Trusted SLA Guided Data Integration On Multi-Cloud Environments

Daniel A. S. Carvalho
(Supervised by Chirine Ghedira-Guegan)
Université Jean Moulin Lyon 3
Centre de Recherche Magellan - IAE
Lyon, France
daniel.carvalho@univ-lyon3.fr

ABSTRACT

Se referir ao projeto como um todo.. This paper introduces the general lines of a rewritring algorithm named *Rhone* that addresses query rewriting for data integration. The originality of *Rhone* is that the rewriting process is guided by quality measures associated to data providers (services) and user preferences. The paper uses a running scenario to describe the *Rhone*'s implementation and gives some hints about its experimental evaluation.

1. INTRODUCTION

Data integration has evolved with the emergence of data services that deliver data under different quality conditions related to data freshness, cost, reliability, availability, among others. Data are produced continuously and on demand in huge quantities and sometimes with few associated metadata, which makes the integration process more challenging. Some approaches express data integration as a service composition problem where given a query the objective is to lookup and compose data services that can contribute to produce a result. Finding the best service composition that can answer a query can be computationally costly. Furthermore, executing the composition can lead to retrieve and process data collections that can require important memory, storage and computing resources. This problem has been addressed in the service-oriented domain [2, 4, 1]. Generally, these solutions deal with query rewriting problems. [2] proposed a query rewriting approach which processes queries on data provider services. [4] introduced a service composition framework to answer preference queries. In that approach, two algorithms based on [2] are presented to rank the best rewritings based on previously computed scores. [1] presented an algorithm that produces and order rewritings according to user preferences. Yet, to our knowledge few works consider quality measures associated both to data services and to user preferences in order to guide the rewriting process.

This paper introduces the early stages of our ongoing work on developing the *Rhone* service-based query rewriting algorithm guided by SLA's. Our work addresses this issue and proposes the algorithm *Rhone* with two original aspects: (i) the user can express her quality preferences and associate them to her queries; and (ii) service's quality aspects defined on Service Level Agreements (SLA) guide service selection and the whole rewriting process.

The remainder of this paper is organized as follows. Section **??** describes the algorithm *Rhone*, proposed in our work. Section **??** describes a running scenario and also implementation issues. Finally, section **??** concludes the paper and discusses our work perspectives.

2. RELATED WORKS

In recent years, the cloud have been the most popular deployment environment for data integration [5]. Existing works addressing this issue can be grouped according to two different lines of research: *(i)* data integration and services [6, 9, 14, 15]; and *(ii)* service level agreements (SLA) and data integration [3, 12].

[6] proposes a query rewriting method for achieving RDF data integration. The objective of the approach is: (i) solve the entity co-reference problem which can lead to ineffective data integration; and (ii) exploit ontology alignments with a particular interest in data manipulation. [9] introduces a system (called SODIM) which combine data integration, service-oriented architecture and distributed processing. The novelty of these approaches is that they perform data integration in service oriented contexts, particularly considering data services. They also take into consideration the requirement of computing resources for integrating data. Thus, they exploit parallel settings for implementation costly data integration processes.

A major concern when integrating data from different sources (services) is privacy that can be associated to the conditions in which integrated data collections are built and shared. [15] focuses on data privacy in order to integrate data. Based on users' integration requirements, the repository supports the retrieval and integration of data across different services. [14] proposes an inter-cloud data integration system that considers a trade-off between users' privacy requirements and the cost for protecting and processing data. According to the users' privacy requirements, the query plan in the cloud repository creates the users' query. This query is subdivided into sub-queries that can be executed in service providers or on a cloud repository. Each option has its

own privacy and processing costs. Thus, the query plan executor decides the best location to execute the sub-query to meet privacy and cost constraints.

Service level agreement (SLA) contracts have been widely adopted in the context of cloud computing. Research contributions mainly concern (i) SLA negotiation phase (step in which the contracts are established between customers and providers) and (ii) monitoring and allocation of cloud resources to detect and avoid SLA violations. [12] proposes a data integration model guided by SLAs in a grid environment. Their work uses SLAs to define database resources. Then, resources can be evaluated (in terms of processing cost, amount of data and price of using the grid) and selected to the integration. A matching algorithm is proposed to produce query plans. The most appropriated solutions based on the QoS are selected as final results. Apart from our previous work [3], to the best of our knowledge, there is no evidence of researches on SLA applied to data integration in a (multi-)cloud context.

The main aspect in a data integration solution is the query rewriting. In the database domain, the query rewriting problem using views have been widely discussed [10, 11, 8, 13]. Similarly, data integration can be seen in the service-oriented domain as a service composition problem in which given a query the objective is to lookup and compose data services that can contribute to produce a result.

