Customized SLAs In Cloud Environments

Nikoletta Mavrogeorgi[1], Vassilios Alexandrou[1], Spyridon Gogouvitis[1], Athanasios Voulodimos[1], Dimosthenis Kiriazis[1], Theodora Varvarigou[1], Elliot K. Kolodner[2]
[1]Department of Electrical and Computer Engineering, National Technical University of Athens, Athens, Greece
[2]IBM Haifa Research Labs, Haifa University, Mt. Carmel, Haifa, Israel
{ nikimav, alexv, spyrosg, thanosv, dimos }@mail.ntua.gr, dora@telecom.ntua.gr, kolodner@il.ibm.com

Abstract— **Nowadays, the demand for online storage is increasing. Customers store their data and can retrieve them anytime and from anywhere without having to care about back up and data recovery. SLAs between Cloud providers and customers determine the level of service guaranteed by the former. A customer can be anyone from an individual person to enterprises. Customers have different needs and preferences. Our proposed system permits the creation and enforcement of customized SLAs, where customers choose the requirements that they need fulfilled. The proposed SLAs are associated with content terms pertaining to the nature of the content of the data stored. SLA enforcement is based on rules that are updated in runtime in order to proactively detect possible SLA violations and handle them in an appropriate manner.**

Keywords-Cloud; SLA schema; SLA Management; content centric storage; policies; proactive SLA violation detection

I. INTRODUCTION

Nowadays, the evolution of IT technologies permits the provision of very interesting and useful services. Cloud storage is one of the major trends in the IT world. It allows users to store their data easily anytime from anywhere and at a low cost, while also being able to retrieve them very fast. The Cloud platforms take care of the infrastructure, the hardware needs and management tasks, such as backup and recovery. All these are abstracted to users. Users only upload and retrieve their files. They don't need any more to care about management tasks or to own demanding hardware.

Anyone can be a Cloud user: a person storing their private data e.g. photos, music, lessons etc., or big organizations and companies. Healthcare, media, telecommunication and enterprises are some of the domains that can take advantage of the Cloud. Some indicative examples can be the following: In healthcare, records regarding patient data can be stored. Each hospital has its own records. Data for a patient can be exchanged through hospitals for having the complete history of the patient. Also, doctors can exchange their medical opinion e.g. for a cardiogram. Similar, as far as telecommunications are concerned, services regarding phone calls can be used, such as a call center service for bank transactions. With regard to media, a journalist can upload his article and provide news to people. Exchange of comments and more information, for instance more images about an event, can be realized in runtime. Enterprises can organize their data and store it according to the department of the enterprise and its location. Roles are assigned and the employees according to their work position in the enterprise will have access only to a specific part of the data.

A contract should be signed in order for these promises to be guaranteed. Authentication and authorization are also necessary. A Service Level Agreement (SLA) is a contract between the customer and the service provider, where the terms and conditions of the offered service are agreed upon.

Currently, many Cloud providers exist and support various facilities with a corresponding cost. Federation provides useful services in the Cloud environments. There is the resource federation where a Cloud provider can use the resources of another Cloud provider, which is hidden from the client. In this case an SLA exists between the Cloud providers. Furthermore, Cloud users may find a most suitable Cloud provider according to their desires. So, a user that will use a new Cloud provider may want to move his data from the old Cloud to the new Cloud. This operation is called on-boarding federation.

In this paper we present an SLA schema which is enriched in comparison to the classic ones. It is based on content terms and contains various sections such as federation determination.

The SLAs are managed by the SLA Management framework. The SLA Management is responsible for the SLA negotiation and the enforcement and maintenance of the SLAs. During the SLA lifecycle, the SLA Management framework checks if the requirements are met based on the monitoring data analysis. Additionally, it detects proactively SLA violations and makes reactive actions for avoiding them. Furthermore, it provides to the users reports and notifications related to the signed SLA, and it advises on updating the SLA in order to gain benefits.

The proposed SLA Management is developed under the EU-funded project VISION Cloud [1],[2],[18]. The goal of VISION Cloud is to support content centric storage and is based on metadata support. The basic storage unit is the container. A container can contain many objects. The number and the location of the replicas are determined in container level. One object has as many replicas as the owning container determines.

Tenants and users constitute the account model. The tenant signs an SLA and creates users with specific roles and access to their data. A container creation is requested by a user and the container is created and stored in the Cloud with the replicas that are calculated according to the SLA.

An innovation of VISION Cloud is the storlet support.

978-0-7695-5094-7/13 $31.00 © 2013 IEEE
DOI 10.1109/3PGCIC.2013.45

Storlet is an executable which provides capabilities for supporting and improving the services that are offered to the tenants, such as data compression, file transformation in various formats, etc. It contains a trigger and a handler. The trigger is the condition which, when fulfilled, invokes the execution of the handler. Some examples of storlets are the speech to text, the text to pdf, the translation to a specific language, a SMS notification etc.

In this paper we present an automated SLA Management mechanism. In Section II the related work is overviewed. Section III describes the SLA and its additions. The next sections demonstrate the proposed SLA schema and the SLA Management. Finally, an example and the conclusions are displayed.

