A Federation Layer for Query Processing over the Web of Linked Data

Xuejin Li[1], Zhendong Niu[1], Chunxia Zhang[2], and Junyue Cao[1]

[1] School of Computer Science, Beijing Institute of Technology
{xuejinli7,junyuecao}@gmail.com, zniu@bit.edu.cn
[2] School of Software, Beijing Institute of Technology
cxzhang@bit.edu.cn

Abstract. Today, the Web of Linked Data has grown considerably. Transparent query access over multiple Linked Data sources is a key challenge for many semantic applications. In this contribution, we present a query federation for executing distributed SPARQL queries on Linked Data. A sample semantic application is used to demonstrate the practicability and efficiency of the presented architecture.

Keywords: Linked Data, Semantic Web, Query Federation.

1 Introduction

In recent years, the World Wide Web has evolved from a global information space of linked documents to one where both documents and data are linked [1]. The Web of Linked Data enables new types of applications which can aggregate data from different data sources and integrate fragmentary information from multiple sources to achieve a more complete view. Answering queries across multiple distributed Linked Data sources is a key challenge for this possibility.

As a reaction to this challenge, researchers have proposed many solutions [2–5]. These approaches can be divided into three main categories: central approach, link traversal based approach and query federation. With the ever-increasing amount of data sources accessible via SPARQL endpoints, federated query processing has attracted more and more attentions [6–8]. We outline two key factors concerning the performance of federated query systems: the accuracy of query decomposition and the efficiency of distributed join execution.

In this demonstration paper we present an architecture for providing integrated access to data sources over the Web of Linked Data. The presented architecture has been implemented in our prototype system - LDMS. The system has the ability of monitoring, managing distributed RDF data sources and allows to retrieve data using SPARQL queries.

In the following we will describe the LDMS system and give a demonstration of its practical applicability in the Semantic application. In section 2 we present our main purpose and give some insights into the framework of LDMS. Next, in section 3 we present the demonstration scenario. Finally, we conclude with some remarks on future work.

Y. Demazeau et al. (Eds.): PAAMS 2014, LNAI 8473, pp. 347–350, 2014.

2 Main Purpose

LDMS[1] is being developed to provide an efficient solution for virtually integrating Linked Data. Figure 1 shows the framework of an application built on top of LDMS. The application layer uses data retrieved by LDMS to provide users with a complete view. An semantic application which provides information about movies is employed for our demonstration (presented in section 3).

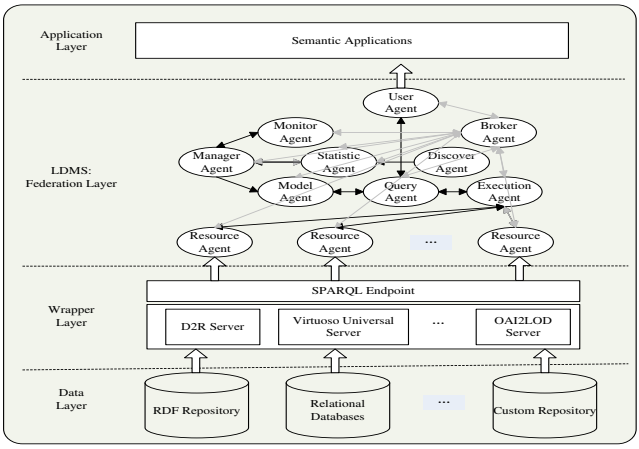

Fig. 1. LDMS System Overview

The federation layer is comprised of a network of cooperating agents communicating by means of the agent communication language ACL. Agents register their network addresses and services in the broker agent. When an agent has accomplished its work or needs help from other agents, it will send a request to the broker agent which routes this request to an appropriate agent or send back an address of the needed agent.

The query federator in LDMS is implemented in java based on Jena API. Clients submit SPARQL queries via user agents. The queries are routed by brokerage agents to specialized agents for data retrieval from remote data sources and integration of intermediate results. This federated query processing is transparent for the user, i.e., it gives the user the impression to query one single RDF graph despite the real data being distributed on the web.

The main technologies used in LDMS are:

1. Query decomposition: Query decomposition affects the performance of query systems in two ways: time performance and result completeness. LDMS uses a SPARQL GRAPH keyword based approach to make the query decomposition to be convenient and accurate.

[1] LDMS is only available as Java source code(eclipse project) from the SVN repository: `https://svn.code.sf.net/p/semwldms/code/LDMS/trunk`

2. Join order: The main goal of join order is to reduce the cost of network transmission. In LDMS, all sub-queries in a query plan are ordered by the number of intermediate results.
3. Group join: To reduce the number of requests and avoid program errors, joins are computed in a group-join which makes a compromise between pipeline join and semi-join.
4. Agent-based architecture: To achieve a flexibility and openness, LDMS adopts the technologies of agent and provides flexible, extensible means to manage data sources.

