we make use of a new composite service made up of single cloud services offered by Google Drive Cloud (Google, 2013). The decision to use Google Drive as a test environment was motivated by its abilities to offer a widely used free open source APIs, providing a flexible way to implement and test SaaS composition. Other cloud environments might be used to test our monitoring approach without impacting the overall actions and performance of monitoring scheme.

Using these single cloud services, a user can upload a file, search for a file or delete a file. These services are composed synchronously or asynchronously. Our composite service offers users a single interface to perform these three operations in one invocation, performed at the client side. We used the following software and hardware infrastructure configurations for our test environment:

- Desktop: Intel processor i7 3770 k at 3.7 Ghz, 32 GB 1600 Mhz DDR3, Windows 7 Pro x64.
- Eclipse IDE.
- Apache JMeter and JMeter Plugins (v1.0.0) (You et al., 2010): JMeter represents the local monitors.
- Global monitor: Java application that interacts with local observers (i.e. JMeter) and reports on the composite SaaS.
- Google Drive APIs V2.
- Google Drive Cloud Storage service/server.

Figure 5 Test bed configuration for multi-monitor model (see online version for colours)



5.2 Single and composite cloud SaaS

We have used the following cloud services to run the experiments of monitoring single and composite SaaS services. The initial step of using these Google Drive services involves three steps that are: (a) enable the drive API, (b) install the Google client library, and (c) run the instrumented SaaS client services.

5.3 Test scenarios

We have used the following scenarios to evaluate the schemes to monitor both single and composite cloud services. These scenarios use an SLA contract to monitor a couple of QoS properties, mainly response time and availability, and then we check if the monitor is able to detect any violations of these SLA agreements. For all the executed scenarios, a monitoring report is generated summarising the monitoring results and reporting violations on pre-agreed SLA properties, if any.

The threshold value for QoS properties related to response time and availability was decided experimentally. We executed many test scenarios considering different network configurations, different client environment set-up, different cloud resources, and different composition model used (synchronous versus asynchronous). We then selected the maximum value for the response time and the minimum value for the availability to be used as a threshold, for example, if an asynchronous invocation is used.

Scenario 1: the goal of this experiment is to evaluate single monitor abilities to handle varying clients' requests and measure corresponding QoS-related values for each request. We generated extensive storage requests to the single SaaS storage service and measured the average response time and availability, using different files sizes ranging from 0.5 MB to 10 MB. We ran the experiment many times, and at each instance we execute the single monitor to measure, detect and report any violation of response time and availability. The monitor uses threshold values for response time as described in the SLA and compares them to the measured values of these properties. The result of single monitoring is reported in a different period of time and average values of response time are reported in Figure 6.

Scenario 2: the goal of these experiments is to check if a single monitor is able to detect SLA contract violations using two different SLAs. These SLAs include, respectively, different threshold values (RT1, RT2) for each QoS property response time and availability. We run the monitoring experiments on the storage SaaS cloud service for a different period of time and report the results of monitoring (Figures 7 and 8).

Table 3 Single and composite SaaS description

SaaS	*Description*
	Single SaaS services
Upload	Read files from a local directory in the computer and upload them to Google Drive.
Search	Search the Google Drive for files with matching parameters. eg: file name contains the word 'cloud'
Delete	Permanently delete the files retrieved by search service.
	Composite SaaS services
Files_operations (Synchronous composition)	Composite synchronously the above three services. Invoke upload service, then search for the uploaded services using keywords, and finally delete the files retrieved by the search.
Files_operations (Asynchronous composition)	Composite asynchronously the above three services. Invoke in parallel upload service, search for the uploaded services using keywords, and delete the files retrieved by the search.

