FedX: A Federation Layer for Distributed Query Processing on Linked Open Data

Andreas Schwarte[1], Peter Haase[1], Katja Hose[2],
Ralf Schenkel[2], and Michael Schmidt[1]

[1]fluid Operations AG, Walldorf, Germany
[2]Max-Planck Institute for Informatics, Saarbrücken, Germany

Abstract. Driven by the success of the Linked Open Data initiative today's Semantic Web is best characterized as a Web of interlinked datasets. Hand in hand with this structure new challenges to query processing are arising. Especially queries for which more than one data source can contribute results require advanced optimization and evaluation approaches, the major challenge lying in the nature of distribution: Heterogenous data sources have to be integrated into a federation to globally appear as a single repository. On the query level, though, techniques have to be developed to meet the requirements of efficient query computation in the distributed setting. We present FedX, a project which extends the Sesame Framework with a federation layer that enables efficient query processing on distributed Linked Open Data sources. We discuss key insights to its architecture and summarize our optimization techniques for the federated setting. The practicability of our system will be demonstrated in various scenarios using the Information Workbench.

1 Introduction

Motivated by the ongoing success of the Linked Open Data initiative and the growing amount of semantic data sources available on the Web, new approaches to query processing are emerging. While query processing in the context of RDF is traditionally done locally using centralized stores, recently one can observe a paradigm shift towards federated approaches which can be attributed to the decentralized structure of the Semantic Web. The Linked Open Data cloud - representing a large portion of the Semantic Web - comprises more than 200 datasets that are interlinked by RDF links. In practice many scenarios exist where more than one data source can contribute information, making query processing more complex. Contrary to the idea of Linked Data, centralized query processing requires to copy and integrate relevant datasets into a local repository. Accounting for the structure, the natural approach to follow in such a setting is federated query processing over the distributed data sources.

While there exist efficient solutions to query processing in the context of RDF for local, centralized repositories [7, 5], research contributions and frameworks for distributed, federated query processing are still in the early stages. In practical terms the Sesame framework in conjunction with AliBaba[1] is one possible sample solution allowing for federations of distributed repositories and endpoints. However, benchmarks have shown poor performance for many queries in the federated

[1] http://www.openrdf.org/doc/alibaba/2.0-beta4/

setup due to the absence of advanced optimization techniques [6]. From the research community DARQ [9] and Networked Graphs [10] contribute approaches to federated SPARQL queries and federated integration. Since both require proprietary extensions to languages and protocols which are not supported by most of today's endpoints, they are not applicable in practical environments.

In this demonstration paper we present FedX, a practical framework for transparent access to data sources through a federation. The framework offers efficient query processing in the distributed setting, while using only protocols and standards that are supported by most of today's data sources.

In the following we will describe the FedX system and give a demonstration of its practical applicability in the Information Workbench. In section 2 we give some insights into the federation layer. Next, in section 3 we present the demonstration scenario. Finally, we conclude with some remarks on future work.

2 FedX - Design and System Overview

FedX[2] is being developed to provide an efficient solution for distributed query processing on Linked Open Data. It is implemented in Java and extends the Sesame framework with a federation layer. FedX is incorporated into Sesame as a `SAIL` (`S`torage `a`nd `I`nference `L`ayer), which is Sesame's mechanism for allowing seamless integration of standard and customized RDF repositories. The underlying Sesame infrastructure enables heterogeneous data sources to be used as endpoints within the federation. FedX implements the logics for optimization and efficient execution of the query in the distributed setting.

Figure 1 shows the architecture of an application built on top of FedX. The application layer provides the frontend to the query processing engine and is necessary for any kind of interaction with the federation. We decided to employ the Information Workbench [8] as such for our demonstration (see section 3). However, any other application can be used as well by utilizing the Sesame API.

The second layer is composed of the Sesame Framework and provides the basic infrastructure for the query processing engine. In particular this includes facilities for query parsing, Java mappings, I/O components, and the API for client interaction.

