SLA-Guided Data Integration on Cloud Environments
Brokering Energy for Providing Sustainable Consumption

Keywords: service level agreement (SLA), data integration, services, cloud, smart energy

Abstract: This paper proposes data integration (lookup, aggregation, correlation) strategies adapted to the vision of the economic model of the cloud such as accepting partial results delivered on demand or under subscription models that can affect the quality of the results; accepting specific data duplication that can respect privacy but ensure data availability; accepting to launch a task that contributes to an integration on a first cloud whose SLA verifies security requirement rather that a more powerful cloud but with less security guarantees in the SLA.

1 INTRODUCTION

The emergence of new architectures like the cloud opens new opportunities to data processing. The possibility of having unlimited access to cloud resources and the "pay as U go" model make it possible to change the hypothesis for processing big data collections. Instead of designing processes and algorithms taking into consideration limitations on resources availability, the cloud sets the focus on the economic cost implied of using resources and producing results by parallelizing their use owhile delivering data under subscription oriented cost models.

Integrating and processing heterogeneous data collections, calls for efficient methods for correlating, associating, filtering them taking into consideration their "structural" characteristics (due to the different data models) but also their quality, e.g., trust, freshness, provenance, partial or total consistency. Existing data integration techniques have to be revisited considering weakly curated and modeled data sets. This can be done according to quality of service requirements expressed by their consumers and Service Level Agreement (SLA) contracts exported by the cloud providers that host these collections and deliver resources for executing the associated management processes.

Our work addresses big data collections integration in a multi-cloud hybrid context guided by user preferences statements and SLA contracts exported by different cloud providers. The objective is to propose an SLA guided continuous data provision and integration system exported as a DaaS by a cloud provider adapted to the vision of the economic model of the cloud such as accepting partial results delivered on demand or under predefined subscription models that can affect the quality of the results; accepting specific data duplication that can respect privacy but ensure data availability; accepting to launch a task that contributes to an integration on a first cloud whose SLA verifies a QoS requirement rather that a more powerful one.

Therefore this paper presents an approach proposing strategies for computing integrated SLAs according to agreed SLAs exported by services and adaptable query rewriting for integrating data sets according to user preference statements. This implies to consider several granularities of SLA: first, at the cloud level; the SLA ensured by providers regarding data; then at the service level, as unit for accessing and processing data, to be sure to fit particular service needs; and finally at the integration level i.e the possibility to process, correlate and integrate big data collections distributed across different cloud storage supports, providing different quality properties to data (trust, privacy, reliability, etc).

Accordingly, the remainder of this paper is organized as follows. Section 2 presents related works that address SLA modelling, integration and SLA guided data management processes. Section 3 gives an overview of our approach for integrating data sets provided by services (i.e., DaaS) by concilating SLA's provided by services and user's profiles expressing QoS preferences about the data they want to consume and the conditions in which they must be processed and delivered. Finally 7 concludes the papers and discusses future work.

2 RELATED WORK

The advent of cloud computing imposed a new model for resource consuming that is not concerned with resource availability but with the *technical and economic* conditions to be fulfilled in order to access a potentially infinite set of resources. Service Level Agreements (SLA) (**?**) can be used in the context of cloud computing to establish those conditions. The deployment of SLAs relies on two principles: (i) The negotiation of conditions, which are statically agreed between the parts and (ii) The monitoring of these conditions during the use of cloud resources in order to detect SLA contract violation.

SLAs are normally expressed in terms of *low level* concepts, like the storage space size or the degree of virtualization. On the user side, the interest is normally on *higher level* concepts such as QoS requirements. The challenge is thus, to couple cloud SLA terms with user ones in order to agree the conditions in which they will interact. Existing works use matching and negociation techniques for addressing this challenge. For instance, (Emeakaroha et al., 2010) proposes a bottom-up approach based on a cloud component that monitors low level measures, analyses and matches them to high level clauses expressed by the user. (Dastjerdi et al., 2012) describes a *Semantic SLA* that can be understood by all parties including providers, consumers, and monitoring services. Their goal is, starting from a high level SLA, to measure and identify how to derive SLA measures at different layers of the cloud. (Ortiz et al., 2013) proposes a set of templates to cloud data services users, each specifying the query type that can be executed with some tread-off in time and corresponding cost. The client chooses among the proposed tempates the one that best corresponds to her requirements. (Chauhan et al., 2011) proposes a set of matching algorithms to identify compatible cloud providers for a given requirements specification by matching SLA parameters. Cloud SLA parameters and application requirements are represented by two models using RDF. RDF definitions are then converted into graph representations. An Induced Propagation Graph is calculated using both models to establish a correspondence between them.

