SLA Management in Clouds

Nikoletta Mavrogeorgi[1], Spyridon Gogouvitis[1], Athanasios Voulodimos[1], Dimosthenis Kiriazis[1], Theodora Varvarigou[1] and 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*

Keywords: Cloud, SLA Schema, SLA Management, Content Centric Storage, Policies, Proactive Violation Detection.

Abstract: The need for online storage and backup of data constantly increases. Enterprises from different domains, such as media and telco operators, would like to be able to store large amounts of data which can then be made available from different geographic locations. The Cloud paradigm allows for the illusion of unlimited online storage space, hiding the complexity of managing a large infrastructure from the end users. While Storage Cloud solutions, such as Amazon S3, have already met with success, more fine-grained guarantees on the provided QoS need to be offered for their larger uptake. In this paper, we address the problem of managing SLAs in Storage Clouds by taking advantage of the content terms that pertain to the stored objects thus supporting more efficient capabilities, such as quicker search and retrieval operations. A system that better captures the content of the signed SLAs is very useful, as the benefits in terms of reducing the management overhead while offering better services are potentially significant.

1 INTRODUCTION

Nowadays, online storage of data is very demanding. Many sectors, such as media, enterprises, medicine and telco need to store data and access them rapidly any time from different geographic locations.

Storage Cloud environments can provide the solution for these demands. The customers do not need to own specialized hardware for storing their data objects or have concerns for management tasks, such as backups, replication levels, etc.. In order for customers to be willing to move their data to Cloud solutions, proper Service Level Agreements (SLAs) should be made available and signed. SLAs are contracts between the customer and the service provider, where the terms and conditions of the offered service are agreed upon.

In this paper, we address the problem of managing SLAs in cloud computing environments exploiting the content term that concerns the stored objects, in order to provide more efficient capabilities.

The proposed SLAs are enriched versions of the traditional ones. Apart from determining the classic elements such as the parties involved, the SLOs and the cost rules, the SLAs additionally include content term determination of the objects that will be associated with this SLA, a fact that permits the Cloud to provide the users with content-centric services.

For instance, let's assume that we want to store scientific papers and news. The papers are requested less often than the news, which are accessed frequently on a daily basis. So, for the daily news it would be better to create more replicas than the papers and store them to many geographic places, in order to have a quicker access.

The content terms permit to the Cloud to support capabilities with different levels, such as the estimation for data transfer and the billing models.

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 video is requested to be stored in the Cloud, then it automatically gets transformed to e.g. mpeg format by the system, which facilitates internal actions in Cloud and it is stored in data centers that provide efficiently video related services.

The following sections describe the proposed SLA schema and the SLA Management mechanism.

2 RELATED WORK

A lot of research and protocols have been done as far as Service Level Agreements (SLA) and the 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 (Lamanna et al., 2003), WS-Agreement (Andrieux et al., n.d.), WSLA (Dan et al., 2004), WSOL (Tosic et al., 2003), and SWAPS (Oldham et al., 2006).

However, weaknesses exist. SWAPS is quite complex and the implementation is not publicly available. WSLA and SLAng 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 lacks SLA related functionalities, such as the capture of the relationship between service provider and infrastructure provider.

The WS Agreement (Web Services Agreement) 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.

A challenge research issue is the proactive violation detection. Many proposals have been done, but very little for cloud environments, e.g. authors of GRIA SLAs (Boniface et al., 2007) suggest a solution for avoiding violations but concerns only Grid environments. In DesVi (Emeakaroha et al., 2011), an architecture is proposed for preventing SLA violations based on knowledge database and case-based reasoning.

3 SLA SCHEMA

The Service Level Agreement is encapsulated in an SLA schema. 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).

3.1 Basic Elements

The SLA element is the outermost element, which encapsulates the entire SLA. An SLA is created by filling an SLA template. In the SLA Template there is the choice to determine 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.

Figure 1 and 2 displays the schema of the SLA and the content term element respectively.

3.2 SLA Properties

An SLA contains, based on the WS agreement, a name element, a context element and a terms element, the latter containing a collection of SLA terms. The context element contains the metadata of the agreement (e.g. participants and dates) and the terms element contain the SLOs, the content term, the responsibilities, and the billing rules.