II. Related Work

A lot of research and protocols have been created as far as Service Level Agreements (SLA) and SLA Management are concerned

SLA schemas are XML schemas that represent the content of an SLA. Some existing approaches for SLA schemas and the corresponding languages to define service description terms are: SLAng [5], WS-Agreement [9], WSLA[7], WSOL[6], and SWAPS [8].

However, weaknesses exist. SWAPS [8] is quite complex and the implementation is not publicly available. WSLA [7] and SLAng [5] have not further development at least since 2009. Apart from this, SLAng does not permit to define management information such as financial terms and WSLA has not formal definition of metrics semantics. WSOL [6] lacks SLA related functionalities, such as the capture of the relationship between service provider and infrastructure provider.

The WS-Agreement (Web Services Agreement) [9] is a Web Services protocol for establishing agreement between two parties using an extensible XML language for specifying the nature of the agreement, and agreement templates to facilitate discovery of compatible agreement parties. It allows arbitrary term languages to be plugged-in for creating domain-specific service description terms.

Many projects contain SLA Management to achieve their goals, such as IRMOS, RESERVOIR, SLA@SOI, SOA4ALL and SLA4D-Grid. Most of them create SLAs based on WS-Agreement or a protocol close to WS-Agreements.

For the aforementioned reasons, in our proposed framework we based on the WS-Agreement and the WS-Agreement Negotiation which are widespread and increase the level of interoperability.

Two challenges research issues are the requirements translation from high level metrics to low level requirements and vice versa [10],[16] and the proactive violation detection. Many proposals have been done for these issues, but very little for Cloud environments. For instance, GRIA SLAs[13] suggest a solution for avoiding violations but concerns only Grid environments.

The LoM2HiS framework [10] proposes the translation of low level metrics to high level terms that are used in Cloud SLAs, but not the reverse translation. Also, they are based on generic characteristics and terms (e.g. availability) which are not application specific.

The LAYSI framework [11] supports two kinds of monitors sensors, the host monitor and the runtime monitor sensor. The latter senses future SLA violation threat based on resource usage experiences and predefined threat thresholds.

In DesVi [12], an architecture is proposed for preventing SLA violations based on knowledge database and case-based reasoning [14]. It also uses the LoM2HiS framework for the requirements translation.

III. SLA

The SLA is the service level agreement between the tenant and the service provider, where the level of a service is formally defined. The SLA is created after a successful SLA negotiation between the tenant and the provider. In the negotiation process, SLA templates are used which provide the choices that the tenant can have according to current supplies of the Cloud.

Some of the elements that are defined in the SLAs are the parties involved, the contract date, the agreement terms and the cost data. The properties that are used in the agreement terms can be divided in functional and non-functional. The latter can contain quantitative (e.g. availability, durability, latency) and qualitative properties. The quantitative properties are constrained through thresholds, commonly named as Service Level Objectives (SLOs).

The proposed SLAs are enriched versions of the traditional ones [4]. Apart from determining the classic elements such as the parties involved, the SLOs and the cost rules, the SLAs include content term determination of the objects that will be associated with this SLA. This fact permits the Cloud to provide the users with content-centric services. The content is linked with performance estimates, decisions for moving computation close to storage, pricing models etc having as a result to provide high performance services, such as quicker search and retrieval of the objects or high performance video streaming speed. Some examples of content terms are telecommunication, media, healthcare, enterprise. Hierarchy of content terms exists. For instance an article for daily news inherits the content term media.

Different SLA templates are given according to the content term selection. For instance in a telecommunication SLA template there are storlets useful for telecommunication services like text to speech, translation, pdf to text etc. and the containers that are created having this SLA associated are stored in data centers that provide efficient telecommunication services. Similarly, the media SLA templates contain advanced data and services regarding media and the created containers are stored in data centers that provide efficiently video related services, such as video streaming. Articles for the daily news which are accessed frequently on a daily basis should be stored in many geographic places, in order to have a quicker access.

Furthermore, specific actions can be executed depending on the SLA content related term, such as storage at specific data centers, execution of compression or format transformation of an object. For instance, if a user requests to store a pdf translated in another language, then the pdf file is stored in the Cloud it automatically gets transformed to a text file and the translation takes place. The file is stored in a data center which contains a storlet for the translation.

The proposed SLA schema contains additional sections for declaring requirements regarding federation and storlets, and renegotiation constraints.

An SLA template contains a section for federation. The user determines if he desires federation or not. If he does, he selects the Cloud providers from which he wants to federate data. Also, he selects requirements like the federation estimation transfer completion time depending on the objects size, if available. Finally, the associated cost is also specified.

Storlets that will be used are also agreed during the SLA negotiation. Some storlets in order to be executed need some minimum hardware requirements as the memory. The SLA contains a section regarding storlets. The tenant during the SLA negotiation selects which storlets wants to use and for each storlet the requirements and the cost agreed. Some requirements examples are hardware requirements, execution time, maximum number of storlet executions for a specific period time etc. For instance, a tenant may be interested in selecting speech to text storlet with requirement execution time less than 500 ms per 100 words.