Recent works on SLA are devoted to extend SLA for including security concerns. (Hale and Gamble, 2012) propose an extension of the WSAgreement, initially developed by the GRAAP working group. Security constraints are expressed over the service description terms (SDTs) and the service level objectives (SLOs) of the SLA. This leads to an interoperable security expression that can be used by users

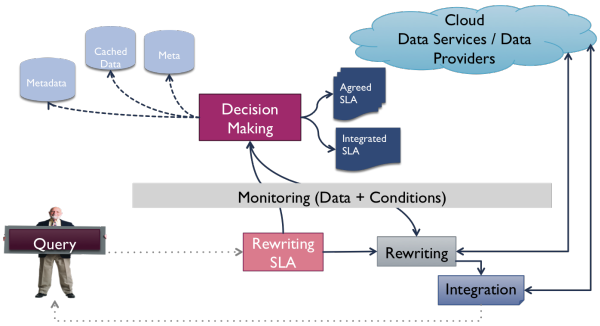

Figure 1: General architecture of an SLA guided data integration system.

for comparing security levels of different cloud service providers. (Luna Garcia et al., 2012) focuses on how to build a SEcSLA Template starting from gathering then categorizing a set of security statements using a semantic tool. The designed template is then used both to express user security requirements and cloud service provider security provision. Finally, some works deal with SLA violation anticipation and SLA failure cascading on violation detection. (Dastjerdi et al., 2012) proposes an SLA dependency model using Web Service Modeling Ontology (WSMO) to build a knowledge database. (Brandic et al., 2010) proposes to anticipate failure by analysing the monitored feature. (Brandic et al., 2010) proposes LAYSI, a layered solution that minimizes user interactions with the system and prevents violations of agreed SLAs.

3 SLA BASED INCREMENTAL DATA INTEGRATION

We propose an SLA guided, continuous data provision and integration approach that consists in three steps starting from the processing of the query to the delivery of the results. Figure 1 shows the general architecture of an SLA guided data integration system. It accesses data services which are data providers deployed in a cloud that provide agreed SLAs. Given a requirement expressing a query and quality of service preferences: cost, provenance, reputation, execution time, the system processes it as follows: (i) derived SLA computation for filtering possible data providers; (ii) query rewriting for computing services compositions that can be used for building results; (iii) managing the integration process including storing partial results, delivering adata and launching and re-launching queries. Intermediate results that are stored as knowledge in order to reduce the overhead of the query evaluation process.

In order to illustrate our approach, consider a smart city that aims at being energy self-sustainable and produce and consume, as much as possible, energy within its geographic area. Producers are characterized according to their location, the amount of energy in Kilowatts-hour that they can sell, the cost of that energy, and the time window in which they can produce it. Consumers are described by their location, their energy requirements during a certain interval of time, the maximum total cost they are ready to pay, and quality of service requirements such as availability and how critical it is to consume this amount of energy. An energy exchange market is established in order to continuously trade energy provision/consumption ensuring that all consumers will have the energy they require at every moment.

Services are deployed in different cloud providers and each service exports an agreed SLA that specifies the economic cost per call, the maximum number of calls that can be done per day, the availability of the service, the average response time when a method is called, the reliability, the privacy of the produced data (whether they can be stored or not), the precision, freshness and provenance of the produced data.

- agreedSLA:⟨cost/call, maxCalls/day, availability, responseTime, reliability, privacy, precision, freshness, provenance⟩.

