Systematic SLA Data Management

Katerina Stamou
supervised by: Prof. Jean-Henry Morin
Institute of Services Science
University of Geneva, Switzerland

ABSTRACT

The cloud computing paradigm emerged with service oriented principles. In the cloud setting, organizations outsource their IT equipment and manage their business processes through virtual services that are typically exchanged over HTTP. Service Level Agreements (SLAs) depict the status of running services. SLAs represent operational contracts that allow providers to estimate their service availability according to their resource capacity.

The SLA data schema and content are operationally defined by the type, volume and relations of service elements that organizations operate on their physical resources. Current lack of a uniform SLA standardization leads to semantic and operational differences between SLAs, that are produced and consumed by different organizations. Such differences prohibit common business SLA practices in the cloud computing domain. Our research introduces systematic SLA data management to describe the formalization, storage and processing of SLAs over distributed computing environments. Services in scope are framed within the cloud computing context.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Online Information Services; H.2 [**Database Management**]: Data models

Keywords

service level agreement; property graph; data management; distributed service management

1. PROBLEM

A basic characteristic of the cloud computing model is the "on-demand" exchange of services. Considerably, cloud SLAs need to be processed also on-demand and over distributed information pools. SLAs become complex in terms of content and structure due to the diversity and plethora of offered services. A flexible and modular data model that allows for the automatic SLA formulation and efficient processing, is required to address these issues.

Our approach is built upon the Web Service Level Agreement (WSLA) language [13] that has been proposed by IBM for distributed service management. Namely, we use the WSLA as the skeleton alphabet of our proposed SLA data model. The latter enables the systematic treatment of SLA information and advances their role from unstructured, temporary data to structured ones that can be processed and analyzed efficiently. In the cloud business setting, efficiency deals with the reliable and on-demand exchange of end-user information and of computational resources.

1.1 Service Level Agreement definition

According to Dan et al.[7], SLAs represent contractual terms and conditions between service providers and customers. Their content assures the mutually agreed service levels between a provider and a consumer. SLAs describe obligatory service provisioning terms. They encapsulate quality of service (QoS) characteristics and functional service properties.

Moreover, SLAs may include a multitude of technical and business service-level objectives (SLOs) along with metrics that enable the computation and measurement of service level targets. Every customer needs to agree with an SLA in order to lease a new service. Traditionally, providers define SLAs, in which they guarantee explicit service-level bounds over a predefined, agreed period.

1.2 SLA data complexity

SLAs consist of semi-structured information and do not follow a fixed schema for their specification. Their semi-structured representation is primarily attributed to the lack of a uniform SLA standard that is abstract enough to apply for diverse business domains within the cloud computing setting. The following facts designate the complexity in handling SLA information:

Heterogeneity: A plethora of diverse services are offered in cloud computing markets. The description of provisioning levels is defined by the customized terminology of each business domain. Additionally, the service and SLA management literature lacks benchmarks that could contribute to the standardized definition and evaluation of service metrics.

Service dependencies: SLA elements are inter-dependent. According to Keller et al.[12], service dependencies represent customer/provider relationships that are reflected to the var-

WWW'14, April 7-11, 2014, Seoul, Korea.
ACM 978-1-4503-2745-9/14/04. http://dx.doi.org/10.1145/2567948.2567952.

ious cooperating components within a distributed service management system.

Real time measurement/data operations: SLAs are used for real-time service monitoring and for auditing of provided service level metrics. Service level values are communicated to dependent SLA elements, whose values are in turn updated. Frequent SLA audits verify the running service status with respect to violations of agreed provisioning levels.

1.3 SLA utility in the cloud

In the cloud computing setting, services are typically exchanged between customers and providers over the HTTP layer. Moreover, computing processes that execute the service management, may also represent the signatories. Cloud business models follow economies of scale, where the long-term average cost decreases by exploiting scaling-up and -out techniques. Cloud service markets support the simultaneous provisioning of multiple, remotely-connected customers and are subject to the volatility of customer demand and of resource availability.