Generally, data integration solutions on the service-oriented domain deal with query rewriting problems. [2] proposes a query rewriting approach which processes queries on data provider services. [4] introduces a service composition framework to answer preference queries. Two algorithms inspired on [2] are presented to rank the best rewritings based on previously computed scores. [1] extends [7] and presentes an refinement algorithm based on *MiniCon* that produces and order rewritings according to user preferences and scores used to rank services that should be previously define by the user. In general, these approaches share the same performance problem as the traditional database algorithms. Furthermore, they do not take into consideration user's integration requirements what can lead to produce rewritings that are not satisfactory to the user in terms of quality requirements and cost.

3. CONTRIBUTIONS

Escrever um paragrafo resumindo as contribuicoes pretendidas no projeto e depois em subsecoes falar de alguns pontos principais... approach, model, schema, algoritmo... fazer essa imagem da abordagem no inkscape...

Let us assume the following medical scenario in which users are able to retrieve and integrate data concerning (i) *patients that were infected by a disease;* (ii) *regions most affected by a disease in specific region*; (iii) *patients' personal information*; and (iv) *patients' DNA information.*

The figure 1 illustrates our vision of data integration. Data is delivered as *data services* deployed in a multi-cloud context. Each *data service* and cloud export their SLA specifying the level of services, the available services and their cost, and the access conditions the user can expect. Therefore, given a user query, her integration quality requirements and her cloud subscription, it is rewritten in terms of cloud services (*data services* and *data processing services*) composition that fulfill the integration requirements and deliver the expected results to the user.

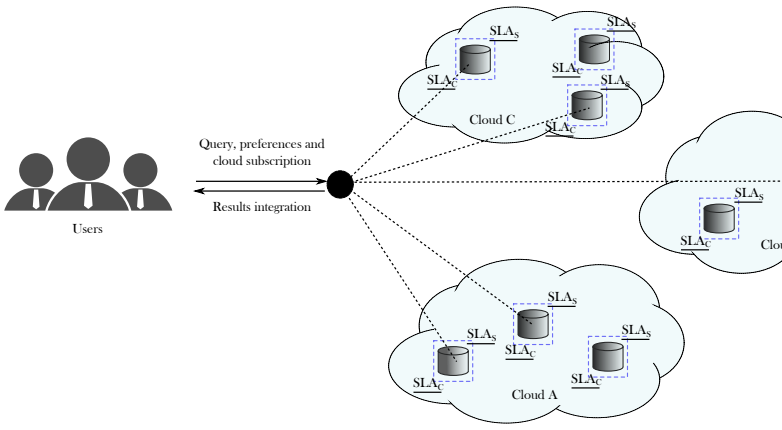

Figure 1: Data integration scenario

Taking into consideration the medical scenario, Doctor *Marcel*'s research is interested in the type of people suffering of *flu* in the Europe. He has at his disposal a set of cloud services delivered by different clouds. To reach his needs, he wants to query the personal information and DNA information from patients that were infected by *flu*, using services with availability higher than 98%, price per call less than 0.2$ and integration total cost less than 5$. This new context brings challenges to data integration, such as:

- **Performance**. In the multi-cloud, the amount of available services is high. Consequently, the time necessary to service compositions (given the high number of services) to the user query is also high.
- **Economic model**. Even with the possibility of having an unlimited access to resources, the user is limited to the resources she has contracted and to the budget she is ready to pay for. Thus, it is necessary to produce the rewritings considering the user integration requirements.
- **Quality**. Part of the rewritings produced to the user query does not satisfy her quality requirements concerning privacy, data provenance, cost, among others. Producing these rewritings implies increasing time processing and cost.
- **Matching problem**. While producing the rewritings, it is necessary to match the user requirements with the different SLAs exported by the cloud providers and cloud services. In the multi-cloud, cloud providers and cloud services export SLAs with different semantics and structure that makes the matching SLA and user requirement challenging. In addition, it also deals with incompatibilities of SLAs.
- **Reuse**. Rewriting and executing the user query is computationally costly in terms of processing time and economic cost. Thus, it is necessary to propose a manner of reusing previous integration in order to save time and money, but also meeting the user expectations.

Thus, motivated by these challenges, a SLA guided data integration approach can be divided in four steps. Given a user query, a set of user preferences associated to it, cloud

providers and cloud services:

SLA derivation. This step creates an *integrated SLA* that includes a set of measures corresponding to the user preferences. The *integrated SLA* guides the query evaluation, and the way results are computed and delivered.

Filtering data services. The *integrated SLA* is used (i) to filter previous SLA derived for a similar request in order to reuse previous results; or (ii) to filter possible data services that can be used for answering the query.

Query rewriting. Given a set of data services that can potentially provide data for integrating the query result, a set of service compositions is generated according to the *integrated SLA* and the agreed SLA of each data service.

