Web Semantics: Science, Services and Agents on the World Wide Web 42 (2017) 1–18

Contents lists available at ScienceDirect

Web Semantics: Science, Services and Agents on the World Wide Web

journal homepage: www.elsevier.com/locate/websem

Decomposing federated queries in presence of replicated fragments

Gabriela Montoya [a,*], Hala Skaf-Molli [a], Pascal Molli [a], Maria-Esther Vidal [b]

[a] LINA – UFR de Sciences et Techniques, Nantes University, 2, rue de la Houssinière. 44322 NANTES CEDEX 3, France

[b] Universidad Simón Bolívar, Baruta, Edo. Miranda - Apartado 89000 Cable Unibolivar Caracas, Venezuela

ARTICLE INFO

Article history:
Received 1 March 2016
Received in revised form
30 October 2016
Accepted 17 December 2016
Available online 21 December 2016

Keywords:
Linked data
Federated query processing
Query decomposition
Fragment replication

ABSTRACT

Federated query engines allow for linked data consumption using SPARQL endpoints. Replicating data fragments from different sources enables data re-organization and provides the basis for more effective and efficient federated query processing. However, existing federated query engines are not designed to support replication. In this paper, we propose a replication-aware framework named LILAC, sparq**L** query decompos**I**tion against federations of rep**L**ic**A**ted data sour**C**es, that relies on replicated fragment descriptions to accurately identify sources that provide replicated data. We defined the query decomposition problem with fragment replication (QDP-FR). QDP-FR corresponds to the problem of finding the sub-queries to be sent to the endpoints that allows the federated query engine to compute the query answer, while the number of tuples to be transferred from endpoints to the federated query engine is minimized. An approximation of QDP-FR is implemented by the LILAC replication-aware query decomposition algorithm. Further, LILAC techniques have been included in the state-of-the-art federated query engines FedX and ANAPSID to evaluate the benefits of the proposed source selection and query decomposition techniques in different engines. Experimental results suggest that LILAC efficiently solves QDP-FR and is able to reduce the number of transferred tuples and the execution time of the studied engines.

1. Introduction

Billions of RDF triples have been made accessible through SPARQL endpoints by data providers.[1] Recent studies reveal unreliability and unavailability of existing public SPARQL endpoints [1]. According to the SPARQLES monitoring system [2] less than a third out of the 545 studied public endpoints exhibits an availability rate of 99%–100% (values for November 2015).

Traditionally in distributed databases, fragmentation and replication techniques have been used to improve data availability [3]. Distributed database administrators are able to design the fragmentation and replication schema according to the applications and the expected workload. The Linking Open Data (LOD) cloud [4] datasets are published by autonomous data providers. Hence fragmentation and replication schema cannot be designed. Clearly, any data provider can partially or totally replicate datasets from other data providers. The LOD Cloud Cache SPARQL endpoint[2] is an example of an endpoint that provides access to total replicas of several datasets. DBpedia live[3] allows a third party to replicate DBpedia live changes in almost real-time. Data consumers may also replicate RDF datasets for efficient and reliable execution of their applications. However, given the size of the LOD cloud datasets, data consumers may just replicate subsets of RDF datasets or replicated fragments in a way that their applications can be efficiently executed. Partial replication allows for speeding up query execution time. Partial replication can be facilitated by data providers, e.g., DBpedia 2016-04[4] consists of over seventy dump files each of them providing different fragments of the same dataset, or can be facilitated by third party systems. Publish–Subscribe systems such as sparqlPuSH [5] or iRap RDF Update Propagation Framework [6] allow to partially replicate datasets. Additionally, data consumers are also autonomous and can declare federations composed of any set of SPARQL endpoints to execute their federated queries.

* Corresponding author.
E-mail addresses: gabriela.montoya@univ-nantes.fr (G. Montoya), hala.skaf@univ-nantes.fr (H. Skaf-Molli), pascal.molli@univ-nantes.fr (P. Molli), mvidal@ldc.usb.ve (M.-E. Vidal).

1 http://stats.lod2.eu.

2 http://lod2.openlinksw.com/sparql.

3 http://live.dbpedia.org/.

4 http://downloads.dbpedia.org/2016-04/core-i18n/en/.

http://dx.doi.org/10.1016/j.websem.2016.12.001
1570-8268/

Consequently, partial or total replication can exist in federations of SPARQL endpoints; replicated data can be at different levels of consistency [7]; and a federated query engine has to be aware of the replication at runtime in order to efficiently produce correct answers. On one hand, if a federated query engine is unaware of data replication, the engine performance may be negatively impacted whenever replicated data is collected from the available sources [8,9]. On the other hand, if a federated query engine is aware of data replication, sources can be efficiently selected [10, 9] and data localities created by replication can be exploited [8] to significantly speed up federated query processing. These data localities, created by endpoints with replicated fragments from different datasets but relevant for federated queries, are not attainable by data providers.

Exploiting replicas can be beneficial. However, replicating data has the intrinsic problem of ensuring data consistency. Traditionally in distributed databases, replicas can be strongly consistent thanks to distributed transactions [3]. However, in the case of the Web, there is no mechanism to ensure that all the available data are strongly consistent. Regarding consistency of replicas, we have identified three main scenarios.