Figure 6 Average response time and availability of SaaS storage service (see online version for colours)

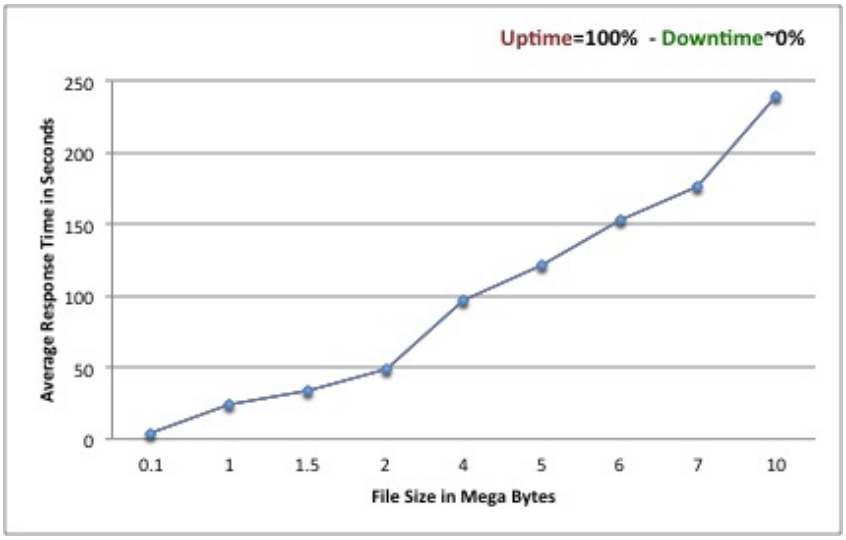


Figure 7 Monitoring SLA of a single SaaS using TR_1 (see online version for colours)

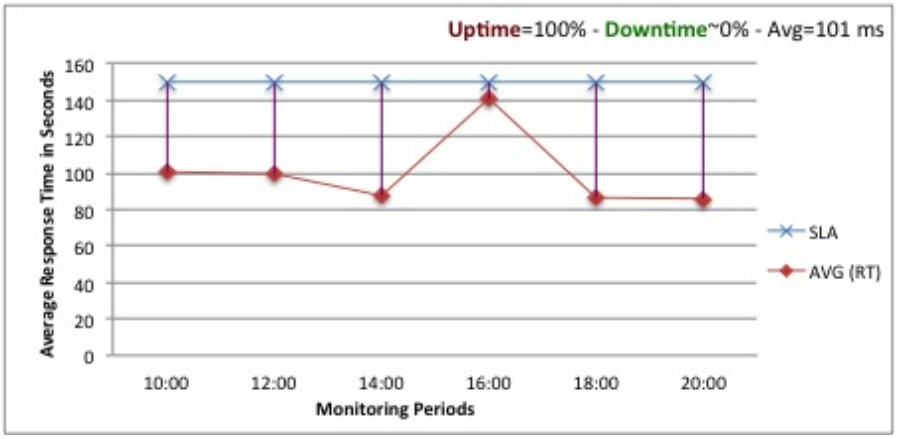
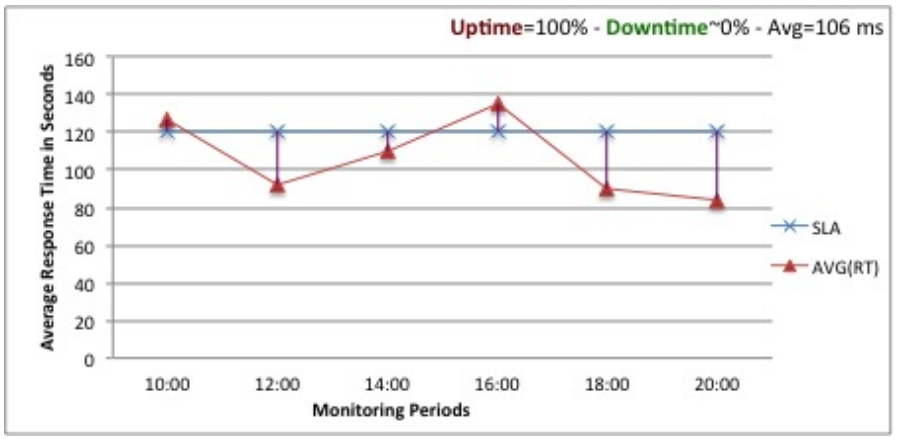


Figure 8 Monitoring SLA of a single SaaS using TR_2 (see online version for colours)



Scenario 3: the objective of this scenario is to test whether the global monitor is able to collect measured QoS of single SaaS retrieved from local monitors observing each single SaaS and calculating the aggregated QoS of the composite SaaS service. It also detects and reports any violations of response time and availability whenever they occur. The composition workflow relies on synchronous invocation of single SaaS to build the composite SaaS. We generated requests to the composite SaaS made up of three single SaaS that are upload, search and delete cloud services. A request consists of uploading a couple of files on Google Drive, searching for these files using a matching key word search, and then deleting these files from Google Drive. The result of these experiments is reported in Figure 9.