As said in previous sections, some of these measures (cost/call, maxCall/day) are static and explicitly specified by the service provider. In contrast, the other measures should be computed by monitoring the conversations between the service and the applications that contact it.

In our example assume that there are several energy provision services (e_1, ... e_n) that can be independent or hubs integrating several energy providers. Each of them is specified by a clause:

- $e_i \equiv$ provider(KW-h, Ecost, rate, location), SLA_i

Cloud providers define also their SLA contracts defining normally subscription contracts that specify, the cost per request (cost/request), the volume of data that can be exchanged per month (I/0 volume/month), the cost of transferring data or applications within the same data centre or between data centres (datatransferCost/region), the storage space (storageSpace). For example some cloud providers enable the customer to choose the zone to install PaaS services and deploy applications (e.g. zone 1 is Europe). If the customer wishes to deploy services in zone 1 but store data in zone 2 the transfer cost will change.

- cloudSLA:⟨cost/request, I/0 volume/month, datatransferCost/region, storageSpace⟩.

According to our approach, energy producers are modelled as data services with associated "agreed" SLAs. In the example, we assume that several producers will be able to supply energy for a given period of time given specific preferences expressed by a consumer. Assume that there are four household energy providers that can be queried individually and two hubs that collect information from the community smartmeters. Hubs will store information about (surplus) energy, available from particular users. We represent these services by $e_1, \ldots, e_4$ and h_1 and h_2. We also suppose the existence of two free location services exporting the following interface: loc(IP) $\leftarrow \langle$ X, Y$\rangle$, meaning that given an IP address it returns a geographic position expressed as a pair of coordinates. All these services can potentially be combined for answering queries.

An energy request is expressed as a query that specifies an energy requirement with QoS preferences independently of the possible producers.

Queries may be expressed as Datalog-like programs or an SQL-like expressions, including spatio-temporal attributes and preferences. For instance, *List of energy providers that can provision 1000 Kwatts/h, in the next 10 seconds, that are close to my city with a cost of 0,50 euros/Kwatt and that are labeled as green?*. The user preferences statement would include their cloud provider contract and quality preferences:

- cloudSLA:⟨0,05 cents per call, 8 Gigabytes I/0 volume/-month, free, 1 Giga⟩.
- preferencesStatement:⟨query total cost, total responseTime, reliability, freshness, precision, provenance, storage⟩.

We consider a simplified SLA cloud contract inspired in the lowest contract provided by Azure: cost of $0,05 cents per call, 8 GB of I/O volume/month, free data transfer cost within the same region, 01 GB of storage. The user is ready to pay maximum $5 as total query cost; she requests that only green energy providers are listed (provenance); at least 85% of precision of provided data, even if they are not fresh; she requires an availability rate of at least 90% and with a response time of 0,01 seconds.

As usual, SLAs will be represented as conditions over variables that will be used in the query. In this context, the compliance to the SLAs will drive the query rewriting process. Indeed, our rewriting process will proceed in stages. User preferences and user SLAs will be used to produce derived SLA to be added to the query. These SLAs will influence the choice of data/service providers. The SLA proposed by the chosen providers need to be combined with the conditions on each proposed rewriting of the query. These combination will influence the decision about the actual services to be used.

4 Deriving a query SLA

The key and original aspect of our proposed data integration and provision process is defined as a vertical mapping of user QoS preferences and agreed SLAs. This leads to a *derived SLA* that guides the evaluation of a query.

A query has associated preferences expressed as macroscopic constraints (i.e. user preferences statement): execution time, pay / no pay, data reliability, provenance, freshness, privacy, partial/full results, delivery mode. These constraints are coupled with the profile of the user which is in general stated in her cloud subscription (amount of assigned storage space, number of requests, I/O transfered Mega bytes, etc.).

We assume that services export SLAs (i.e., agreed SLAs) that define measures that can be either expressed as constants, computed (dynamically) by monitoring the execution and conversations associted to services, and hybrid they can be statically stated but they change at execution time. A service agreed SLA is expressed through an XML document using the specification WSLA (Web service level agreement [1]). The service SLA measures that we consider are: response time, availability, price/call, reliability, data production rate. Other measures are associated to the conditions in which the service is called or to the precision and recall of their produced data given a request.