The proposed system permits the tenant to renegotiate his SLA.. During the initial SLA negotiation, the constraints that can be renegotiated are defined. Apart from the classical requirements, renegotiation is permitted for changing federation and for storlets updates.

Finally, customized SLAs are provided. The tenant can create SLAs according to his desires and the SLAs are not restricted just to the availability as most Cloud providers currently support. During the SLA Negotiation a list of requirements is provided. Each requirement is associated with different levels. The higher is the level, the higher is the cost. The tenant can choose whichever requirements he/she desires. Each requirement has different levels (for instance high, medium and slow availability) and the tenant selects for each requirement the level that better fits to his needs. For instance, he may care about availability and throughput, and the availability to be very important for him so to choose the highest level whereas for the throughput to be satisfied with the medium level and to choose this one. Additionally, the tenant can have different SLA for each container. For instance, he can have two containers, one for media and one for papers and for the former to sign an SLA with content term ‘media’ which provides him streaming services, whereas for the latter to chose a low-cost SLA as he doesn’t need to access frequently the papers.

IV. SLA Schema

In the proposed approach, the Service Level Agreement is encapsulated in an SLA schema [4]. The basic element is the SLA element, which defines what data can be contained in an SLA (e.g. cost, contract dates, user requirements, fines, etc) and in which format. The language that is used for the SLA schema is XSD (XML Schema Document). In the proposed schema the WS-Agreement was used. The main additions are the content term determination and the sections regarding federation and storlets.

A. *Basic elements*

The SLA element is the outermost element, which encapsulates the entire SLA. An SLA is created by filling an SLA template. The SLA Template contains a section for determining the content term that will concern the SLA.

The proposed SLA schema consists of the following basic elements:

- SLA: this element represents the SLA.
- SLATemplate: this element represents the SLA template, that is, the template that the user should fill out in order to create an SLA.
- ContentTerm: this element represents the content term that concerns the SLA (e.g. media, enterprise, healthcare, telco, video, scientific paper etc.).
- TermAttribute: this element represents the QoS metrics (e.g. availability, jitter, ingest rate, geographic constraints etc.).
- Requirement: this element represents the SLOs that the user poses. The requirements are an expression of the term attributes of the concerned SLA content term. Assuming that the user wants to use the termAttribute "availability", then an expression could be: "availability>0.999". (Note: the requirements must be something that can be measured).
- Responsibility: this element represents the responsibilities that the user or the provider should have (e.g. the provider should send him monthly reports and notify the user for an SLA violation).
- Cost: this element represents the cost (e.g. the user should pay 10 cents per 1GB upload).
- Penalty: this element represents the fines that the provider should pay in case the user's requirements are violated.

B. *SLA properties*

An SLA contains, based on the WS-Agreement, a name element, a context element and a terms element:

- Name
    - id: the SLA's identifier Context (contains metadata)
- Context (SLA metadata)
    - providerId: the identifier of the provider
    - customerId: the identifier of the customer (user)
    - dates: the contract data and the date from which and until which the SLA is valid
    - templateID: the id of the template that was used
- Terms (main body of the agreement)
    - contentTermId: the identifier of the content term that concerns this SLA
    - requirements: the SLOs that the user poses
    - federation: the federation agreement of the user.

- storlets: the storlets that the user desires associated with the requirements
- responsibilities: the responsibilities that the customer or the provider should have
- cost: the billing rules
- penalties: the fines that the provider should pay in case that the customer's requirements are violated
- changesAgreements: the rules that are signed in case that the customer or the provider desires to change some data from the signed SLA during the SLA lifecycle (used during the SLA renegotiation)

C. Content term properties

The basic properties that compose the Content Term element are the following:

- id: the identifier of this content term
- name: the name of the content term
- description: the description of the content term
- terms: the associated metrics (e.g. ingest rate) with this content term
- services: the associated services with this content term

The content terms support inheritance. Therefore, a content term can be an extension of another. There are general term attributes that are inherited by all content terms, such as availability, durability, geographic location constraints, etc.

V. SLA MANAGEMENT

A. Description

The SLAs are managed by the SLA Management framework, which is responsible for the SLA negotiation and the SLA enforcement and maintenance. During the SLA lifecycle, the SLA Management framework checks if the SLA requirements are met and sends to the user reports and notifications related to the signed SLAs.

B. Architecture

The architecture of the SLA Management is depicted in Figure 1. In VISION Cloud there are four layers the Data Interface Layer (DIL), the Data Operating Layer (DOL), the Management Interface Layer (MIL) and the Management Operating Layer (MOL). SLA Management belongs to management layer. At MIL it handles the operations that are supported to the users and the corresponding component is the SLA Negotiator, whereas at MOL there are the operations for the internal functionality of VISION Cloud and the corresponding component is the SLA Enforcer.

The basic components of the SLA Management are the following:

Templates Generator. This component is responsible for generating SLA templates. It creates SLA templates based on the capabilities that the Cloud can provide and on content terms.