Figure 9 Monitoring response time of synchronous composite SaaS (see online version for colours)

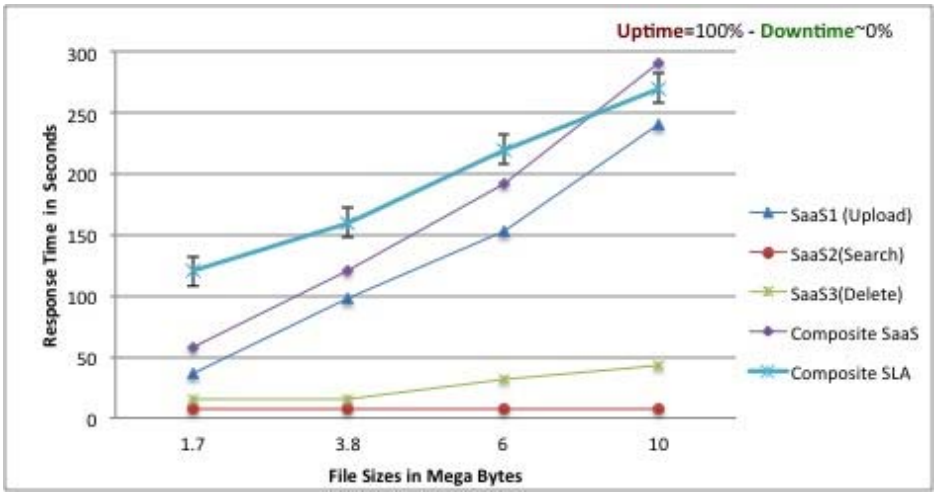
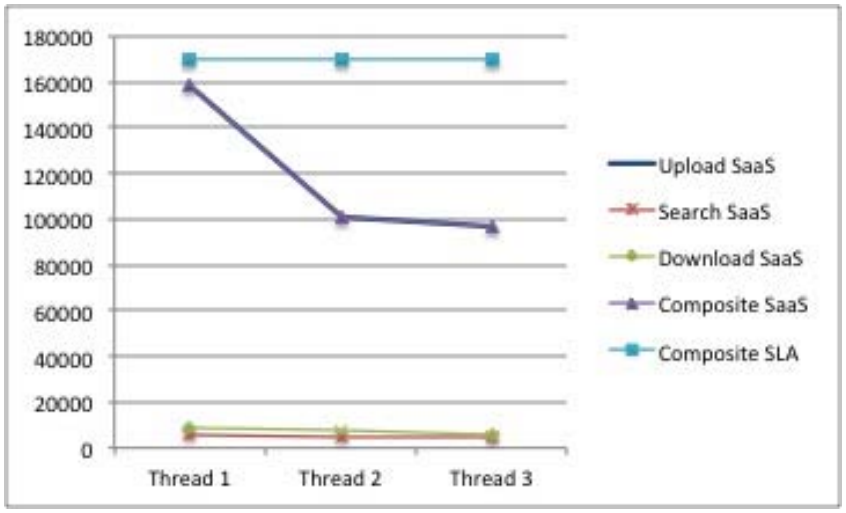


Figure 10 Monitoring response time of asynchronous composite SaaS (see online version for colours)



Scenario 4: in this scenario, we used the same set-up of scenario 3; the only difference is that the composition relies on an asynchronous invocation of single SaaS to build the composite SaaS. We generated three threads of requests to the composite SaaS made up of three SaaS that are upload, search and download cloud services. We instantiate the local monitors to measure the response time and availability of single services participating in the composition; and instantiate the global monitor to retrieve the measured QoS values, to analyse the collected traces and report violations of QoS if any. The result of these experiments is reported in Figure 10.