An example of computed measure is the cost of retrieving the list of energy providers within a region with their KWatt cost. The cost is determined by the cost of the calls. This request includes the price of calling a service (e.g., between 0,25 - 0,50 euros depending on the data service), plus the price of data transmision according to the amount of trasnmitted MegaOctets through the network and the type of subscription of the user for using the network.

Given agreed SLA's and a user preferences statement the challenge is to compute a *derived SLA* that maps SLA measures and preferences attributes. The derived SLA is defined as a set of measures that correspond to the user preferences computed as a function of different static, computed and hybrid measures. The *derived SLA* will guide the way the query will be evaluated, and the way results will be computed and delivered.

In the example, some of the user preferences statement measures are used for defining a dervied SLA that, as said in previous section, will guide the evaluation of the query. These measures are defined as a function of the measures used by the agreed SLAs and by the cloud SLA contract.

- query total cost: $\sum_{i=1...n}$ cost(s_i) + data transfer $\leq$ \$5
- total response time: of services + data transfer
- availability: *(of services involved)* $\leq 90\%$,
- freshness: non
- precision: *avg (precision of services involved)* $\leq 85\%$
- provenance: all services involved must be $green$
- storage: *partial results size* $\leq 1Giga$

Therefore, we propose a classification of SLA measures that represents the relationship between fine grained measures used by agreed SLAs and coarse grained measures used in user preferences statements. It specifies also how to compute coarse grained measures with fine grained ones. For example, data precision will be computed as a function of availability, freshness and provenance exported by data services. The derived SLA can be seen as a set of inequations that have to be solved during the execution of a service composition. Since some of them can only be determined at execution time, the decision on which services will participate in the evaluation of the query is approximated. We will discuss this issue in the next sections.

This step may lead either to the rejection of integration in case of total incompatibility, or to a negotiation between SLA which will lead us to the proposal for a negotiated SLA integration and thus the need for an adaptive setting. Negociation of this type of SLA depends strongly on the request sent and the services deployed at the arrival time of the application on the cloud. This negotiation can be expensive and may not scale well.

Given a query and its preferences statement, the system finds service compositions that produce results meeting the required constraints, as discussed in the following section.

5 Query Rewriting

Query rewriting is a well-known problem in the database domain. The problem consists in transforming an abstract query into a set (or list) of lower-level queries that can be solved by available databases. Query rewriting is guided by the schema of abstract and concrete databases. The answers to the lower level queries are combined in order to obtain the result to be returned to the user.

The query rewriting problem can be generalized to the case of services. In this case, the query to be rewritten is seen as an abstract service composition, to be expressed in terms of concrete services. Query

[1] http://www.research.ibm.com/people/a/akeller/-Data/WSLASpecV1-20030128.pdf

rewriting techniques have been adapted to the context of service composition (Barhamgi et al., 2010; Zhao et al., 2011; da Costa et al., 2013). In (da Costa et al., 2013) the authors present an algorithm to automatically refine high-level specifications of service compositions into lower-level ones. The method is based on the MiniCon algorithm (Pottinger and Halevy, 2001) for query rewriting.

The case of more general services (*i.e*, services that maintain and update information) is a generalization of the information-provision case. Unlike the simpler case, where only the service interfaces need to be considered, the general case requires considering the functional and non-functional aspects of the query and available services. In this context, the *Local as View* (*LAV*) methods of query rewriting (Levy, 2000) are suitable. In the LAV approach, the rewriting process is guided by the specification of both the query to be rewritten and the available databases. The specification of the composition to be produced as well as the specification of each available service are used in the case of general services. These specifications may detail both the functional and non-functional behaviour of each service, including SLA.

We propose an approach consists in generating several translations of an abstract service specification into compositions over concrete services. The solutions proposed are ranked and may be coded into concrete workflows as shown in the following section. The next example shows the main features of the approach proposed in (da Costa et al., 2013) and that is extended in the case of SLA guided services based data integration.

