Quality-Based Data Integration On Cloud Environments

Daniel Aguiar da Silva Carvalho
(Supervised by Chirine Ghedira-Guegan, Genoveva Vargas-Solar and Nadia Bennani,
with inputs from Plácido A. Souza Neto)
Université Jean Moulin Lyon 3, Centre de Recherche Magellan - IAE, Lyon - France
daniel.carvalho@univ-lyon3.fr

ABSTRACT

In the traditional databases theory, data integration is a problem of merging data from data sources in order to provide a unified view of this data to the user. This PhD project address data integration in multi-cloud environments. Current data integration systems implies consuming data from data services deployed in cloud contexts and integrating the results. The project takes a new angle of the problem by proposing an SLA-based data integration approach in a multi-cloud environment which considers users' integration preferences (including quality constrains and data requirements) to be matched with the services' quality constraints extracted from their SLAs while delivering the results. The objective is to enhance the quality and performance on data integration taking into consideration the economic model imposed by the cloud. At this stage, Our first results have shown that quality and performance can be enhanced, and the cost of the integration can be minimized by using our approach.

1. INTRODUCTION

In recent years, the cloud has been the most popular deployment environment for data integration [2]. 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. However, the lack of an associated meta-data describing the data, currently produced in huge quantities, makes the integration process more challenging. Data integration in service-oriented architectures can be seen 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 [1, 3, 4, 6, 7].

[1] described their approach for modeling, discovering, selecting and composing data services while enforcing providers' privacy and security policies. [3] proposed a rewriting method based on SPARQL for RDF data integration. The work focused on avoiding ineffective data integration by solving the entity co-reference problem. [4] introduced SODIM, a system which combines data integration, service-oriented architecture and MapReduce distributed processing. [7] presented a solution focusing on data privacy in order to integrate data. [6] developed 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' requirements, the query is created and executed meeting privacy and cost constraints. The novelty of these approaches is that they perform data integration in service-oriented and cloud contexts, particularly considering data services. They also take into consideration the requirement of computing resources for integrating data. Thus, they exploit parallel settings for implementing costly data integration processes. They are mainly focused on performance and privacy aspects putting aside other users' integration requirements such as data provenance, data integrity, confidentiality, reliability, availability, whether she wants to use free services, how much she is ready to pay for the integration, among others. Moreover, even with the cloud on-demand resources provisioning (which implies an associated cost), the user is limited to her cloud subscription and maximum budget she is ready to pay for her desired integration.

In cloud computing, the quality conditions under which services are delivered to users are agreed in contracts called service level agreements (SLA). We strongly believe that SLAs can be re-modeled in order (i) to cover the limitations regarding the users' integration requirements (including quality constraints and data requirements); and (ii) to enhance the quality in the current data integration solutions. Works on SLA in cloud computing mainly concern (i) the *negotiation* of contracts between customers and providers; and (ii) *monitoring* of cloud resources to detect SLA violations. Yet, to the best of our knowledge, few works considers SLAs in order to guide the entire data integration in multi-cloud environments. Similarly to our idea, [5] proposed a SLA-based data integration model for grid environments. The approach uses SLAs to define database resources and evaluated them in terms of processing cost, amount of data and price of using the grid. In addition, a matching algorithm is proposed to produce query plans using the selected resources.

In multi-cloud context, current SLA models are not sufficient to cover the data integration requirements. Usually, SLAs are related to cloud resources, and not to the services. Thus, new models should be defined to include users' integration preferences concerning data requirements and qual-

ity constraints. In summary, the objectives of this PhD project contribute as following: (1) design of SLA model for data integration; (2) proposal of a data integration approach adapted to the vision of the economic model of the cloud. The originality of our approach consists in guiding the entire data integration solution taking into account (i) user preferences statements; (ii) SLA contracts exported by different cloud providers; and (iii) several QoS measures associated to data collections properties (for instance, trust, privacy, economic cost); and (3) validation of our approach in a multi-cloud scenario.

2. NEW VISION OF DATA INTEGRATION

In our work, we consider a new vision of data integration in a multi-cloud context (see figure 1). Data are accessed and retrieved as *data services* deployed in *cloud providers* geographically distributed. *Data services* and *cloud providers* export their SLA specifying the level of services they can guarantee to users. Two types of SLA are considered: (1) cloud SLA (SLA_C), agreements between a *data service* and a *cloud provider*; and (2) service SLA (SLA_S), agreements between *users* and *data services*. The figure 2 illustrates a cloud SLA schema.