5.4 Results and discussion

The results reported from the above scenarios evaluate four main characteristics of our monitoring scheme: (a) ability to handle varying SaaS client's requests and measuring QoS for each of these requests, (b) ability to monitor different SLAs of different single SaaS and detect violations of QoS as soon as they occur, (c) ability to monitor a composite SaaS and measure the QoS of the composition, and (d) ability to monitor an asynchronous and synchronous composite SaaS and report violations of QoS as soon as they occur. Figure 6 illustrates the response time and availability for different requests sent to the cloud SaaS. These requests handled different files' sizes. We average this measure for each number of generated requests and file sizes as they are uploaded on the Google Drive. The graph clearly shows that the response time values increase proportionally with the size of the uploaded files. This proves that our single monitoring scheme maintains an accurate SaaS response time while scaling with the number of requests and size of handled files. The same figure also illustrates a stable value of service's availability, which is always maintained at 100% for all requests generated to the upload cloud service.

Figures 7 and 8 illustrate the monitoring results of a single SaaS conducted during a period of time. Figure 7 reports the measured values of response time and availability of SaaS. The monitored values of these two properties remain within the outer bound SLA thresholds. Thus, no violation has been detected. However, Figure 8 illustrates a situation where monitoring response time revealed two violations when an inner bound SLA is used. This proves that the single monitor has the ability to detect violations of QoS.

Figure 9 illustrates the monitoring results of a synchronous composite SaaS made of three single SaaS services that are upload, search and delete invoked sequentially. The graph shows that the response time increases proportionally to the size of handled files for each single SaaS as well as the calculated composite response time of the composite SaaS. Search and delete services exhibit lower response time than the upload service as the latter requires more processing hence more time. The composite response time showed higher values of response time since it is calculated as the total response time of all composing SaaS services because we are using synchronous compositions of SaaS services. With regard to the availability of composite SaaS, the measured availability value revealed a maximum availability of 100% over the monitoring period. The global monitor reports a violation of response time of the composite SaaS that is bounded by an SLA, as shown is Figure 9; this violation has been detected and reported after the uploaded file size exceeded is around 10 MB.