For evaluations With FedBench[2], LDMS shows a significant improvement of query performance compared to state-of-the-art federated SPARQL query systems, namely SPLENDID and FedX.

3 Demonstration

To illustrate the practicability of LDMS, define a demonstration scenario using the previously presented framework. The sample application is developed to provide information about movies. It integrates data from multiple Linked Data sources for a complete view of movies. The scenario steps from the user's point of view are summarized in the following and illustrated in figure 2.

1. Data source discovery. Discovery Agent in LDMS discovers new data sources by periodically accessing a global data registry institution.
2. Data source management. Management Agent in LDMS provides an interface for system managers to add, update or delete data sources.
3. Query submission. User Agent in LDMS verifies user queries, transfers them to query agent and returns query answers to clients.
4. Query evaluation and result presentation. In LDMS, Query Agent decomposes original queries into some sub-queries and makes query plans. Execution Agent applies its optimizations and interacts with Resource Agents for executing query plans. Then, Query Agent integrates the intermediate results returned by Execution Agent. Finally, User Agent transforms the final result answers to the application layer for presentation.

For the demonstration we simulate a real-world environment using the FedBench benchmark. It includes two subsets of data sources in the Linked Data cloud: Cross Domain and Life Science. FedBench focus on testing and analyzing the performance of federated query processing strategies on semantic data such as those of LDMS. Since LDMS provides a decoupled architecture and improves the query performance compared to existing solutions, it is valuable effort for providing an infrastructure of the development of semantic applications.

[2] FedBench project page:`http://code.google.com/p/fbench/`

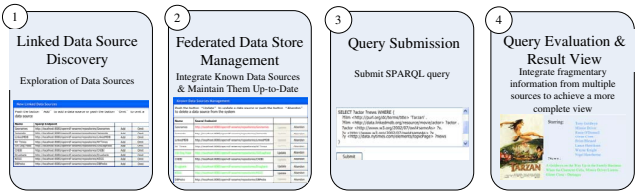

Fig. 2. Illustration of the Demonstration Workflow

4 Conclusions

In this paper we have presented an infrastructure for developing semantic applications. LDMS uses the SPARQL GRAPH based approach to decompose original queries and the group-join approach to improve distributed join operations. Besides, it implements a decoupled architecture which is the key to system scalability and extensibility. The evaluation based on FedBench benchmark indicates a significant improvement about query response time. In the future version, we propose to combine more advanced optimization techniques into LDMS for further improving query performance.

Acknowledgment. This work was supported by the National Natural Science Foundation of China (#61272361, #61370137).

References

1. Bizer, C., Heath, T., Berners-Lee, T.: Linked data-the story so far. International Journal on Semantic Web and Information Systems (IJSWIS) 5(3), 1–22 (2009)
2. Quilitz, B., Leser, U.: Querying distributed RDF data sources with SPARQL. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 524–538. Springer, Heidelberg (2008)
3. Hartig, O., Bizer, C., Freytag, J.-C.: Executing SPARQL queries over the web of linked data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds.) ISWC 2009. LNCS, vol. 5823, pp. 293–309. Springer, Heidelberg (2009)
4. Ladwig, G., Tran, T.: Linked data query processing strategies. In: Patel-Schneider, P.F., Pan, Y., Hitzler, P., Mika, P., Zhang, L., Pan, J.Z., Horrocks, I., Glimm, B. (eds.) ISWC 2010, Part I. LNCS, vol. 6496, pp. 453–469. Springer, Heidelberg (2010)
5. Langegger, A., Wöß, W., Blöchl, M.: A semantic web middleware for virtual data integration on the web. In: Bechhofer, S., Hauswirth, M., Hoffmann, J., Koubarakis, M. (eds.) ESWC 2008. LNCS, vol. 5021, pp. 493–507. Springer, Heidelberg (2008)
6. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: Optimization techniques for federated query processing on linked data. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 601–616. Springer, Heidelberg (2011)
7. Görlitz, O., Staab, S.: Splendid: Sparql endpoint federation exploiting void descriptions. In: COLD (2011)
8. Stuckenschmidt, H., Vdovjak, R., Houben, G.J., Broekstra, J.: Index structures and algorithms for querying distributed rdf repositories. In: Proceedings of the 13th international conference on World Wide Web, pp. 631–639. ACM (2004)