The federation layer is implemented as an extension to Sesame in form of a `SAIL` and constitutes FedX. FedX utilizes the basic Sesame infrastructure described above, and adds the necessary functionality for data source management, endpoint communication and - most importantly - optimizations for distributed query processing. Data sources can be added to a FedX federation in form of any repository mediator, where the latter means a supported Sesame repository implementation. Standard implementations are provided for local, native Sesame repositories as well as for remote SPARQL endpoints. Furthermore custom mediators can be integrated by implementing the appropriate Sesame interface. With these mediators different types of federations are possible: purely local ones consisting of native, local Sesame repositories, endpoint federations or hybrid forms.

[2] FedX project page: http://iwb.fluidops.com/FedX

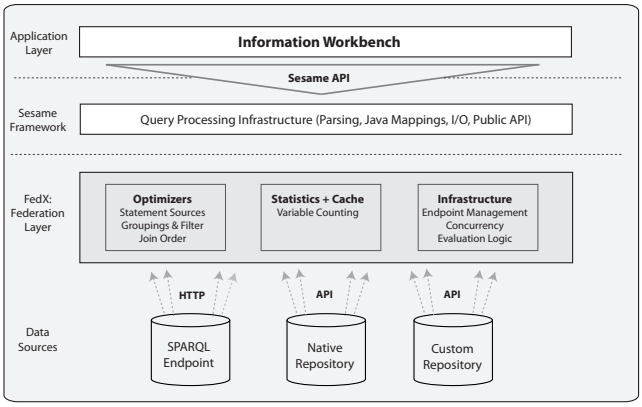

Fig. 1: FedX System Overview

Federated query processing in FedX is comprised of the following steps. First, a global query is formulated against a federation of data sources. The global query is then parsed and optimized for the distributed setting. In particular it is split into local subqueries that can be answered by the individual data sources. Results of these local queries are merged in the federator and finally returned in an aggregated form. The whole process is transparent for the user, i.e. data appears to be *virtually integrated* in a single RDF graph.

Most crucial to the performance of such a query processing engine is the use of optimization techniques. Especially in the federated, distributed setting it is essential to apply new approaches to reduce the number of requests to the endpoints. Besides various generic techniques, FedX integrates some more sophisticated optimizations for the distributed environment. The combination of our applied join order optimization and the grouped subqueries reduce the number of intermediate results and requests tremendously, and are thus the major contributions for improving query performance in the distributed setting. The following listing gives an overview.

- **Statement sources:** Prior to query evaluation, all statements of the given SPARQL query are examined for their relevant data sources to avoid unnecessary communication during query processing.
- **Filter pushing:** SPARQL filter expressions are pushed down whenever possible to allow for early evaluation.
- **Parallel processing:** Concurrency is exploited by means of multithreaded execution of join and union computations.
- **Join order:** Join order tremendously influences performance since the number of intermediate results determines overall query runtime. In FedX the variable counting technique proposed in [3] supplemented with various heuristics is used to estimate the cost for each join. Following a greedy approach the joins are then executed in ascending order of cost.
- **Bound joins:** To reduce the number of requests and thus the overall runtime, joins are computed in a block nested loop join.
- **Groupings:** Statements which have the same relevant data source are coexecuted in a single SPARQL query to push joins to the particular endpoint.

First benchmarks with FedBench[3] indicate a significant improvement of query performance compared to existing solutions[4]. For many queries proposed in [6] a performance gain of more than 90% can be achieved resulting in improvements of an order of magnitude, timeouts do not occur any longer. This is in particular due to the improved join order and the other above mentioned optimizations.

3 Demonstrating FedX in the Information Workbench

With the goal of illustrating the practicability of our system we provide a demonstration scenario using the previously discussed architecture. We employ the Information Workbench for demonstrating the federated approach to query processing with FedX. The Information Workbench is a flexible platform for Linked Data application development and provides among others frontend facilities for our UI as well as the integration with the backend, i.e. the query processing layers. In our demonstration we show a browser based UI allowing dynamic access and manipulation of federations at query time as well as ad hoc query formulation, then we execute the optimized query at the configured data sources using FedX, and finally we present the query results in the platform's widget based visualization components. The scenario steps from the user's point of view are summarized in the following and illustrated in figure 2.