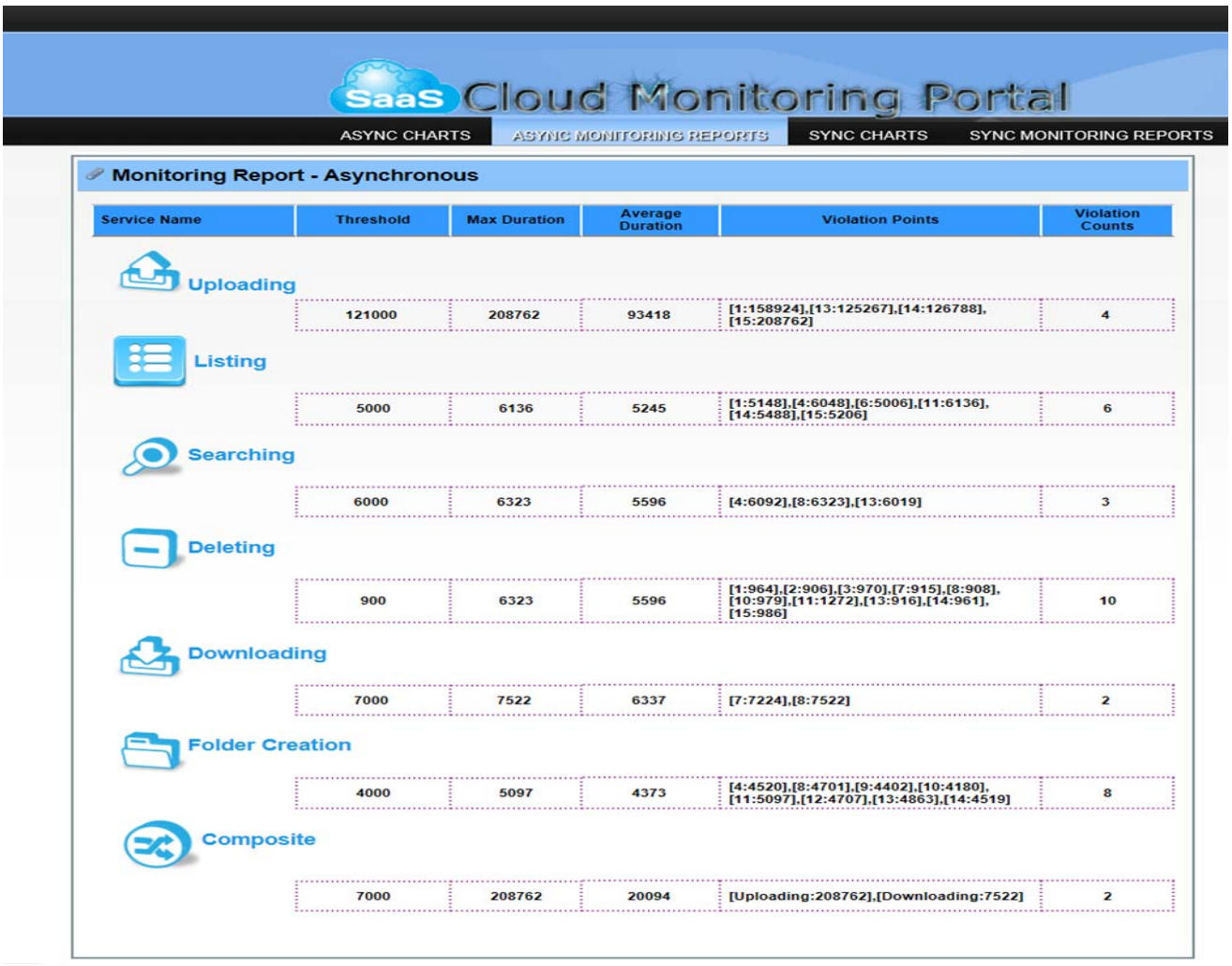
Figure 10 illustrates the monitoring results of an asynchronous composite SaaS made of three single SaaS that are upload, search and download invoked in parallel. We executed three threads each of which compose these three cloud services, and then the monitor measures the average response time for each composition thread. In asynchronous composition, the composite response time is the maximum response time among all the compositing services; in this scenario, the upload service. Therefore, composite SaaS response time value is equivalent to the response time of the upload service. As shown in the graph, the global monitor detects no violations, thus the cloud provider maintains the composite pre-agreed SLA.

5.5 Visualisation portal

In the following, we describe the visualisation module we have developed to report visualisation results. This module is a web portal that we developed using an open source portal development platform called Liferay (Sezov, 2012). Both clients and providers of cloud SaaS can access the monitoring portal. Each user is granted credentials to access the visualisation portal online. They can eventually view the monitoring data, graphs and reports of all monitoring activities that were undertaken on only their registered SaaS services, as shown in Figure 12. The portal includes two configurations: the first one is to report the results of monitoring synchronous composite SaaS and the second one to visualise monitoring results of asynchronous composite SaaS. The portal features dynamic readings collected in real time, and corresponding graphs are generated on the fly for each monitored SaaS. These graphs show a comparison of these readings against the contracted thresholds. For example, the first portlet in Figure 11 shows a table including the list of SaaS services considered for monitoring (e.g. uploading, folder creation, searching) and for each service the average response time measured over a different time period. The other portlets of the same figure show for each service from the list of SaaS services the corresponding graphs reporting the monitoring results along with their association with the threshold contracted in the SLA. The portal has also the additional feature of summarising the monitoring results into a compiled report, as shown in Figure 12. These reports include the monitoring results for each single SaaS, as well the results of composite SaaS and single SaaS services, which are responsible for the QoS violation. Monitoring results of a single SaaS include the threshold used, the maximum RT, the average RT, list of violation points and the number of violations. Monitoring results of composite SaaS include the same information reported for single SaaS in addition to the list of SaaS service(s) causing RT violations. This is a paramount feature that will be used to trigger adaption procedures based on the identified single SaaS sources of QoS violation.