The primary SLA utility engages their role as measurement instruments in the service execution. SLA management requires clear and granular information flow such that SLA data can be retrieved and processed rapidly. However, current cloud markets provide static SLA documents that do not allow for any processing. Such contracts typically do not go into depth with respect to technical service-level details. Instead, they primarily include terms and conditions that are peripheral to the service functionality. The SLA utilization by public marketplaces would be feasible if SLAs were formalized automatically, following a modular data model that permits their efficient processing.

2. STATE OF THE ART

2.1 SLA formalization

The distributed computing community has driven research and technical advancements on SLA management to cover immense needs for resource reservation through the monitoring of running computational tasks. In contrast to other forms of contracts for IT services, the values of SLA elements have to be measured and monitored during workload execution to audit potential service-level violations or to verify adherence to the agreed SLOs. In the literature, SLAs are hardly viewed as end-user documents but merely as automated processes that assist the monitoring and scheduling of computational tasks.

The IBM research on utility computing [3] initially addressed the need for automated SLA formalization with the specification of the Web Service Level Agreement (WSLA) [13] language. The Grid Resource Allocation and Agreement Protocol (GRAAP) working group followed with the Web Services Agreement (WS-Agreement) specification [1] as a language and a protocol to manage SLAs.

Both specifications use a tree data structure to represent the SLA information. Tree branches illustrate separate SLA sections and tree leaves inner section terms. XML has been used in both schemas as the standard means of exchanging information among web services. According to the WSLA language, an SLA consists of the following core elements:

Parties: Signatory parties consist of one service provider and one service customer. Supporting parties represent third parties that operate on behalf of either or both signatories.

Service definition: Service objects represent description terms and include SLA parameters that contain properties and indicate quantitative as well as qualitative metrics.

Obligations: A provider defines guarantees in the form of obligations either as SLOs or as action guarantees. SLOs represent measurable targets that a provider promises to fulfill during service execution. SLO values can be verified via measurement through monitoring. Action guarantees cover tasks that the provider, one or more supporting parties or, in some cases, the customer will take to establish the promised service levels for one or more SLA parameters.

The core elements of the language are further divided into granular sub-elements. In the case of service objects or operations, such information reaches the level of URI sources that can be monitored and measured. In the case of service obligations, the information granularity includes the specification of business level objectives as well as the definition of evaluation functions and expressions that typically follow first order logic. In addition, the obligations section includes the formalization of penalties in case of service-level violations and of conditions for re-imbursements and rewards that represent penalty complements.

2.2 Monitoring, auditing and resource management

The literature highlights numerous research results on the monitoring [19, 4] of service-level terms and the scheduling [8, 5, 6] of computational jobs. Upon agreement initiation, an SLA document is managed by available monitoring and auditing services until the execution of running tasks is complete.

SLA documents must enclose all necessary parameters for the measurement and evaluation of active service and resource capacity levels. In [10] Ludwig et al. define '**services**' as the processing of computationally intensive tasks that may involve network operations, the exchange, for example, of SOAP or REST messages over HTTP or a combination of computing workloads that may affect several network and system layers. In [5] Czajkowski et al. define '**resources**' as any capability that can be shared and exploited in a networked environment, including physical computational resources.

The automated SLA processing enables the discovery and mapping of available resources and therefore requires the monitoring and auditing of active workloads. Monitoring data is used for the effective allocation of resources. Programming processes differentiate among types of reservations, conditions and policies that are enforced during the service runtime. Changes in resource availability, e.g. storage space, are communicated via resource management processes.The SLA content helps with the establishment of commonly accepted policies [6] that enable the smooth interoperation and the successful completion of running tasks.

2.3 SLA negotiation and brokering

The research community has proposed negotiation and brokering protocols [10] for SLA management. SLA negotiation enables the exchange of counter offers between contracting parties in the process of agreeing on service levels.

According to Czajkowski et al.[6], the need for negotiation naturally derives from the conflicting interests of contracting parties. On one hand, customers need to understand provisioning terms and to receive guarantees on requested service

levels. On the other hand, providers have to maintain control over resource provisioning and adhere to the promised service levels. Furthermore, in [8] Foster et al. highlight the role of resource brokers for the allocation of resources and for the monitoring of service endpoints.