Negotiator. The Negotiator is responsible for the SLA negotiation between the user and the provider. The Negotiator gets the SLA templates from the Templates Generator and asks the tenant to choose and fill a template. When an SLA is signed, it informs the interactive systems of the created SLAs (e.g. Accounting and Billing, Global View, SLA Configurator). Also, it receives notifications from other components during the evaluation and informs the user with reports and notifications (e.g. for SLA violation, for the current cost, etc.).

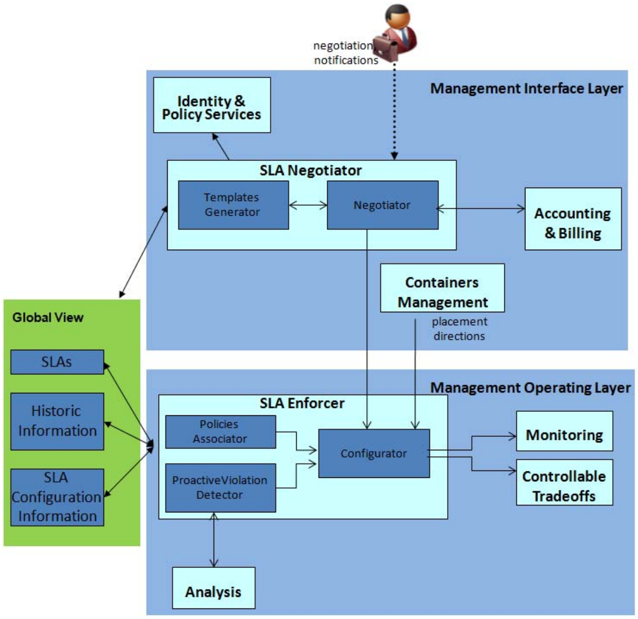


Figure 1: SLA Management architecture

Policies Associator. This Policies Associator is responsible for deciding which policies should be followed as far as an SLA is concerned. It links the policies provided by the administrators with the current SLA and provides them to the Configurator.

Configurator. This component is responsible for creating the configuration that concerns a created SLA. It receives information from a) the Policies Associator (i.e. the policies associated with the specific SLA) and b) from the Negotiatior in order to obtain specific values (e.g. thresholds on parameters), which are included in the SLA

The Configurator decides on the parameters that will be monitored and the policies that will be checked during the SLA lifecycle. The policies are checked using the monitoring data. The monitoring parameters are sent to Monitoring component. Policies regarding forecast requests are sent to the Analysis component. When the condition of a policy is validated, the Configurator processes the data, recalculates the policies that should be checked and makes corrective actions if it is necessary (e.g. creation of an additional replica).

Additionally, the Configurator translates the SLA requirements that the user determines during the SLA negotiation to low-level requirements that are used in the internal system. For instance, in case that a user requested "availability>99,9%", this is translated to " replicas = 3".

Finally, the Configurator computes placement directions, that is, the number of replicas of a container and the clusters that will be stored. These data are decided according to the SLA requirements and the geographic constraints and are sent by Containers Management to the Placement which executes the placement of the objects.

Proactive Violation Detector. The Proactive Violation Detector detects and handles proactively impendent SLA violations using monitoring data and forecasts. It creates policies for detecting proactively SLA violations and in case that a violation is detected it decides for which reactive actions will be performed.

1) Interactive Components.

The components with which the SLA Management interacts are:

Monitoring. This component measures data during the SLA lifecycle (e.g. the capacity of a hard disc) which are needed for checking the SLA policies and the costs.

Analysis. The Analysis component is responsible for making forecasts for the desired metrics. These forecasts are used for the proactive detection of SLA violations. The SLA Enforcer computes and sends metrics with some thresholds to the Analysis for making forecasts. When Analysis forecasts that a metric will reach the given threshold, it notifies the SLA Enforcer, which checks and handles proactive violation detections.

Placement. This component is responsible for storing the objects

Global View. The Global View is a database that stores the necessary data related to an SLA. These data are stored and accessed by the SLA Management. A distributed database is required and for this reason Cassandra is used.

Accounting and Billing. This component is responsible for computing the user charges according to the user operations and the billing rules that are agreed upon at the SLA negotiation stage.

Identity and Policy Services. This component checks the requestor authentication and authorization of the SLA services.

Compliance. The SLA Management sends logging actions to the Compliance component (e.g. SLA creation, SLA violation, SLA deletion, etc).

C. SLA Enforcement

The SLA Enforcer is responsible for the SLA enforcement and checks continuously if the SLAs are met or if SLA violations occur. It has also a mechanism for detecting proactively impeding violations and it tries to avoid them with corrective actions. Therefore, during an SLA lifecycle, the SLA Enforcer sends a) to Monitoring the metrics that it wants to be monitored and in which periods; and b) to the Analysis the thresholds of the metrics for which it wishes to get notified when forecasts infer that a metric reaches the given forecast. When it receives events that forebode an SLA violation occurrence, the SLA Enforcer examines the current data and recomputes the policies and the forecasts that it needs. It also makes corrective actions if needed. For instance, it may ask from the Placement to move some replicas to other data centers in order to improve the throughput.

Figure 2 depicts the sequence diagram that is followed when the SLA Enforcer receives an event from Monitoring or Analysis and in Figure 3 the policies reconfiguration phase is demonstrated.