Integrating a query result. The service compositions are executed in one or several clouds where the user has a subscription. The execution cost of service compositions must fulfill the *integrated SLA* (that expresses user requirements). Here, the clouds resources needed to execute the composition and how to use them is decided taking in consideration the economic cost determined by the data to be transferred, the number of external calls to services, data storage and delivery cost.

Although *the SLA derivation* is the big challenge while dealing with SLAs and particularly for adding quality dimensions to data integration, the focus in this paper is our query rewriting algorithm which deals with user preferences and SLAs exported by different cloud providers and data services. Here, we are assuming that there is a mechanism responsible to extract the services' quality aspects from SLA, and to provide this information as input to the algorithm. The figure 2 illustrates the structure of SLA and its measures that are considered in the approach we will detail in the next section.

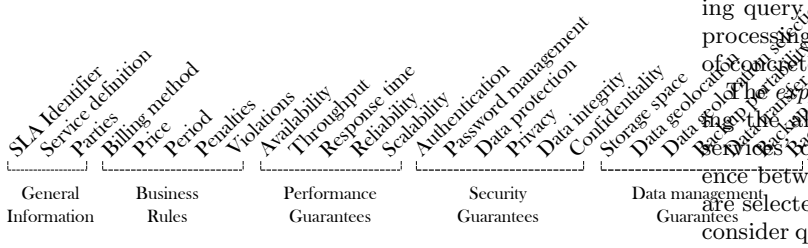

Figure 2: Cloud SLA

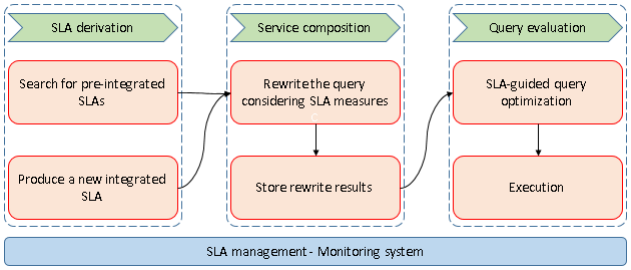

Figure 3: A sample black and white graphic (.pdf format).

4. PRELIMINARY RESULTS

colocar parte dos resultados do adbis + um grafico do custo da reescrita.. This section describes the experiments performed as proof of concept to the algorithm. The Rhone prototype is implemented in Java. It includes 15 java classes in which 14 of them model the basic concepts (*query*, *abstract services*, *concrete services*, etc), and 1 responsible to implement the core of the algorithm.

Currently, our approach runs in a controlled environment. Different experiments were produced to analyze the algorithm's behavior. We will present two experiments: *experiment 1* and *experiment 2*. The service registry used has 100 concrete services. In each experiment, there are a set of tests in which the number of concrete services varies from 5 until to reach 100.

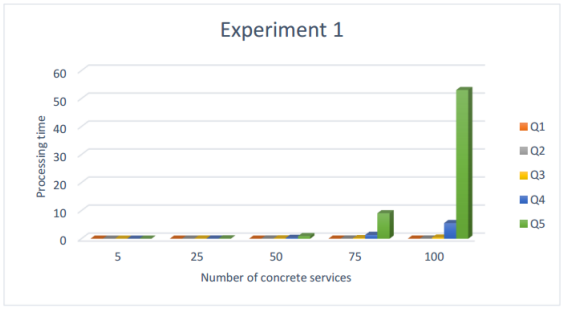

Figure 4: Query rewriting evaluation.

In the *experiment 1* (figure 4), there are five different queries that differ on the quantity of abstract services (increasing from 2 to 6). Analyzing the first experiment, it is easily to identify that the algorithm shares the same problem as existing query rewriting approaches using views: increasing the processing time when the size of the query and the number of concrete services increase.

The *experiment 2* (figure 5) presents the results while testing the algorithm in the presence of user preferences and services' quality aspects extracted from SLAs. The difference between *Test 1* and *Test 2* concerns the way services are selected and the query is rewritten. Once *Test 1* do not consider quality measures as any other existing query rewriting approach, *Test 2* uses the user preferences statements and services' quality aspects to guide the service selection and query rewriting. Both include queries with six abstract services and quality requirements concerning availability, response time, price per call and integration cost (total cost). The figure 5 shows our results.

The results while considering user preferences and SLAs are promisingly. The *Rhone* increases performance reducing rewriting number (around 50 percent) which allows to go straightforward to the rewriting solutions that are satisfactory avoiding any further backtrack and thus reducing successful integration time. Moreover, once the services selection and service composition (rewritings) process are fully guided by the user requirements and SLA, the algorithm avoid producing and executing composition that are not interest for the user. In this sense, the reduces the integration economic cost while delivering the expected results.