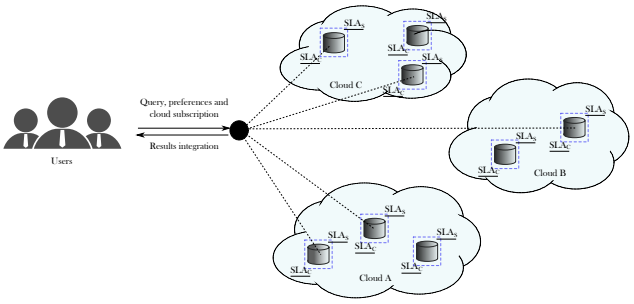

Figure 1: Data integration scenario

Therefore, given a user query, his/her integration quality requirements and cloud subscription, the query 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. This new vision brings challenges to data integration, such as: (i) Performance issues. Given the huge amount of *data services*, *data processing services* present in the multi-cloud context, the data integration tasks can require a huge amount of cloud resources and processing time; (ii) Economic model. The cloud provides resources in an on-demand and pay-per-use way to cloud consumers. However they are limited to the resources they have contracted and to the budget they are ready to pay for. Thus, it is crucial to design new methods for data integration adapted to this new context; (iii) Quality issues. Some rewritings produced and executed to the user query could not satisfy his/her quality requirements concerning privacy, data provenance, cost, among others. Consequently, users may become dissatisfied with the obtained results; (iv) SLA heterogeneity. In the multi-cloud, *cloud services* and *cloud providers* export their SLAs with different semantics and structure making the matching of users' integration requirements with the SLAs more challenging. Moreover, while integrating services, we can also face incompatibilities of SLAs such as measures with the same meaning but with different names, measures with the same name but with different associated values and units, among others; and (iv) 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.

Motivated by these challenges, our data integration approach adapted to the multi-cloud is detailed in the next sections.

2.1 SLA-based data integration

We propose our SLA-based data integration approach which includes (i) looking for services that can be used as *data services* and *data processing services*, the last are responsible to retrieve and integrate data; (ii) processing data retrieval and integration; and (iii) delivering results to the user considering her quality requirements, context and resource consumption. In this sense, our approach is divided in four steps as illustrated in the figure 3.

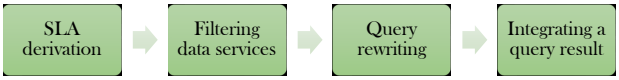

Figure 3: SLA-based data integration workflow.

Given a user *query*, a set of associated user *preferences*, *cloud providers* and *cloud services*:

SLA derivation. In this step, we compute what we call an *derived SLA* that matches user' integration requirements (including quality constraints and data requirements) with the SLA's provided by *cloud services*, given a specific user cloud subscription. The user may have general *preferences* depending on the context she wants to integrate her data such as economic cost, bandwidth limit, free services, and storage and processing limits. The *SLA derivation* is the big challenge while dealing with SLAs and particularly for adding quality dimensions to data integration. Furthermore, the *derived SLA* guides the query evaluation, and the way results are computed and delivered.

Filtering data services. The *derived SLA* is used (i) to filter previous *integration SLA* derived for a similar request in order to reuse results; or (ii) to filter possible *cloud services* that can be used for answering the query. The SLA exported by a selected *cloud service* should satisfy the user *preferences*.

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 *derived SLA* and the agreed SLA of each *data services*.

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 *derived SLA*. 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.

The algorithm 1 describes in general lines our approach which integrate data on multi-cloud environments considering quality aspects. As input, we consider (1) a user query

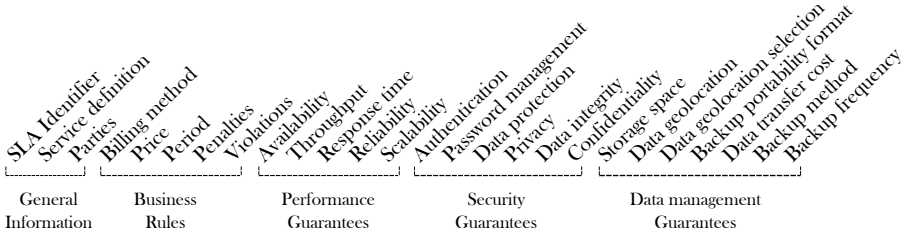

Figure 2: Cloud SLA model

Q; (2) a set of user preferences P; (3) a user cloud subscription C; (4) a set of previous *integrated SLA* $\mathcal{I}_{\mathcal{SLA}}$; (5) a set of *data services* DS; (6) a set of *data processing services* DPS; and (7) a set of *cloud providers* CP. As output, the integration results IR is delivered to the user while fulfilling her preferences and cloud subscription.