Example 1 (Service Refinement by Rewriting) Let us consider the case of the smart city scenario, where particulars can participate on an energy trading pool. We suppose that this hypothetical city has a large number of households that sell their energy surplus to other consumers. The city has four information hubs, located in four different neighbourhoods, connected to very small energy producers. Other producers are be contacted directly, though the services of three different cloud providers. Each cloud provider exports its own interfaces. SLAs are defined for each individual energy provider server, each hub, and each cloud provider.

When an on-line consumer searches servers to buy energy, a composed web service is generated, in order to fetch each individual or hub service and to start energy reservation procedures. Depending on the location of the consumer and producers, different conditions and constraints may apply. Also cloud providers publish the conditions for using their services. These conditions are expressed as user preferences and SLAs. Additionally, other non-functional requirements (such as authentication or security requirements) may apply.

In this context, the user expresses an energy order, her location and payment information. A composite web service is generated to fulfil the order. The generated service composition should consider the nearest providers, in accordance to the agreed SLA.

In order to produce a personalised service composition for each user, the algorithm in (da Costa et al., 2013) takes into account the specification of the composition (which may be produced by the user's browser, including context information). The specification of each available service is also considered (this specification should be given by the service provider). □

Given a a set of services that can possibly be composed and the derived SLA, a service composition must be produced. Some of the inequations of the derived SLA should be included in the service compositions that answer the query. In the case of our query example the following composition can be used for answering it:

$$\begin{aligned} Q_1 \quad \equiv \quad & \langle X_1, Y_1 \rangle = loc_1(IP), \\ & \langle X_2, Y_2 \rangle = loc_1(Eprovider), \\ & D = distance(\langle X_1, Y_1 \rangle, \langle X_2, Y_2 \rangle), \\ & \langle C_1, E_1 \rangle = e_1(\ldots), \ldots \langle C_4, E_4 \rangle = e_4(\ldots), \\ & D \leq 1.5\mathsf{Km}, \;\; sum(C_1 : C_4) \leq 5, \\ & sum(E_1 : E_4) \geq 1000\ \mathsf{kw-h}, \\ & \mathsf{totalResponseTime} \leq 10\ \mathsf{seg} \end{aligned}$$

In fact our approach would generate a number k of service compositions, combining as much as possible the services available such that the constraints of the derived SLA are verified. Yet, the algorithm should be modified to take into account the different SLAs. The next example illustrates one of the limits of the automatic composition algorithm.

Example 2 (Incremental queries) Let us return to the energy trade system of Example 1. Suppose that the user requires to retrieve a list of star providers daily and weekly that can deliver "*One MegaWatthour*". In this case, each individual and hub database will be queried and the list will be produced by adding data obtained from them, until the one MegaWatthour capacity is reached. Let us suppose that, in order to minimize the the communications between servers, the lists need to be produced incrementally. In this case, the composition produced by the service refinement may need to include an iteration (to aggregate partial results). The data produced by the warehouse servers will be processed in batch and the process will end once the list reaches the desired capacity.

To our knowledge, the incremental production of a solution is beyond the scope of the current methods for rewriting service compositions and represents a challenge to the area. □

6 Dealing with the resources consumed by the evaluation process

Our data provision and integration approach relys on data services deployed on one or different providers and it is delivered as a DaaS. This DaaS uses resources from a cloud and this use is guided by an economic model (stated in a cloud subscription) that puts a threshold on the amount of resources to be invested in a query evaluation process. It is thus important to optimize the use of these resources.

We believe that the optimization of this process can occur at two levels: first at the level of agreed SLA exported by services. Indeed, queries requesting the same services compositions will have clauses in their SLAs that are more conditions of use of the infrastructure (ie not storing the data produced by a service). Instead of recomputing the derived SLA every time, we propose to store it and reuse it for other queries.

Second, precalculated queries and partial results can be also stored in cache or in a persistent supprt. Rewritting results can be also stored and reduce execution and resources consumption time when evaluating similar queries. Storing or not such SLAs will depend on the cloud subscription of the user the issued the query (data access, intermediate storage capacity , cost of storage , etc ...).