5. CONCLUSIONS

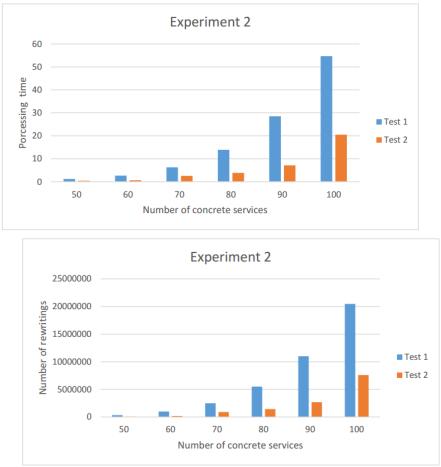

Figure 5: Results concerning processing time (left-side) and rewritings number (right-side).

recapitular as ideias do projeto e o que vamos fazer... This work proposes a query rewriting algorithm for data integration quality named *Rhone*. Given a query, user preferences and a list of concrete services as input, the algorithm derives rewritings in terms of concrete services that matches with the query and fulfill the user preferences. The formalization and experiments are presented. The results show that the *Rhone* reduces the rewriting number and processing time while considering user preferences and services' quality aspects extracted from SLAs to guide the service selection and rewriting. We are currently performing improvements in the implementation and setting up a multi-cloud simulation in in order to evaluate the performance of the *Rhone* in such context.

6. REFERENCES

[1] C. Ba, U. Costa, M. H. Ferrari, R. Ferre, M. A. Musicante, V. Peralta, and S. Robert. Preference-driven refinement of service compositions. In *Int. Conf. on Cloud Computing and Services Science, 2014*, Proceedings of CLOSER 2014, 2014.

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

[3] N. Bennani, C. Ghedira-Guegan, M. Musicante, and G. Vargas-Solar. Sla-guided data integration on cloud environments. In *2014 IEEE 7th International Conference on Cloud Computing (CLOUD)*, pages 934–935, June 2014.

[4] K. Benouaret, D. Benslimane, A. Hadjali, and M. Barhamgi. FuDoCS: A Web Service Composition System Based on Fuzzy Dominance for Preference Query Answering, Sept. 2011. VLDB - 37th International Conference on Very Large Data Bases - Demo Paper.

[5] D. A. S. Carvalho, P. A. Souza Neto, G. Vargas-Solar, N. Bennani, and C. Ghedira. *Database and Expert Systems Applications: 26th International Conference, DEXA 2015, Valencia, Spain, September 1-4, 2015, Proceedings, Part II*, chapter Can Data Integration Quality Be Enhanced on Multi-cloud Using SLA?, pages 145–152. Springer International Publishing, 2015.

[6] G. Correndo, M. Salvadores, I. Millard, H. Glaser, and N. Shadbolt. SPARQL query rewriting for implementing data integration over linked data. In *Proceedings of the 1st International Workshop on Data Semantics - DataSem '10*, page 1, New York, New York, USA, Mar. 2010. ACM Press.

[7] U. Costa, M. Ferrari, M. Musicante, and S. Robert. Automatic refinement of service compositions. In F. Daniel, P. Dolog, and Q. Li, editors, *Web Engineering*, volume 7977 of *Lecture Notes in Computer Science*, pages 400–407. Springer Berlin Heidelberg, 2013.

[8] O. M. Duschka and M. R. Genesereth. Answering recursive queries using views. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, PODS '97, pages 109–116, New York, NY, USA, 1997. ACM.

[9] G. ElSheikh, M. Y. ElNainay, S. ElShehaby, and M. S. Abougabal. SODIM: Service Oriented Data Integration based on MapReduce. *Alexandria Engineering Journal*, 52(3):313–318, Sept. 2013.

[10] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, Dec. 2001.

[11] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proceedings of the 22th International Conference on Very Large Data Bases*, VLDB '96, pages 251–262, San Francisco, CA, USA, 1996. Morgan Kaufmann Publishers Inc.

[12] T. Nie, G. Wang, D. Shen, M. Li, and G. Yu. Sla-based data integration on database grids. In *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, volume 2, pages 613–618, July 2007.

[13] R. Pottinger and A. Halevy. Minicon: A scalable algorithm for answering queries using views. *The VLDB Journal*, 10(2-3):182–198, Sept. 2001.

[14] Y. Tian, B. Song, J. Park, and E. nam Huh. Inter-cloud data integration system considering privacy and cost. In J.-S. Pan, S.-M. Chen, and N. T. Nguyen, editors, *ICCCI (1)*, volume 6421 of *Lecture Notes in Computer Science*, pages 195–204. Springer, 2010.

[15] S. S. Yau and Y. Yin. A privacy preserving repository for data integration across data sharing services. *IEEE T. Services Computing*, 1(3):130–140, 2008.