3. PROPOSED APPROACH

We propose a directed property-graph data model for the formalization and management of SLA information. According to Rodriguez [15], a property graph represents a multi-relational graph, where nodes and edges contain arbitrary attributes in the form of key/value pairs. Property graphs can prove valuable as they encapsulate all the information that is required to thoroughly manage an SLA document. Moreover, the direction of edges is helpful to indicate dependencies among internal SLA components. Finally, the SLA data modeling through graphs allows for flexible schemas, e.g. hypergraph representations, alternative index sets and filtering rules. Such modeling attributes permit the complete representation of resources and of any other element (metrics, guarantees, conditions, etc.) that is part of service and SLA management.

The proposed SLA data model is supported by one or more graph-aware DBMS that enable the graph management from the application layer (e.g. through a RESTful web interface). We assume that the SLA graph manipulation involves asynchronous computational tasks to achieve the simultaneous handling of HTTP operations with respect to a given client-server architecture. Moreover, we assume that an SLA document contains conditions, whose evaluation may provoke additional data operations to chains of SLA elements.

The structured accommodation of the SLA content into nodes and edges permits their direct identification and retrieval. Graph queries are expressed as regular path queries or a graph DBMS may support a property-graph domain specific language (DSL) to interact with underlying graphs.

3.1 SLA graph data model formalization

We define an SLA database as a finite directed graph, where nodes represent SLA elements and edges represent directed relationships between SLA elements. The SLA graph data model uses the hierarchical structure and alphabet of the WSLA specification for the skeleton SLA database.

We denote the SLA graph alphabet as $SLA_{alphabet} \supset \{WSLA_{alphabet}, A_{custom}\}$, where A_{custom} symbolizes a non-finite alphabet that is bounded by a custom business domain.

An SLA can be expressed as a directed graph $SLA_g = (N, E)$, where N represents a set of nodes $N \supset n_i \in SLA_g$ and E a set of ordered node-pairs $e_i = (n_i, n_{i+1})$, where $E \supset e_i \in SLA_g$. Given a directed edge e_i in SLA_g, n_i represents the predecessor node of n_{i+1},thus denoting a direct path from n_i to n_{i+1}.

The graph database considers service dependencies and internal relationships between elements. To model the skeleton graph of the SLA graph database, we first transform primary WSLA language components into element sets by taking into account their cardinalities and relationships. Table 1 illustrates identified element sets. We use a subscript to denote their cardinality in any SLA graph instance.

Table 1: SLA language elements - Set representation

SLA$\supset \{Signatory_2, Obligations_1, ServiceInfo_1, \{ServiceDescription_i\}\}$, where $i \in [0, n]$
$Signatory_2 \supset \{SupportParty_i\}$, where $i \in [0, n]$
$Obligations \supseteq \{\{SLO_i\}, \{ActionGuarantee_i\}\}$, where $i \in [1, n]$
$ServiceDefinition_i \supset \{\{ServiceObject_i\}, \{Operation_i\}\}$, where $i \in [0, n]$
$ServiceObject_i, Operation_i \supset \{SLAparameter_i\}$, where $i \in [0, n]$
$SLAparameter_i \supset \{\{ResourceMetric_i\}, \{CompositeMetric_i\}\}$, where $i \in [0, n]$
$CompositeMetric_i \supseteq \{\{ResourceMetric_i\}, \{CompositeMetric_j\}, Function_j\}$, where $i \in [1, n]$ and $j \in [0, n]$
i, j indicate the cardinality of element subsets and $n \in \mathbb{N}$.

3.2 SLA graph analysis

Figure 1 illustrates the skeleton SLA graph that is composed of 19 nodes and 22 directed edges. Every node is identified by a unique id that is assigned on the node definition. Similarly, edges in the SLA graph are also identified by a unique id. Additionally, a vertex and an edge can have an arbitrary number of attributes associated with them.