1) Functionality of the SLA Enforcer

The SLA Enforcer, through its modules, provides the following functionality:

a) Requirements Translation

The SLA Enforcer is responsible for the requirements translation from the high-level QoS metrics to the low-level ones and vice-versa. Translation from the high-level metrics specified in an SLA (e.g. durability) to low-level metrics by which the internal system works (e.g. number of replicas) is important during SLA management. It is also needed for checking the feasibility of the requested QoS metrics, for generating policies in order to ensure the SLA enforcement and for the placement execution.

a) Container Configuration

During container creation, the SLA Enforcer is contacted by the Container Management component to obtain the QoS requirements stemming from the chosen SLA. Moreover, the SLA Enforcer is responsible for tuning the Monitoring and Analysis components with appropriate parameters needed for the SLA enforcement. These include the metrics that should be monitored and the threat thresholds. Also, it provides to the Container Management component placement requirements (how many replicas and in which locations) in low-level terms.

b) Proactive SLA Violation Detection

One of the most significant responsibilities of the SLA Enforcer is to enable proactive SLA violation detection. The SLA Enforcer calculates for each metric the threat threshold, which is more restrictive than the one signed, and sends it to the Analysis component. The Analysis component receives monitoring information for this metric and calculates trends and patterns. When Analysis forecasts that a metric will reach the given threshold, it notifies the SLA Enforcer. Then, the SLA Enforcer makes decisions on the actions that are needed to prevent the imminent SLA violation and it reconfigures the system with new appropriate policies and monitoring parameters.

For example, if an SLA contains a term on the provided throughput, then the Analysis component will be configured to provide forecasts on this metric. When the forecasts shows that the threshold is about to be exceeded, the SLA Enforcer is notified. This, in turn, decides for a corrective action (e.g. replica creation, replica movement, request redirection to a cluster which is less loaded). For instance, it may conclude that one more replica needs to be created in a new geographic location and notifies the Placement component to take action (e.g. create a new replica).

The proactive SLA violation detection is based on resource usage experiences and historical data and uses the case base reasoning (CBR). CBR is the process of solving problems based on past experience. In the knowledge database used the conditions are stored under which a violation is going to be realized, and the preventative actions