Figure 11 Dynamic monitoring results generated from monitoring asynchronous composite SaaS (see online version for colours)

Figure 12 Monitoring report generated from monitoring asynchronous composite SaaS (see online version for colours)



6 Related work

Cloud computing is a relatively a new area. Hence, many research initiatives are concurrently tackling different aspects of cloud computing including data and service security, legal and privacy issues, cloud services management, and monitoring.

With regard to cloud service management and monitoring, which is the scope of this paper, a couple of research works have been proposed and focused on a diversity of service management aspects. The most comprehensive one proposed by Meng and Liu (2012) describes a monitoring scheme combining both the state monitoring capabilities and the performance-enhancement functionalities of cloud services. However, this work ignores the autonomic nature of monitoring and its significance in enforcing pre-agreed SLAs. It overlooks how a monitoring entity can handle the heterogeneity of different cloud services mainly SaaS, PaaS and IaaS. Shicong et al. (2012) used state monitoring approach to solve the problem of impaired communication caused by dynamics around different nodes. Therefore, they proposed to expose and handle communication dynamics such as message delay and loss in cloud-monitoring environments.

Cicotti et al. (2012) propose a QoS monitoring as a service that is built on top of the SRT-15 (the SRT-15 Research Project, http://srt-15.unine.ch), a cloud-oriented and Complex Event Processing (CEP) based platform. In particular, they presented the main components of the monitoring architecture such as SLA analyser, violation certifier and KPIs meter and illustrate operation and internals with respect to a case study related to an Internet of Things (IoT) application.

Cloudkick (2014) is proposed to include a centralised cloud server-monitoring tool for multiple cloud server providers in addition to dynamic server management tools. A dashboard allows a cloud user to monitor its cloud servers and list all the measured performance metrics.

An important category of research focused on cloud SLA definition, guarantee and security enforcement (Chazalet, 2010; Alhamad et al., 2010a; Alhamad et al., 2010b; de Chaves et al., 2010). Chazalet (2010) proposes a separation of concern regarding probes, monitoring information collection and contracts compliance. Alhamad et al. (2010a) present the main criteria that should be considered at the stage of designing the SLA in cloud computing. Alhamad et al. (2010b) and de Chaves et al. (2010) address some issues in cloud security. Alhamad et al.

(2010b) define a trust model to be used in the cloud and de Chaves et al. (2010) attempt to provide an overview on security and the difficulties faced during the process to define some security metrics.

Wang En et al. (2013) propose a physical resources-oriented model that aims at insuring QoS in cloud services mainly confirming an always on availability. Morariu et al. (2013) present a cloud-monitoring solution made up of several open source modules. This solution is platform-dependent, as it requires two proprietary platforms, namely: IBM CloudBurst 2.1 and IBM TSAM. The solution can only be used by services' providers and can monitor physical resources (e.g. CPU, memory) as well as HTTP and J2ME load. Lin et al. (2013) present another platform-dependent monitoring solution based on the Apache CloudStack. This solution can monitor virtual machines, primary and secondary storages, and management systems. Monitoring data is scattered over distributed databases so it can be available to other modules. Kamel et al. (2013) propose an approach to converge to a precise QoS model based on collected delay measurements of interactions between mobile clients and cloud providers. Violations of QoS are then detected against this QoS model. QoS-MONaaS monitoring solution proposed by Adinolfi et al. (2012) is based on the SRT-15 cloud platform to monitor QoS at the business process level. As a part of the STR-15 project, this architecture is platform-dependent and porting to other platforms requires adaptation.

Xiao et al. (2011) propose a framework for QoS specification, QoS-aware service selection, QoS monitoring, and QoS violations handling. The paper does not detail technical aspects of the monitoring activities. Mdhaffar et al. (2011) propose a platform based on an aspect-oriented programming. It focuses on fault tolerance and uses aspect-oriented programming to reduce the required overhead. Ferretti et al. (2010) present a load-balancer platform to support QoS in cloud services. In case of QoS violations, the platform offers dynamic reconfiguration to satisfy the specified QoS. Ma et al. (2013) describe a lightweight framework for IaaS and SaaS parameter measurements at both physical and virtual machine instances. The framework is module-centralised and based on plug-in bundles.