As discussed in previous sections, the derived SLA associated to service compositions can have free measures that can only be evaluated at run-time. In order to do so, we assume that there is a monitoring system observing and aggregating events for computing resources consumption, execution and time cost, volumes of data transfered when services deliver results. These computations are used dynamically for instanciating free variables in the derived SLA and thus determine wether te contract is respected by the execution of a query. Since this is monitered dynamicalle, the evaluation of the query can be adapted if the SLA is not being respected anymore.

7 CONCLUSIONS AND FUTURE WORK

This paper introduces the challenge of integrating data from distributed data services deployed on different cloud providers guided by service level agreements (SLA) and user preferences statements. The data integration problem is stated as a continuos data provision problem that has an associated economic cost and that uses automatic learning techniques for ensuring different qualities of delivered data (fresh, precise, partial).

Current big data settings impose to consider SLA and different data delivery models. We believe that given the volume and the complexity of query evaluation that includes steps that imply greedy computations, it is important to combine and revisit well-known solutions adapted to these contexts. We are currently developing the strategies and algorithms sketched here applied to energy consumption applications as the one described in the paper and also to elections and political campaign data integration in order to guide decision making on campaign strategies.

REFERENCES

Barhamgi, M., Benslimane, D., and Medjahed, B. (2010). A query rewriting approach for web service composition. *IEEE T. Services Computing*, 3(3):206–222.

Brandic, I., Emeakaroha, V., Maurer, M., Dustdar, S., Acs, S., Kertesz, A., and Kecskemeti, G. (2010). Laysi: A layered approach for sla-violation propagation in self-manageable cloud infrastructures. In *Computer Software and Applications Conference Workshops (COMPSACW), 2010 IEEE 34th Annual*, pages 365–370.

Chauhan, T., Chaudhary, S., Kumar, V., and Bhise, M. (2011). Service level agreement parameter matching in cloud computing. In *Information and Communication Technologies (WICT), 2011 World Congress on*, pages 564–570.

da Costa, U. S., Alves, M. H. F., Musicante, M. A., and Robert, S. (2013). Automatic refinement of service compositions. In Daniel, F., Dolog, P., and Li, Q., editors, *ICWE*, volume 7977 of *Lecture Notes in Computer Science*, pages 400–407. Springer.

Dastjerdi, A. V., Tabatabaei, S. G. H., and Buyya, R. (2012). A dependency-aware ontology-based approach for deploying service level agreement monitoring services in cloud. *Softw. Pract. Exper.*, 42(4):501–518.

Emeakaroha, V., Brandic, I., Maurer, M., and Dustdar, S. (2010). Low level metrics to high level slas - lom2his framework: Bridging the gap between monitored metrics and sla parameters in cloud environments. In *High Performance Computing and Sim-*

ulation (HPCS), 2010 International Conference on, pages 48–54.

Hale, M. and Gamble, R. (2012). Secagreement: Advancing security risk calculations in cloud services. In *Services (SERVICES), 2012 IEEE Eighth World Congress on*, pages 133–140.

Levy, A. Y. (2000). Logic-Based Techniques in Data Integration. In Minker, J., editor, *Logic-Based Artificial Intelligence*, chapter 1, pages 575–595. Kluwer Academic Publishers, Norwell, MA, USA.

Luna Garcia, J., Langenberg, R., and Suri, N. (2012). Benchmarking cloud security level agreements using quantitative policy trees. In *Proceedings of the 2012 ACM Workshop on Cloud computing security workshop*, CCSW '12, pages 103–112, New York, NY, USA. ACM.

Ortiz, J., de Almeida, V. T., and Balazinska, M. (2013). A vision for personalized service level agreements in the cloud. In *Proceedings of the Second Workshop on Data Analytics in the Cloud*, DanaC '13, pages 21–25, New York, NY, USA. ACM.

Pottinger, R. and Halevy, A. Y. (2001). Minicon: A scalable algorithm for answering queries using views. *VLDB J.*, 10(2-3):182–198.

Zhao, W., Liu, C., and Chen, J. (2011). Automatic composition of information-providing web services based on query rewriting. *Science China Information Sciences*, pages 1–17.