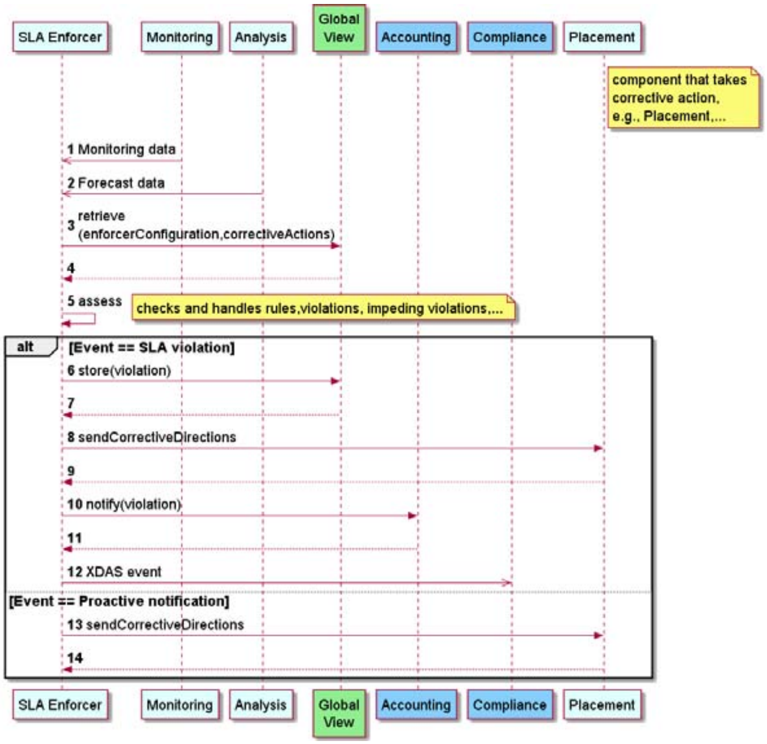


Figure 2: SLA enforcement mechanism sequence diagram: event receiver

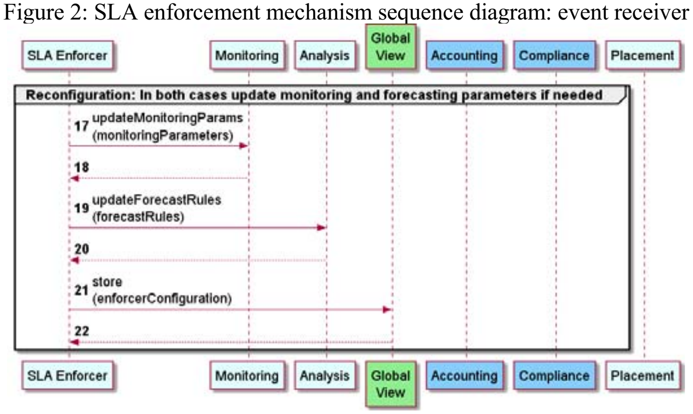
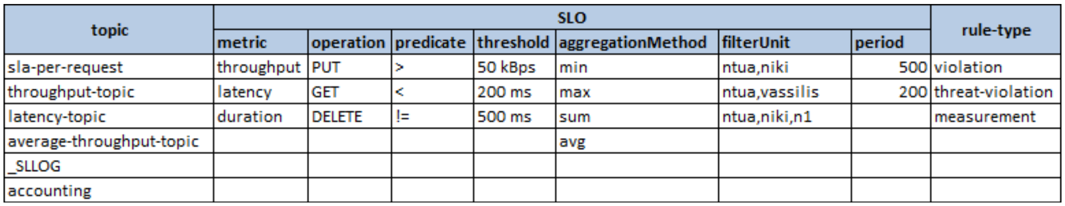
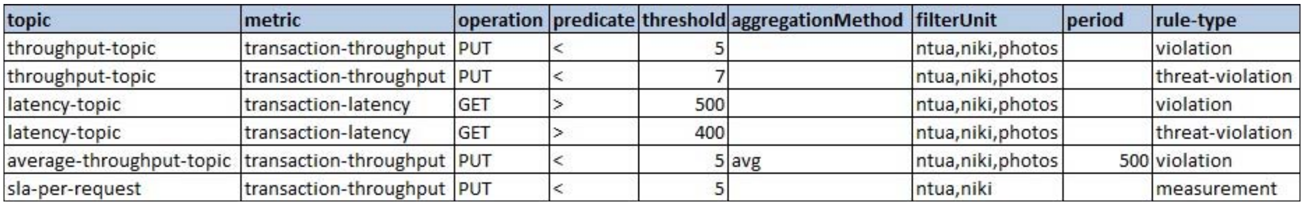

SLA Enforcer
Monitoring
Analysis
Global View
Accounting
Compliance
Placement
Reconfiguration: In both cases update monitoring and forecasting parameters if needed
17 updateMonitoringParams (monitoringParameters)
18
19 updateForecastRules (forecastRules)
20
21 store (enforcerConfiguration)
22
SLA Enforcer
Monitoring
Analysis
Global View
Accounting
Compliance
Placement

Figure 3: SLA enforcement mechanism sequence diagram: reconfiguration



topic	SLO							rule-type
	metric	operation	predicate	threshold	aggregationMethod	filterUnit	period	
sla-per-request	throughput	PUT	>	50 kBps	min	ntua,niki	500	violation
throughput-topic	latency	GET	<	200 ms	max	ntua,vassilis	200	threat-violation
latency-topic	duration	DELETE	!=	500 ms	sum	ntua,niki,n1		measurement
average-throughput-topic					avg			
_SLLOG								
accounting								

Figure 4: monitoring parameters fields and indicative values



topic	metric	operation	predicate	threshold	aggregationMethod	filterUnit	period	rule-type
throughput-topic	transaction-throughput	PUT	<	5		ntua,niki,photos		violation
throughput-topic	transaction-throughput	PUT	<	7		ntua,niki,photos		threat-violation
latency-topic	transaction-latency	GET	>	500		ntua,niki,photos		violation
latency-topic	transaction-latency	GET	>	400		ntua,niki,photos		threat-violation
average-throughput-topic	transaction-throughput	PUT	<	5	avg	ntua,niki,photos	500	violation
sla-per-request	transaction-throughput	PUT	<	5		ntua,niki		measurement

Figure 5: monitoring policies examples

and solutions that should be performed for avoiding the violations.

a) SLA Violation Handling

When an SLA violation occurs, a notification is sent to the Accounting and Billing in order for the provider to be charged with the agreed penalty. Moreover, the SLA Enforcer stores appropriate information in the Global View so as to be used for preventing future SLA violations.

2) Policesbased mechanism

The SLA Enforcement is based on policies that are created runtime according to the SLOs and the monitoring data. There are two policies categories: a) one for the monitoring metrics that are sent to Monitoring and b) another for the forecast metrics that are sent to Analysis.

The SLOs defined in the SLAs are the requirements that should not be violated. Some of the metrics that are checked in the SLA enforcement are the inbound throughput, the outbound throughput, the latency and the durability.

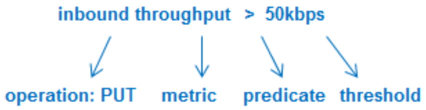


Figure 6: SLO and monitoring policiy fields association

The monitoring policies exist in order to check the signed SLOs. An SLO consists of the metric (e.g. throughput, latency, etc.), the predicate (e.g. >, < etc.) and the threshold (e.g. 50 kbps, 500 ms etc.). Also, the type of the operation (e.g. PUT, GET, etc.) should be declared. Let's assume that a tenant desires inbound throughput to be more than 50 kbps. In this case the throughput is the metric, the inbound represents that the operation is 'PUT' and the threshold is 50 kbps (see Figure 6). Besides these data, the entity that concerns the requirement is declared. This is called filter unit and for instance can be a container, a user, etc. However, a tenant may desire requirements for aggregated data. For instance, a tenant may desire average throughput at least 50 kBps for all the requests of a container for each 10 minutes. So, there are two additional fields that concern the aggregation: a) the aggregation method which represents the method that will be executed for the aggregation, e.g. average, maximum, minimum etc. and b) the period which demonstrates per which period the aggregation is requested.