More specifically, the SLA consists of the following properties (see Figure 1):

- 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
    - 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.

3.3 Content Term Properties

The basic properties that compose the Content Term element are the following (see Figure 2):

- 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.

4 SLA MANAGEMENT

4.1 Description

The SLA Management is responsible for the creation of an SLA and its enforcement and maintenance.

The SLA Management is based on the aforementioned SLA schema. It takes into consideration content related terms that will be included in the SLAs since content-centric access to storage poses the need to use content-related information. Moreover, the system is one of the consumers of the analyzed monitored data in order to enable proactive SLA violation. To this direction, the specific SLA terms as well as policies that may be set by the administrators of the infrastructure (e.g. perform replication in case of a given percentage of increase in user requests) will be analyzed in the SLA management component and propagated to the Analysis in order to pro-actively trigger events that will prevent possible SLA violations

The main operations of this component are the SLA negotiation, the SLA templates generation, the SLAs creation, the proactive SLA violation detection and handling and the configuration of the rest components when a new SLA is created (it creates policies that should be checked during the SLA lifecycle, it sends the parameters to be monitored to the Monitoring component, it informs Accounting and Billing for the new SLA and its cost related data, etc.). As far as the proactive SLA violation detection is concerned, this component gets the analyzed monitored data, it processes them and creates policies that are sent to Analysis to proactively trigger events.

4.2 Architecture

The architecture of the SLA Management is displayed in Figure 3. The basic components of the SLA Management are the following:

4.2.1 Templates Generator

This component is responsible for generating SLA templates. It creates SLA templates based on the capabilities that the Cloud can provide and based on content terms, having as a result to provide dynamic and customized SLAs

4.2.2 Negotiator

The Negotiator is responsible for the SLA negotiation between the user and the provider. When an SLA is signed, it notifies the interactive components. Also, it informs the user with reports and notifications (e.g. for SLA violation). The steps are depicted in Figure 4.

4.2.3 Policies Associator

This component is responsible for the linkage of policies with SLA templates. This will be achieved through historical data and the way specific policies had an effect on the SLA violations.

The Policies Associator decides for the policies that should be followed for the concerned SLA and provides them to the Configurator.

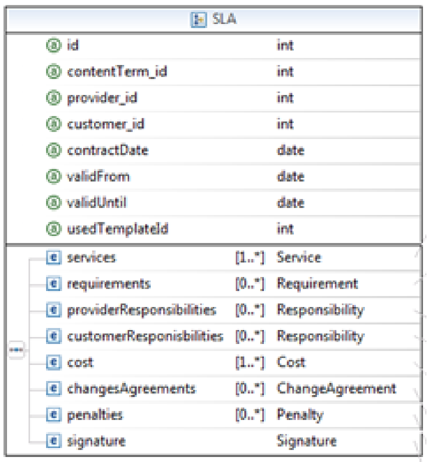

Figure 1: SLA element.

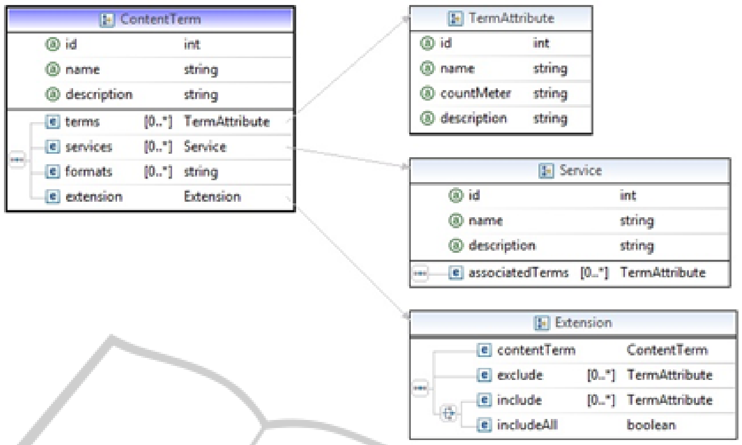

Figure 2: Content Term element.