The skeleton SLA graph of Figure 1 has an average degree of 1.684 and an average path length of 2.148. Figure 1 highlights each node degree by mapping it to the node size. The degree represents the sum of incoming and outgoing edges for each node and thus provides an initial view on node communication patterns in the SLA graph. Still, as Figure 1 depicts only the skeleton of the SLA data model, the degree distribution is subject to change in larger SLA graph databases that represent actual cloud service descriptions.

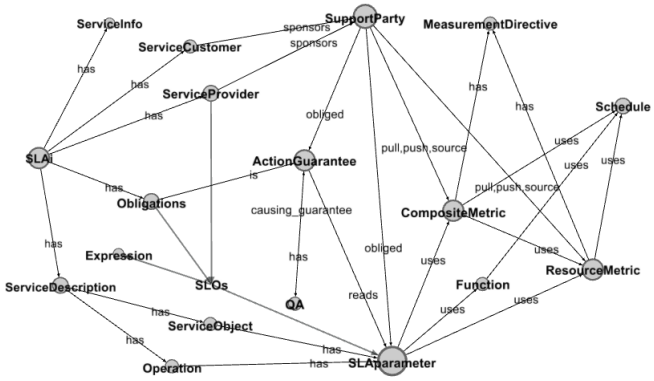

Figure 1: SLA skeleton graph - Degree representation

With the term **dependency** Keller et al. [12] define the relationship between a dependent service or application component that requires an operation performed by an antecedent component in order for the former to execute its function. Keller et al. use the terms **antecedent** and **dependent** to express the two counterparts of the relationship.

In the graph, service dependencies are denoted as edges that are outgoing for the dependent node and incoming for the antecedent node, $<Dependent_{tail}, Antecedent_{head}>$. Furthermore, the edge direction $<Tail_{node}, Head_{node}>$ in the SLA graph of Figure 1 also denotes the nesting of SLA elements according to the set representation of Table 1.

Edge labels in the SLA graph denote operational and business relationships among SLA node pairs. Edge labels are denoted as "weak" or "strong" to highlight the relationship significance in the SLA processing [16]. Additionally, representative edge-weights are assigned according to the out-degree of the head-node in the directed relationship. Both the classification of edge labels as well as the assignment of edge-weights are indicative to demonstrate the data model possibilities. The graph data model can be extended according to business domain and application specific needs.

4. METHODOLOGY

4.1 Evaluation objectives

The SLA graph data model is evaluated with respect to the following objectives:

Modularity expresses the degree to which the graph elements can be partitioned into sub-graphs that may exist in the network [2]. The skeleton SLA graph has a modularity of 0.392 with an initial partition of 2 communities. In a large SLA graph, we may have service compositions that belong to different topologies. The modularity metric is important, because it can be used for the identification of service element communities that generate more cost or vice versa more added value for the provider.

Data accuracy and **completeness** represent data query metrics with respect to the accuracy and usefulness of a data query result. In the case of the SLA graph, data accuracy measures the correctness of query processing, given a query and its retrieved output. Data completeness measures the convenience of query planning, given a set of queries that need to be answered simultaneously and correctly. The NetworkX Python programming library [9] provides a set of basic breadth- and depth-first search algorithms, which can be used on the graph data model to answer a variety of SLA management questions.

Efficiency and **performance**: The directed graph schema of the SLA data model allows for economic value comparison and efficient service pricing using an appropriate SLA cost model in combination with economic efficiency metrics (e.g. Pareto efficiency[1]). Graph-based practices, like the max-flow min-cut theorem [14] can also be used for the evaluation of resource management efficiency. On the other hand, performance deals with the query processing throughput, when exercised on distributed data repositories that communicate over HTTP.

Dynamic SLA data represent information that is subject to time, resource availability and customer demand. The values of SLOs as well as those of composite and resource metrics are examples of temporal data as they are affected by all previous parameters. The SLA graph model can be transformed into a dynamic network that inherits the topology of the SLA skeleton model and where the property values of SLA vertices and edges are subject to changes with respect to defined time-intervals.