All the above works exhibit limitations that are summarised as follows:

- Highly depend on the underlying infrastructure proposed for monitoring cloud services.
- Not transparent enough to cope with dynamic nature of cloud services (e.g. elasticity and scalability).
- Do not consider monitoring of composite cloud services.
- Do not adapt to various cloud services' characteristics (e.g. self-service provisioning).
- Do not evaluate the generated monitoring overhead form messages required to conduct monitoring.

Our proposed framework addresses some of these limitations and extends the above works to suggest an independent and adaptive monitoring service infrastructure. The latter should offer a flexible monitoring service to assess QoS performance indicators of consumed services as well as any QoS violations that might be detected. Our monitoring approach entails benefits for both service consumers and providers. On one the hand, service consumers will be assisted to decide on shifting to another service provider, given the frequency and severity of detected QoS violations. On the other hand, service providers will be having their services monitored to allow attracting a higher number of clients, and gaining a landmark reputation in the marketplace. The monitoring scheme is expected to provide a flexible, yet adaptable solution to various cloud services. It also allows a monitoring of composite cloud service that is made up of various other cloud services offered by one or more cloud provider. Therefore, monitors adjust their interfaces, points of control and observation, and monitoring operations to the type of service being monitored and to different cloud services.

7 Conclusion

Previous works on monitoring cloud services are limited by proprietary infrastructure, scalability and scope of monitored service types. In this paper, we proposed an approach to monitor SaaS services and report QoS violations as soon as they occur. Our platform-independent approach monitors both single and composite SaaS services to ensure SLA terms are met. Consequently, two monitoring configurations have been proposed to handle both situations: the single- and multi-monitor-based models to observe both single and aggregated services. Traces collected from these monitors are used to compare against the SLA-derived values, based on which violation alerts are triggered. A display is proposed to support the detection of these alerts in a dashboard-like visualisation process. A series of experiments have been conducted to evaluate the proposed monitoring models to evaluate their capability to detect QoS violations in single SaaS monitoring and composite SaaS monitoring schemes. The obtained results show that our monitoring schemes are indeed able to detect QoS violations based on QoS performance bounds used in SLA specifications. As a future work we are planning to extend our solution to cover adaptation when violation of SLA is detected. Adaptation can take different autonomic actions including service replacement, SLA renegotiation and/or SLA contract cancellation.

References

Adinolfi, O., Cristaldi, R., Coppolino, L. and Romano, L. (2012) 'QoS-MONaaS: a portable architecture for QoS monitoring in the cloud', *8th International Conference on Signal Image Technology and Internet Based Systems (SITIS)*, pp.527–532.

Alexander, K. and Heiko, L. (2002) *The WSLA Framework: Specifying and Monitoring Service Level Agreements for Web Services*, IBM Research Reports, New York.

Alhamad, M., Dillon, T. and Chang, E. (2010a) 'Conceptual SLA framework for cloud computing', *4th IEEE International Conference on Digital Ecosystems and Technologies (DEST)*, IEEE, pp.606–610.

Alhamad, M., Dillon, T. and Chang, E. (2010b) 'SLA-based trust model for cloud computing', *13th International Conference on Network-Based Information Systems (NBiS)*, IEEE, pp.321–324.

Armbrust, M., Fox, A., Griffith, R., Joseph, A.D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A. and Stoica, I. (2010) 'A view of cloud computing', *Communications of the ACM*, Vol. 53, pp.50–58.

Buyya, R., Broberg, J. and Goscinski, A.M. (2010) *Cloud Computing Principles and Paradigms*, Wiley, Hoboken, NJ, USA.

Buyya, R., Chee Shin, Y. and Venugopal, S. (2008) 'Market-oriented cloud computing: vision, hype, and reality for delivering IT services as computing utilities', *10th IEEE International Conference on High Performance Computing and Communications (HPCC'08)*, pp.5–13.