1. **Linked Open Data discovery.** Data sources can be visually explored and discovered using a global data registry.
2. **Federation setup.** The federation is constructed and/or modified dynamically on demand using a browser based self-service interface. Discovered Linked Data repositories can be integrated into the federation with a single click.
3. **Query definition.** A query can be formulated ad hoc using SPARQL or selected from a subset of the FedBench queries. The predefined queries are designed to match the domain-specific data sources and produce results.
4. **Query execution using FedX and result presentation.** The formulated query is submitted to the backend using the Sesame API and processed within FedX. After the query is parsed by Sesame, FedX applies its optimizations and evaluates the query at the data sources given by the dynamically configured federation. Finally, results are returned to the application layer for presentation in the widget based visualization components provided by the Information Workbench.

For the demonstration we use cross domain and lifescience datasets and queries as proposed in the FedBench benchmark. Those collections span a subset of the Linked Open Data cloud and are useful to illustrate practical applicability of query processing techniques such as those of FedX. Since FedX improves

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

[4] For an initial comparison we employed the AliBaba extension for the Sesame framework. To the best of our knowledge AliBaba provides the only federation layer available that does not require any proprietary extensions (e.g. SPARQL extensions).

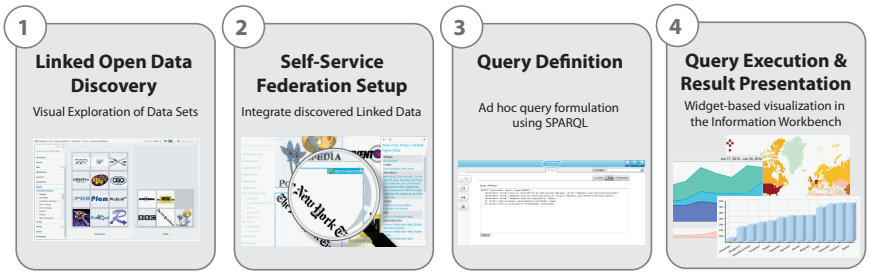

Fig. 2: Illustration of the Demonstration Workflow

query response time compared to existing solutions, and moreover since the total runtime for most queries is in a range that is considered responsive, it is a valuable contribution for practical federated query processing.

4 Conclusion and Future Work

In this paper we have presented FedX and a practical demonstration within the Information Workbench. FedX provides a flexible federation layer integrated into the Sesame Framework which is suitable for practical application scenarios. First evaluation benchmarks have indicated that response time and query performance are such, that FedX in conjunction with a suitable application layer can be considered a highly valuable framework for federated query processing in today's settings. In future versions more advanced optimization techniques will be integrated into FedX to further improve query performance. Current ideas include the use of caching, integration of statistics (e.g. voiD [2]), and finally advanced grouping of subqueries to reduce the number requests.

References

1. Jeen Broekstra et al. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In *The Semantic Web - ISWC 2002*. Springer, 2002.
2. Keith Alexander et al. Describing Linked Datasets – On the Design and Usage of voiD. In *Proceedings of the Linked Data on the Web Workshop*, 2009.
3. Markus Stocker et al. SPARQL basic graph pattern optimization using selectivity estimation. In *WWW*, pages 595–604. ACM, 2008.
4. Olaf Görlitz et al. Federated Data Management and Query Optimization for Linked Open Data. In *New Directions in Web Data Management*. 2011.
5. Orri Erling et al. Rdf support in the virtuoso dbms. In *CSSW*, 2007.
6. Peter Haase, Tobias Mathäß, and Michael Ziller. An Evaluation of Approaches to Federated Query Processing over Linked Data. In *I-SEMANTICS*, 2010.
7. Thomas Neumann and Gerhard Weikum. Rdf-3X: a RISC-style engine for RDF. *PVLDB*, 1(1), 2008.
8. Peter Haase et al. The Information Workbench - Interacting with the Web of Data. Technical report, fluid Operations & AIFB Karlsruhe, 2009.
9. Bastian Quilitz and Ulf Leser. Querying distributed RDF data sources with SPARQL. In *ESWC*, 2008.
10. Simon Schenk and Steffen Staab. Networked graphs: a declarative mechanism for sparql rules, sparql views and rdf data integration on the web. In *WWW*, 2008.