[1] http://en.wikipedia.org/wiki/Pareto_efficiency

The **SLA management orchestration** includes SLA and service elements that are specified by a custom application domain. The representation of such elements and of their relationships through nodes and edges enables the utilization of graph-based practices for their management. For example, the max-flow min-cut theorem [14] can be applied on subgraphs of the SLA data model to simulate optimization problems with respect to the distribution of cloud resources between tenants. The NetworkX programming library [9] provides a Python implementation of the max-flow min-cost algorithm that can be used for the simulation.

4.2 Experimentation design, evaluation tools

To support the experimentation of the SLA graph data model, we create synthetic SLA data using the Zipfian distribution[2]. Unfortunately, current lack of a uniform SLA standard decreases the value of SLAs in cloud markets and hampers the collection of information coming from real SLAs.

For the graph analytics we use the NetworkX programming library [9] and Gephi[3], an open source graph visualization and analysis platform. NetworkX provides numerous graph algorithms (e.g. traversal, clustering, etc), which can be adapted to the experimentation objectives. Gephi, on the other hand, supports a full-featured visualization platform that can be used for testing with dynamic SLA data.

We evaluate the SLA graph with respect to query throughput performance over HTTP using as our scenario the filtering and matching of SLA templates over cloud service marketplaces. Therefore we setup a client-server architecture, where client requests represent customer SLA requirements that are matched with existing SLA templates. The Tornado[4] web server represents a minimal asynchronous network library, which we use for the exchange of query data between multi-threaded client instances and defined data repositories. We run tests using the AllegroGraph RDFstore[5] over virtual resources, where the SLA graph data schema is adapted to the RDF semantics.

5. RESULTS

In [17] we demonstrated that SLA templates can be treated as facets of service provisioning. We simulated a service marketplace, where SLA templates are stored in two different DBMS, MongoDB and MySQL. The service marketplace receives simultaneous HTTP requests in the range of [100, 1'000'000]. The requests content represents customer SLA requirements that are mapped into data queries and processed simultaneously by the respective DBMS backends. The result-sets are sent back to the web clients.

Our work in [17] evaluated the data processing efficiency of the two DBMS over HTTP. It moreover demonstrated the usage of query templates for the handling of incoming HTTP requests, where we simulated the filtering - matching process of customer provisioning requirements. The graph in Figure 2 shows the results for both MySQL and Mongo databases from running with 2, 10 and 20 requested parameters over HTTP. The y-axis represents the average total time for each performed run and the x-axis indicates the

[2] http://docs.scipy.org/doc/numpy/reference/generated/numpy.random.zipf.html

[3] http://gephi.org/

[4] http://www.tornadoweb.org/en/stable/

[5] http://www.franz.com/agraph/allegrograph/

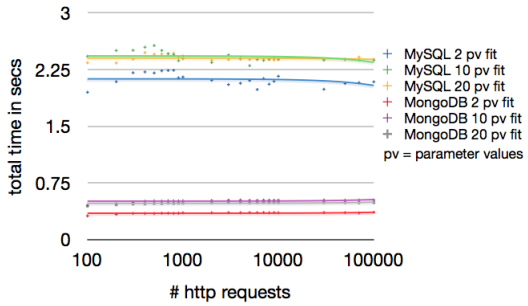

Figure 2: Average total filtering processing time over HTTP: MySQL and MongoDB

gradually increased number of incoming requests that the web server receives. The average time is close to constant for both MySQL and MongoDB. Derived curves for all runs are fitted to highlight the small range of fluctuactions in the query processing results.

In [16], we elaborated on SLA data management characteristics that need to be considered in the design of data models for SLA documents. Our SLA data analysis arranged SLA terms according to data management attributes and operations during the service execution. In [18] we demonstrated a realistic scenario of cloud data service provisioning, where we classified real description attributes of currently offered data services. We then used the SLA graph model, where the collected data service attributes were mapped and formalized an SLA template for a RDBMS service that is managed as a property graph.

6. CONCLUSIONS AND FUTURE WORK

SLAs for IT services currently consist of semi-structured information that is subject to volatile processing and frequent updates. Provided SLAs are unbounded in terms of length and content. A systematic approach to SLA data management is required for the efficient handling of SLA information over virtual, distributed resources.