- First, if specific dataset versions are replicated, then the replicas are always perfectly synchronized, e.g., a replica of DBpedia 3.8 is always perfectly synchronized with DBpedia 3.8. This case is especially pertinent when a federated query is defined on a particular version of the available datasets in order to ensure reproducible results.
- Second, if the replicated data is locally modified, the local data is no longer a replica. For instance, if processes of data quality assessment are performed by a data consumer on a replica of DBpedia 3.8, changes to the replica need to be evaluated based on the trustiness of the data consumer. Clearly, data consumers have to change their federation according with what source they trust.
- Third, if the most up-to-date dataset versions are used, because strong consistency cannot be ensured in the context of LOD cloud datasets, replicas may be unsynchronized during query execution. Therefore, it is possible that some of the replicas used to evaluate the query have not integrated all the latest changes before queries are executed. This third scenario can be handled by measuring the replica divergence and the divergence incurred by the sources used to evaluate the query. Out of date replicas can be pruned during the source selection as proposed in [11].

In this paper, for the sake of simplicity, we work under the assumption that replicas are perfectly synchronized as in the first scenario and focus on query processing optimization under this assumption.

Query processing against sources with replicated data has been addressed in [8], while the related problem of query processing against sources with duplicated data has been tackled in [10,9]. These three approaches prune redundant sources at source selection time. This selection may prevent the decomposer from assigning joins between a group of triple patterns to the same endpoint(s), even if this choice produces the most selective sub-query. To illustrate, consider a BGP with three triple patterns $tp1$, $tp2$, and $tp3$. Suppose a SPARQL endpoint $C1$ is relevant for $tp1$ and $tp3$, while $C2$ is relevant for $tp1$ and $tp2$. The source selection strategies implemented by these approaches, will prevent from assigning $tp1.tp3$ to $C1$ and $tp1.tp2$ to $C2$, even if these sub-queries generate less intermediate results. Consequently, as we show in Section 2, managing replication only at source selection time may impede a query decomposer to generate the most selective sub-queries.

In this paper, we exploit the replication knowledge during query decomposition, to generate query decompositions where the limitations of existing replication-aware source selection approaches are overcome. Furthermore, we formalize the query decomposition problem with fragment replication (QDP-FR). QDP-FR corresponds to the problem of finding the sub-queries to be sent to the endpoints that allow the federated query engine to compute the query answer, while the number of tuples to be transferred from the endpoints is minimized. We also propose an approximate solution to QDP-FR, called LILAC, sparq**L** query decompos**I**tion against federations of rep**L**ic**A**ted data sour**C**es, that decomposes SPARQL queries and ensures complete and sound query answers, while reducing the number of transferred tuples from the endpoints.

Specifically, the contributions of this paper are:

- We outline the limitations of solving the source selection problem independently of the query decomposition problem in the context of replicated and fragmented data. We propose an approach where these two federated query processing tasks should be interleaved to support engines in finding better execution plans.
- Based on the replication-aware framework introduced in [8], we propose a query decomposition strategy that relies on this framework to exploit fragments replicated by various endpoints.
- We formalize the query decomposition problem with fragment replication (QDP-FR).
- We propose a sound and complete algorithm to solve the QDP-FR problem.
- We reduce the QDP-FR problem to the set covering problem and use existing set covering heuristics to produce *good* solutions to the QDP-FR problem.
- We extend federated query engines FedX and ANAPSID to perform LILAC query decomposition, i.e., we extend the engines and create the new engines LILAC + FedX and LILAC + ANAPSID. We study the performance of these engines and compare them with existing engines FedX, DAW + FedX, Fedra + FedX, ANAPSID, DAW + ANAPSID, and Fedra + ANAPSID. Results suggest that query decompositions produced by LILAC contribute to reduce the number of transferred tuples and the query execution time.

The paper is organized as follows. Section 2 provides background and motivation. Section 3 defines replicated fragments and presents the query decomposition problem for fragment replication. Section 4 presents the LILAC source selection and query decomposition algorithm. Section 5 reports our experimental results. Section 6 summarizes related works. Finally, conclusions and future work are outlined in Section 7.

2. Background and motivation

In this section, we illustrate the impact that exploiting metadata about replicated fragments has on federated query processing. First, we assume that data consumers replicate fragments composed of RDF triples that satisfy a given triple pattern; URIs in the original RDF dataset are kept for all the replicated resources. In Fig. 1(a), a fragment of DBpedia is illustrated. This fragment comprises RDF triples that satisfy the triple pattern `?film dbo:director ?director`; triples included in Fig. 1(a) correspond to a copy of the DBpedia RDF triples where URIs are preserved. Fragments are described using a 2-tuple *fd* that indicates the authoritative source of the fragment, e.g., DBpedia; the triple pattern that is met by the triples in the fragment is also included in *fd*, e.g., `?film dbo:director ?director`.

Fig. 1(b) depicts a federation of three SPARQL endpoints: $C1$, $C2$, and $C3$; these endpoints expose seven replicated fragments from DBpedia and LinkedMDB. Replicated fragments correspond