Chazalet, A. (2010) 'Service level agreements compliance checking in the cloud computing: architectural pattern, prototype, and validation', *5th International Conference on Software Engineering Advances (ICSEA)*, IEEE, pp.184–189.

Cicotti, G., Coppolino, L., Cristaldi, R., D'Antonio, S. and Romano, L. (2012) 'QoS monitoring in a cloud services environment: the SRT-15 approach', *Parallel Processing Workshops, Lecture Notes in Computer Science*, Vol. 7155, pp.15–24.

CloudKick (2014) *Cloud Monitoring and Management*. Available online at: http:/http://www.cloudkick.com

de Chaves, S.A., Westphall, C.B. and Lamin, F.R. (2010) 'SLA perspective in security management for cloud computing', *6th International Conference on Networking and Services (ICNS)*. IEEE, pp.212–217.

Ferretti, S., Ghini, V., Panzieri, F., Pellegrini, M. and Turrini, E. (2010) 'QoS-aware clouds', *IEEE 3rd International Conference on Cloud Computing (CLOUD)*, pp.321–328.

Google (2013) *Google Drive*. Available online at: https://drive.google.com/

Kamel, A., Al-Fuqaha, A., Kountanis, D. and Khalil, I. (2013) 'Towards a client-side QoS monitoring and assessment using generalized Pareto distribution in a cloud-based environment', *IEEE Wireless Communications and Networking Conference Workshops (WCNCW)*, pp.123–128.

Lin, K., Tong, W., Zhang, L. and Hu, C. (2013) 'SCM: a design and implementation of monitoring system for cloudstack', *International Conference on Cloud and Service Computing (CSC)*, pp.146–151.

Ma, K., Sun, R. and Abraham, A. (2013) 'Toward a module-centralized and aspect-oriented monitoring framework in clouds', *Journal of Universal Computer Science*, Vol. 19, pp.2241–2265.

Mdhaffar, A., Ben Halima, R., Juhnke, E., Jmaiel, M. and Freisleben, B. (2011) 'AOP4CSM: an aspect-oriented programming approach for cloud service monitoring', *11th IEEE International Conference on Computer and Information Technology (CIT)*, pp.363–370.

Meng, S. and Liu, L. (2012) 'Enhanced monitoring-as-a-service for effective cloud management', *IEEE Transactions on Computers*, pp.1–14.

Morariu, O., Borangiu, T. and Morariu, C. (2013) 'Multi-layer QoS monitoring in private clouds', *24th International Workshop on Database and Expert Systems Applications (DEXA)*, pp.236–240.

Rimal, B.P., Eunmi, C. and Lumb, I. (2009) 'A taxonomy and survey of cloud computing systems', *5th International Joint Conference on INC, IMS and IDC (NCM'09)*, pp.44–51.

Sezov, R. (2012) *Liferay in Action: the Official Guide to Liferay Portal Development*, Shelter Island, NY, Manning.

Shicong, M., Iyengar, A.K., Rouvellou, I.M., Ling, L., Kisung, L., Palanisamy, B. and Yuzhe, T. (2012) 'Reliable state monitoring in cloud data centers', *IEEE 5th International Conference on Cloud Computing (CLOUD)*, pp.951–958.

Torkashvan, M. and Haghighi, H. (2012) 'CSLAM: a framework for cloud service level agreement management based on WSLA', *6th International Symposium on Telecommunications (IST)*, pp.577–585.

Wang En, D., Wu, N. and Li, X. (2013) 'QoS-oriented monitoring model of cloud computing resources availability', *5th International Conference on Computational and Information Sciences (ICCIS)*, pp.1537–1540.

Xiao, L., Yun, Y., Dong, Y., Gaofeng, Z., Wenhao, L. and Dahai, C. (2011) 'A generic QoS framework for cloud workflow systems', *IEEE 9th International Conference on Dependable, Autonomic and Secure Computing (DASC)*, pp.713–720.

You, J., Zhang, L., Wang, H., Sun, Y. and Cao, G. (2010) 'JMeter-based aging simulation of computing system', *International Conference on Computer, Mechatronics, Control and Electronic Engineering (CMCE)*, pp.282–285.