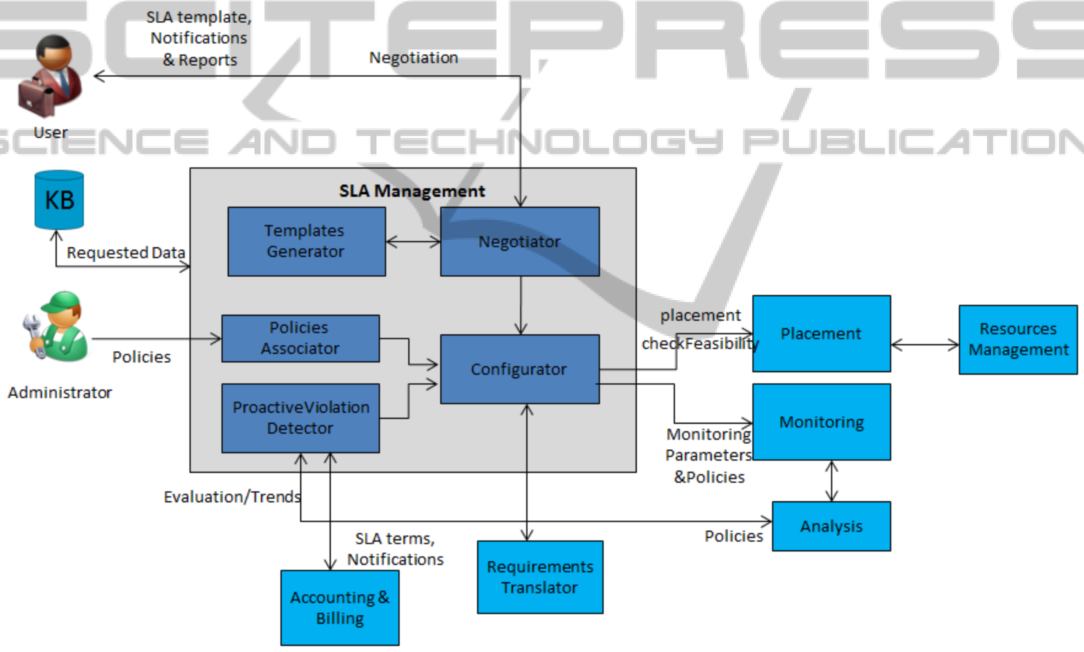

Figure 3: SLA Management Architecture.

4.2.4 Configurator

The Configurator creates the necessary configuration regarding the SLA in order the system to support the customer requested requirements. It receives the signed SLA from the Negotiator in order to obtain specific values (e.g. thresholds on parameters), and decides which parameters will be monitored and which policies will be checked. Also, it consults the Placement where the uploaded objects should be stored and with how many replicas depending on the SLA constraints. In table 1, an example of an SLA and its SLA configuration is depicted.

4.2.5 Proactive Violation Detector

This component handles the proactive SLA violation detection. 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. It calculates for each metric the threat threshold, that is, a more restrictive threshold than the signed one and sends to the Analysis the metric and the threat threshold.

Analysis receives monitoring information for this metric and calculates trends and patterns. When Analysis forecasts that a metric reaches the given

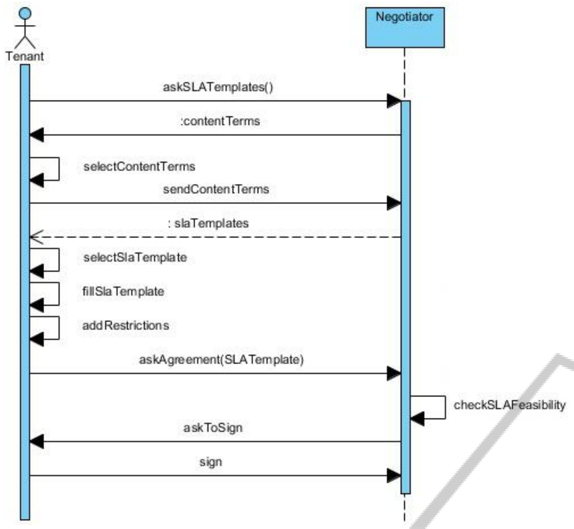

Figure 4: SLA Negotiation.

threshold, it notifies the SLA Management. Then, SLA Management handles this notification and makes reactive actions (e.g. informs placement to create a new replica when throughput reaches the threat threshold) in order to avoid the imminent violation. The conditions and the solution are retrieved by the knowledge database which contains data based on historical data.