The aforementioned fields describe an SLO. In the monitoring policies there are some additional data for handling the SLA enforcement. These are:

- the topic: the topic is like grouping policies. It is needed for registering to the monitoring policies for requesting events of this topic. For each topic the events handler is different.
- the rule-id: the rule id is the identifier of the policy and it is used for communication of the SLA Management with the Monitoring . The SLA Management requests from the monitoring to check the registered rule and the monitoring returns the rule id. When a rule is triggered, then the monitoring sends to the SLA Management an event which contains the topic, the rule id, the value of the metric in the time that the rule was triggered and the timestamp. From the rule id, the SLA Management retrieves the rest data of the rule and it handles the event according to the rule topic and the rule type.
- the rule-type: the rule type indicates why this rule is created. For instance it is created for checking violations, for detecting proactively impendent SLA violations or simply for measurements where he can observe for instance the user usages and later to suggest better and less costly SLAs during the renegotiation. According to the rule-type there is different process for handling the event.

In Figure 4, the fields of a monitoring policy and some indicative values for each one are demonstrated. In Figure 5 there are some monitoring policies examples. Let's assume that a tenant requests throughput for each PUT request to be more than 5 kbps. For this request, there are the two first policies in the table in Figure 6. The first requires notification in case of violation, that is, in case that the throughput is less than the requested threshold 5 kbps. So, here the predicate is '<' and the rule type is 'violation'. The second policy is for the proactive SLA violation detection. In this policy the threshold is 7 kbps which is more restrictive that the signed one and the rule type is 'threat-violation'.

VI. Example

In this section, an example is demonstrated (Figure 7). Let's assume that the Cloud provider is a bank which provides banking services and the customer wants to send remittance by phone banking. The bank provides services either online or by phone banking having an automated voice response system (AVRS).

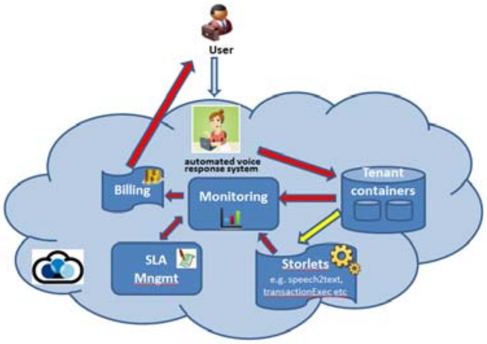

User
automated voice response system
Billing
Monitoring
Tenant containers
SLA Mngmt
Storlets
e.g. speech2text, transactionExec etc

Figure 7: bank provider example

The customer calls the bank and the AVRS responds giving the supported services. The customer selects the service that he desires which in our case is to send remittance. The AVRS requests authentication data from the customer and the customer gives them. This voice record is stored in the Cloud and the storlet 'speech2text' is triggered which automatically transforms the voice record to text and then with another trigger the authentication is checked. If the authentication is successful the process is continued. Data regarding the remittance such as receiver and amount are requested by the AVRS. Similarly, the voice record is transformed to text and the transaction is executed. All these data are stored in the Cloud with the timestamp and can be retrieved from the user anytime e.g. for getting his transactions.

In order to use such services, the customer firstly has signed an SLA. The requirements are customized according to the tenant's needs. For instance, the tenant may not to care very much about the service to be supported anytime as to be completed safely having its data encrypted in order not to be hacked. So, the SLA contains security and encryption requirements having availability in a medium level, which doesn't mean that is low but is less than the highest provided one. During the SLA Negotiation, the SLA Management requests from the tenant to upload his identity and other input required for the bank for instance internal revenue service documents.

During the SLA lifecycle, when a transaction is requested, the storage data and the storlet executions are detected by the monitoring. The monitoring information is processed by the SLA Management for checking SLA violations and for updating the monitoring policies. Finally, the monitoring information is used by the Billing for calculating the user bill for the agreed period according to the SLA. Finally, in case of SLA violations the SLA Management informs billing and the agreed penalty is charged to the provider.

VII. Conclusions

In conclusion, we proposed an automated SLA Management for content centric storage. It depends on an enriched SLA schema which contains content terms and sections for storlets and federation. The SLA Management takes advantage of the chosen content terms and supports services to the customer more efficiently and with less cost. During the SLA enforcement dynamic rules are created and updated in order to detect and handle proactively SLA violations. Also, dynamic SLAs are supported, as the SLA templates are generated according to the current supplies. Further research should be done on complicated requirements and on corrective actions.

Acknowledgment

This paper is supported by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n 257019, in the context of the VISION Cloud Project.

References

[1] H. Kolodner, D. Naor, S. Tal, S. Koutsoutos, N. Mavrogeorgi, S. Gogouvitis, D. Kyriazis, and E. Salant, "Data-intensive Storage Services on Clouds: Limitations, Challenges and Enablers,"in eChallenges e-2011 Conference, 2011.