The approach begins looking for previous *integrated SLAs* in our registry that can be matched with the user' query and preferences (lines 2 - 6), and adding them to the set pI (line 4). If previous *integrated SLA* are identified (line 7), the one that perfectly matches with the user preferences is selected (line 8). Then, a new *integrated SLA* is computed to user query based on the previous *integrated SLA*, her preferences and cloud subscription (line 9). The service compositions produced to a previous query is reused based its *integrated SLA* (sI). Otherwise, a new *integrated SLA* is created to user request given the query, her preferences and cloud subscription (line 11). The SLA information of DS and DPS is extracted and included in the set of $\mathcal{S}$(line 12). This new *integrated SLA* is updated by calling the *Rhone* procedure (line 13). The *Rhone* is responsible of (i) selecting *data services* (in DS) and *data processing services* (in DPS) which satisfy the user preferences based on their SLAs; and (ii) producing service compositions that completely match with the query and fulfill the user preferences. Finally, the integration results (IR) are produced and delivered to the user considering her cloud subscription. Note that while executing compositions, it is necessary to satisfy the user requirements and cloud subscription, and also to dynamically update the *integrated SLA* adding the information of contracts that were established to allow the integration. Finally, the *integrated SLA* is stored in our registry to be used in a next query.

2.2 SLA-based query rewriting algorithm

To serve a proof of concept to our approach, we develop a query rewriting algorithm which is guided by users' integration requirements and service level agreements exported by different data services and cloud providers. Researches have refereed to data integration 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. Currently, we have formalized and developed a rewriting algorithm that considers user preferences and services' quality aspects while selecting services and producing rewritings called *Rhone* (See algorithm 2, invoked in the algorithm 1, line 12). The *Rhone* algorithm consists in four macro steps:

Selecting data services. The *Rhone* looks for *data services* (line 2) that can contribute to answer the query while satisfying the user' preferences;

Algorithm 1 - SLA-based data integration

Input: Query (Q), user preferences (P), user cloud subscription (C), a set of *data services* (DS) and *data processing services* (DPS) and a set of previous *integrated SLA* ($\mathcal{I}_{\mathcal{SLA}}$).
Output: Integration results delivered considering the user cloud subscription.

```
1: pI ← ∅
2: for all SLA_i in I_SLA do
3:    if macthes(Q, P, SLA_i) then
4:       pI ← pI ∪ {SLA_i}
5:    end if
6: end for
7: if pI ≠ ∅ then
8:    sI ← chooseBest(P, pI)
9:    newSLA_i ← deriveSLA(sI, P, C)
10: else
11:    newSLA_i ← deriveSLA(Q, P, C)
12:    S ← extract(DS, DPS)
13:    newSLA_i ← updateSLA(Rhone(Q, S))
14: end if
15: IR ← execute(newSLA_i)
16: return IR
17: end function
```

Building candidate service descriptions. A candidate service description (CSD) describes how a *data service* can contribute to answer the query. CSD maps a *data service* (including its variables) to the entire query or part of it. For all selected services, the *Rhone* tries to build CSDs when the mapping of variables is possible (line 3).

Combining CSDs. All CSDs are combined (line 4) producing a list of CSD combinations. Each combination can be a rewriting of the query.

Producing rewritings. Each list of CSDs is checked in order to verify whether it is a rewriting of the query or not (line 9). To be a valid rewriting, it is forbidden to have CSDs covering the same part of the query. Rewritings are produced while the user' requirements are being respected (lines 6-14). The originality of our algorithm concerns the functions $\mathcal{T}_{\text{init}}[\![\ ]\!]$, $\mathcal{T}_{\text{cond}}[\![\ ]\!]$ and $\mathcal{T}_{\text{inc}}[\![\ ]\!]$. They are responsible to initialize, check and increment user preferences associated to the integration (such as the total cost). This means that these preferences are initialized (line 6), and for each element in the CSD list they are incremented (line 11), and rewritings are produced while they are respected (line 8). The result of this step is the list of valid rewritings of the query (line 14).