For instance, let's assume that we have an SLA with durability 99,5% and geographical constraints that the objects are preferred to be stored in Greece and Israel. The equivalent low level requirements will be the creation of 3 replicas (at least one in Greece and at least one in Israel). If the data centers in Greece have no more free space, then a reactive action would be either to move some replicas to other countries in order to get free space or to buy new hardware in Greece. So, for the concerned SLA, the free space of the data centers in these countries should be checked in order to identify when the data centers are going to get filled.

Following we demonstrate the data that are exchanged (simplified examples, where some data are not demonstrated). The policies for an SLA that SLA Management sends to Analysis is:

```
{
 'sla_id': '20120204',
 'metrics':{'CPU': { 'max': '75%' },
            'memory': { 'max': '65%'}}
}
```

Let's assume that Analysis found that the CPU utilization for the SLA with id '20120204' will reach the threat threshold and it will be 75% in 20 minutes from now, then it will send the following notification to the SLA Management:

```
{
   'sla_id': '20120204',
   'timestamp': '2012-05-12_13:08'
   'metric': 'CPU',
   'forecast': {
                'value': '75%',
                'time': '20 min'
               }
}
```

Then, the SLA Management will find the reactive action that should be taken. In this case, a new replica should be created.

4.2.6 Interactive Components

The SLA Management in order to accomplish its tasks needs some interactions with other components. The basic external components that the SLA Management interacts are:

- Monitoring: measures data during the SLA lifecycle (e.g. CPU speed) which are needed for checking the SLA policies.
- Analysis: is responsible for forecasting the metrics computing trends and patterns.
- Placement: decides in which data centers the requested objects will be stored.
- Knowledge Base (KB): is a database that stores data related to an SLA.
- Requirements Translator: translates the SLA requirements that the user determines to low level requirements with which the internal system communicates. For instance, the requirement "availability>0.999" could be translated to "number of replicas = 3".
- Accounting and Billing: charges the user according to the user operations.

5 CONCLUSIONS

Concluding, we presented an enriched SLA schema which contains content terms and an automated SLA Management for content centric storage, which exploits the content terms and support services to the customer more efficiently and with less cost. Also, dynamic SLAs are supported, as the SLA templates are generated according to the current supplies. More work should be done for the requirements translation and the analysis of metrics in order to detect trends and patterns of the metrics. Finally, examination of complicated requirements should be done and renegotiation should also be added in the proposed framework in order to provide a complete automated SLA Management.

Table 1: Service ‘SLA configuration’ with its input parameter and output.

Input: SLA	Output :SLA Configuration
{	{
"slaId": "0",	"slaId": "0",
"customerId": "ntua",	"customerId": "ntua",
"slaRequirements": {	"lowLeveRequirements": {
"durability":"99%",	"wishlist": [[[1], [Greece, Israel]],
"preferredRegions": ["Greece",	[[1], [Italy]]]],
"Israel", "Italy"],	"blacklist": [Russia],
"blacklist": ["Russia"],	"distance": [2, 100]
}	}
}	}

ACKNOWLEDGEMENTS

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

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.*

A. Voulodimos, S. Gogouvitis, N. Mavrogeorgi, Roman Talyansky, D. Kyriazis, S. Koutsoutos, V. Alexandrou, E. Kolodner, P. Brand, T. 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.

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.

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.

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.

N. Oldham, K. Verma, A. Sheth, and F. 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. DOI=10.1145/1135777.1135879.

A. Dan, D. Davis, R. Kearney, R. King, A. Keller, D. Kuebler, H. Ludwig, M. Polan, M. Spreitzer, and A. Youssef, Web Services on demand: WSLA-driven Automated Management, *IBM Systems Journal, Special Issue on Utility Computing, Volume 43*, Number 1, pages 136-158, IBM Corporation, March, 2004.

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.pdf.

V. C. Emeakaroha, M. A. S. Netto, R. N. Calheiros, I. Brandic, R. Buyya, C. A. F. De Rose ;“Towards autonomic detection of SLA violations in cloud infrastructures”, *Future Generation Computer Systems (November 2011)*, doi:10.1016/j.future.2011.08.018.

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.