[2] Voulodimos, A.; Gogouvitis, S.V.; Mavrogeorgi, N.; Talyansky, R.; Kyriazis, D.; Koutsoutos, S.; Alexandrou, V.; Kolodner, E.; Brand, P.; Varvarigou, T., "A Unified Management Model for Data Intensive Storage Clouds," Network Cloud Computing and Applications (NCCA), 2011 First International Symposium on 21-23 Nov. 2011

[3] Athanasios Voulodimos, Spyridon Gogouvitis, Nikoletta Mavrogeorgi, Roman Talyansky, Dimosthenis Kyriazis, Stefanos Koutsoutos, Vasileios Alexandrou,Elliot Kolodner, Per Brand, Theodora Varvarigou, *"A Unified Management Model for Data Intensive Storage Clouds,"* IEEE First International Symposium on Network Cloud Computing and Applications, Toulouse, France, November 21-23, 2011

[4] Mavrogeorgi, N.; Gogouvitis, S.; Voulodimos, A.; Katsaros, G.; Koutsoutos, S.; Kiriazis, D.; Varvarigou, T.; Kolodner, E.K., *"Content Based SLAs in Cloud Computing Environments,"* Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on , vol., no., pp.977,978, 24-29 June 2012

[5] D. D. Lamanna, J. Skene, and W. Emmerich, "SLAng: A Language for defining Service Level Agreements," in FTDCS '03: Proceedings of the The Ninth IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'03), (Washington, DC, USA), IEEE Computer Society, 2003

[6] V. Tosic, B. Pagurek, and K. Patel, "WSOL - A Language for the Formal Specification of Classes of Service for Web Services.," in ICWS (L.-J. Zhang, ed.), pp. 375–381, CSREA Press, 2003.

[7] IBM Web Service Level Agreements (WSLA) Project: http://www.research.ibm.com/wsla/

[8] Nicole Oldham, Kunal Verma, Amit Sheth, and Farshad Hakimpour. 2006. Semantic WS-agreement partner selection. In Proceedings of the 15th international conference on World Wide Web (WWW '06). ACM, New York, NY, USA, 697-706.

[9] A. Andrieux, K. Czajkowski, A. Dan, K. Keahey, H. Ludwig, T. Kakata, J. Pruyne, J. Rofrano, S. Tuecke, M. Xu: Web Services Agreement Specification (WS-Agreement), GFD.107.http://www.ogf.org/documents/GFD.107

[10] Emeakaroha, Vincent C.; Brandic, Ivona; Maurer, Michael; Dustdar, Schahram; , "Low level Metrics to High level SLAs - LoM2HiS framework: Bridging the gap between monitored metrics and SLA parameters in cloud environments," High Performance Computing and Simulation (HPCS), 2010 International Conference on , vol., no., pp.48-54, June 28 2010-July 2 2010.

[11] Brandic, I.; Emeakaroha, V.C.; Maurer, M.; Dustdar, S.; Acs, S.; Kertesz, A.; Kecskemeti, G.; , "LAYSI: A Layered Approach for SLA-Violation Propagation in Self-Manageable Cloud Infrastructures," Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual , vol., no., pp.365-370, 19-23 July 2010.

[12] Vincent C. Emeakaroha, Marco A. S. Netto, Rodrigo N. Calheiros, Ivona Brandic, Rajkumar Buyya, César A. F. De Rose ;"Towards autonomic detection of SLA violations in cloud infrastructures", Future Generation Computer Systems (November 2011).

[13] M. Boniface, S.C. Phillips, A. Sanchez-Macian, M. Surridge, Dynamic service provisioning using GRIA SLAs, in: Proceedings of the 5th International Workshops on Service-Oriented Computing, ICSOC'07, 2007.

[14] Katsaros, G.; Gogouvitis, S.; Mavrogeorgi, N.; Voulodimos, A.; Kiriazis, D.; Varvarigou, T.; Talyansky, R.; , "A Holistic View of Information Management in Cloud Environments," Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on 24-29 June 2012

[15] Gogouvitis, S.V.; Alexandrou, V.; Mavrogeorgi, N.; Koutsoutos, S.; Kyriazis, D.; Varvarigou, T., "A Monitoring Mechanism for Storage Clouds," Cloud and Green Computing (CGC), 2012 Second International Conference on 1-3 Nov. 2012

[16] George Kousiouris, Dimosthenis Kyriazis, Spyridon V. Gogouvitis, Andreas Menychtas, Kleopatra Konstanteli and Theodora A. Varvarigou, "Translation of Application-level Terms to Resource-level attributes across the Cloud Stack Layers ,", Computers and Communications (ISCC), 2011 IEEE Symposium on , vol., no., pp.153-160, June 28 2011-July 1 2011

[17] George Kousiouris,, George Vafiadis, Marcelo Corrales, "A Cloud Provider Description Schema for Meeting Legal Requirements in Cloud Federation Scenarios", 12th IFIP WG 6.11 Conference on e-Business, e-Services, and e-Society, I3E 2013, Athens, Greece, April 25-26, 2013

[18] http://www.visioncloud.eu/