Algorithm 2 - *Rhone*

Input: A query Q, a set of user preferences, and a set of concrete services $\mathcal{S}$.
Output: A set of rewritings R that matches with the query and fulfill the user preferences.

function $Rhone(Q, \mathcal{S})$
$\mathcal{L}_{\mathcal{S}} \leftarrow SelectCandidateServices(Q, \mathcal{S})$
$\mathcal{L}_{\mathcal{CSD}} \leftarrow CreateCSDs(Q, \mathcal{L}_{\mathcal{S}})$
$I \leftarrow CombineCSDs(\mathcal{L}_{\mathcal{CSD}})$
$R \leftarrow \emptyset$
$\mathcal{T}_{\text{init}}[\![\ \mathcal{A}gg(Q)\ ]\!]$
$p \leftarrow I.next()$
while $p \neq \emptyset$ **and** $\mathcal{T}_{\text{cond}}[\![\ \mathcal{A}gg(Q)\ ]\!]$ **do**
  if $isRewriting(Q, p)$ **then**
    $R \leftarrow R \cup Rewriting(p)$
    $\mathcal{T}_{\text{inc}}[\![\ \mathcal{A}gg(Q)\ ]\!]$
  end if
  $p \leftarrow I.next()$
end while
return R
end function

3. PRELIMINARY RESULTS

We have developed a prototype of our query rewriting algorithm which takes into consideration users' requirements and services' quality aspects extracted from SLAs. The *Rhone* prototype is implemented in Java.

Currently, our approach runs in a local controlled environment simulating a mono-cloud including a service registry of 100 concrete services. Experiments were produced to analyze the algorithm's behavior concerning performance, and quality and cost of the integration. The figures 4 and 5 summarize our first results.

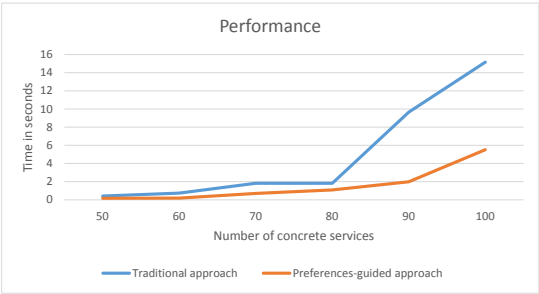

Figure 4: Performance evaluation.

The experiments include two different approaches: (i) the *tradition approach* in which user preferences and SLAs are not considered; and (ii) the *preference-guided approach* which considers the users' integration requirements and SLAs.

The results while considering user preferences and SLAs are promisingly. The *Rhone* increases performance reducing rewriting number which allows to go straightforward to the rewriting solutions that are satisfactory avoiding any further backtrack and thus reducing successful integration time (Figure 4). Moreover, using the *preferences-guided approach* to meet the user preferences, the quality of the rewritings produced has enhanced and the integration economic cost has considerable reduced while delivering the expected results (Figure 5).

4. CONCLUSIONS AND FUTURE WORKS

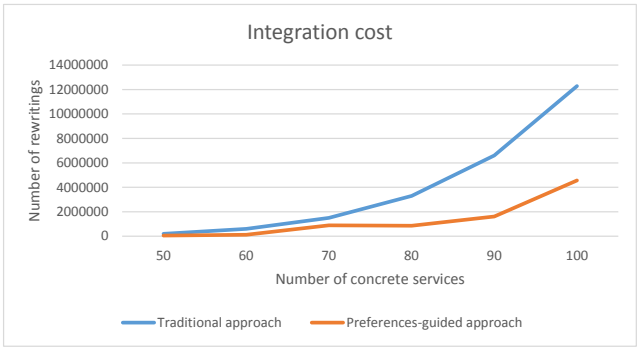

Figure 5: Performance evaluation.

This paper introduces a new vision of data integration adapted to the cloud economic model. It also proposes a new approach for data integration based on user integration requirements and SLA. In addition, an query rewriting algorithm called *Rhone* serves as proof of concept. Our preliminary results have shown 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. Furthermore, the integration quality is enhanced, and it is adapted to cloud economic model reducing the total cost of the integration.

SLA incompatibilities are not treated in this paper. Currently we are working on this issue, and improving our SLA model and schema for data integration adapted to the multi-cloud context. In addition, we are focusing on evaluating the proposed approach on a multi-cloud environment. Then, we will validate our entire quality-based data integration approach.

5. REFERENCES

[1] D. Benslimane, M. Barhamgi, F. Cuppens, F. Morvan, B. Defude, and E. Nageba. Pairse: A privacy-preserving service-oriented data integration system. *SIGMOD Rec.*, 42(3):42–47, Oct. 2013.

[2] D. A. S. Carvalho, P. A. Souza Neto, G. Vargas-Solar, N. Bennani, and C. Ghedira. *Can Data Integration Quality Be Enhanced on Multi-cloud Using SLA?*, pages 145–152. DEXA 2015, Proceedings, Part II. Springer International Publishing, 2015.

[3] 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.

[4] 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.

[5] T. Nie, G. Wang, D. Shen, M. Li, and G. Yu. Sla-based data integration on database grids. In *COMPSAC 2007*, volume 2, pages 613–618, July 2007.

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

[7] 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.