Our SLA graph data model follows the language semantics and structural element hierarchies that were defined by the WSLA specification [13]. Our approach advances the SLA role from measurement instruments to value-added information that can be used for on-demand service provisioning as well as for use-cases of cost predictive modeling and service-cost behavior analytics.

Current evaluation objectives have been summarized in Section 4. Our future goal is to integrate the evaluated SLA graph data model with a pricing cost model [11] that considers both the service provider revenue objectives as well as the customer QoS criteria.

7. ACKNOWLEDGMENTS

K.Stamou would like to thank Dr. Verena Kantere for her comments. This work is supported by the Swiss National Science Foundation (SNSF), grant number 200021-146603/1.

8. REFERENCES

[1] A. Andrieux et al. Web Services Agreement Specification (WS-Agreement), 2005.

[2] V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10), 2008.

[3] M. Buco et al. Utility computing SLA management based upon business objectives. *IBM Systems Journal*, 2004.

[4] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. Grid information services for distributed resource sharing. In *10th IEEE International Symposium on HPDC*, 2001.

[5] K. Czajkowski, I. Foster, and C. Kesselman. Agreement-Based Resource Management. *Proc. of the IEEE*, 2005.

[6] K. Czajkowski, I. Foster, C. Kesselman, V. Sander, and S. Tuecke. SNAP: A protocol for negotiating service level agreements and coordinating resource management in distributed systems. In *Job scheduling strategies for parallel processing*. 2002.

[7] A. Dan, H. Ludwig, and G. Pacifici. Web service differentiation with service level agreements. *White Paper IBM Corporation*, 2003.

[8] I. Foster, A. Roy, and V. Sander. A quality of service architecture that combines resource reservation and application adaptation. In *8th International Workshop on Quality of Service*, 2000.

[9] A. Hagberg, D. Schult, and P. Swart. NetworkX. `http://networkx.github.io/`. Accessed: March, 2013.

[10] H.Ludwig, A.Dan, and R.Kearney. Cremona: an architecture and library for creation and monitoring of WS-agreements. In *ICSOC*. ACM, 2004.

[11] V. Kantere, D. Dash, G. Francois, S. Kyriakopoulou, and A. Ailamaki. Optimal service pricing for a cloud cache. *IEEE Transactions on Knowledge and Data Engineering*, 23(9):1345–1358, 2011.

[12] A. Keller, U. Blumenthal, and G. Kar. Classification and Computation of Dependencies for Distributed Management. In *Proc. of the Fifth IEEE Symposium on Computers and Communications (ISCC 2000)*, ISCC '00. IEEE Computer Society, 2000.

[13] H. Ludwig et al. Web Service Level Agreement (WSLA) Language Specification, 2003.

[14] C. H. Papadimitriou and . Steiglitz, Kenneth. *Combinatorial optimization : algorithms and complexity / Christos H. Papadimitriou, Kenneth Steiglitz.* Mineola, N.Y. : Dover Publications, 1998.

[15] M. Rodriguez. Property Graph Algorithms. `http://markorodriguez.com/2011/02/08/property-graph-algorithms/`. accessed: July, 2013.

[16] K. Stamou, V. Kantere, and J. Morin. SLA data management criteria. In *Scalable Cloud Data Management Workshop, IEEE BigData Conf. Silicon Valley, October 2013*, 2013.

[17] K. Stamou, V. Kantere, and J. Morin. SLA template filtering: a faceted approach. In *Proc. of the 4th International Conference on Cloud Computing, GRIDs, and Virtualization, IARIA Cloud Computing, Valencia, Spain*, 2013.

[18] K. Stamou, V. Kantere, J. Morin, and M. Georgiou. A SLA Graph Model for Data Services. In *5th International workshop on Cloud Data Management, ACM International Conference on Information and Knowledge Management*, 2013.

[19] P. Stelling, I. Foster, C. Kesselman, C. Lee, and G. von Laszewski. A fault detection service for wide area distributed computations. In *7th IEEE International Symposium on HPDC*, 1998.