FedBench: A Benchmark Suite for Federated Semantic Data Query Processing

Michael Schmidt[1], Olaf Görlitz[2], Peter Haase[1], Günter Ladwig[3], Andreas Schwarte[1], and Thanh Tran[3]

[1] Fluid Operations AG, Walldorf, Germany
[2] Institute for Web Science and Technology, University of Koblenz-Landau, Germany
[3] Institute AIFB, Karlsruhe Institute of Technology, Germany

Abstract. In this paper we present *FedBench*, a comprehensive benchmark suite for testing and analyzing the performance of federated query processing strategies on semantic data. The major challenge lies in the *heterogeneity of semantic data use cases*, where applications may face different settings at both the data and query level, such as varying data access interfaces, incomplete knowledge about data sources, availability of different statistics, and varying degrees of query expressiveness. Accounting for this heterogeneity, we present a highly *flexible benchmark suite*, which can be customized to accommodate a variety of use cases and compare competing approaches. We discuss design decisions, highlight the flexibility in customization, and elaborate on the choice of data and query sets. The practicability of our benchmark is demonstrated by a rigorous evaluation of various application scenarios, where we indicate both the benefits as well as limitations of the state-of-the-art federated query processing strategies for semantic data.

1 Introduction

Driven by the success of the Linking Open Data initiative, the amount of semantic data that is published on the Web in the form of RDF is increasing at a tremendous pace. While offering great potentials for innovative applications that integrate heterogeneous data from different sources, on the data management side, this development comes along with a variety of new challenges, last but not least due to the sheer amount of data that may be utilized by such applications. Most research contributions in the context of RDF data processing have focused on the problem of query evaluation over local, centralized repositories (see e.g. [2, 24, 19]) – and for these scenarios different benchmarks have been proposed [11, 6, 21]. Accounting for the decentralized nature of the Semantic Web, though, one can observe an ongoing shift from localized to federated semantic data processing, where independent endpoints provide data, and semantic data applications utilize both local repositories and remote data sources at the same time to satisfy their information needs. In response to this paradigm shift, different federated RDF processing strategies – targeted at different use cases and application scenarios – have been proposed [15, 14, 16, 9, 17].

L. Aroyo et al. (Eds.): ISWC 2011, Part I, LNCS 7031, pp. 585–600, 2011.

With distributed semantic data processing becoming increasingly important, we identify a clear need for *a benchmark tailored to the problem of federated semantic data query processing.* We employ a broad definition of *semantic data*, which includes Linked Data sources, datasets, and ontologies represented in RDF. The main challenge here lies in the diversity and heterogeneity of semantic data use cases, and the demands they pose to a benchmark: First, we can observe *heterogeneity at data level* along several dimensions: applications are facing different physical distribution of datasets, different interfaces for data access, incomplete knowledge about the existence of entry points into the Web of data, and different types of metadata and statistics. Apart from the challenges at data level, applications may also exhibit *different demands w.r.t. query evaluation*, including aspects such as query languages, expressiveness, and ranking.

The overall setting in which a concrete semantic data application is settled may have severe impact on query processing strategies. Ultimately, there cannot exist a single "one-size-fits-all" benchmark to measure each and every aspect of an application – or to compare the performance of orthogonal federated query processing strategies. Hence, taking an existing benchmark and distributing its data across several endpoints may cover some, but not all challenges that arise in the context of federated semantic data processing. What is needed instead is a *collection* of *customizable* benchmark scenarios that *accommodate a multitude of dimensions* as well as *essential challenges* – and from which one can choose data sets, queries, and settings that fit the specific needs.

Contributions. (1) Based on a review of federated query processing scenarios our community has dealt with so far, we discuss orthogonal *dimensions at data and query level* that can be used to classify existing approaches. (2) Accounting for the heterogeneity of these dimensions, we present a rich collection of *queries*, *data*, and *data statistics*, which can be flexibly combined and customized. This makes our benchmark generic enough to cover a broad range of use cases, such as testing the performance of the underlying federation approach, data access mechanisms, static optimization based on metadata and statistics, queries with varying characteristics, and many more. All queries and datasets were carefully chosen to reflect a variety of domains, query patterns, and typical challenges in query processing (in particular in distributed settings). While some of the queries were specifically designed to test and vary in these aspects, others were taken from prototypical, domain-specific use cases (e.g. in the Life Science domain) built by participants in other projects. (3) In order to *show the flexibility* and illustrate *the broad range of scenarios covered by FedBench*, we provide results for selected scenarios and implementations, identifying areas where ongoing work is required. Our results are published in a Wiki, where we also maintain data, statistics, queries, and scenarios. We invite researchers and benchmark users to customize and extend the benchmark suite according to their own needs.

We point out that the resulting benchmark suite is available online[1], including a flexible and extensible Open Source Java-based evaluation framework.

[1] See `http://fbench.googlecode.com/`

Related Work. Apart from benchmarks that target structural properties of RDF schemas (such as [18]), several benchmarks for RDF, RDFS, and OWL data processing have been proposed. The Lehigh University Benchmark [11], for instance, has been designed to test the reasoning capabilities of systems over a single ontology. The SPARQL-specific, use-case driven Berlin SPARQL Benchmark [6] comes with a set of queries implementing meaningful requests on top of an eCommerce scenario modeled in RDF. Complementary, the SPARQL Performance Benchmark (SP^{2}Bench) [21] puts a stronger focus on language-specific features of the SPARQL query language, addressing optimization in complex scenarios. None of the above benchmarks considers federation at data level, nor does provide data collections consisting of multiple interlinked datasets.

To the best of our knowledge, the only work addressing the latter issue was our previous work in [13], which – focusing on selected federation approaches – served as a starting point for designing FedBench. Going far beyond this initial work, in this paper we present a holistic benchmark suite, including a variety of new data and query sets, new scenarios such as Linked Data access (i.e., via HTTP requests), an automated evaluation framework (with support for various metrics like counting the number of requests, automated evaluation, interfaces for connecting new systems etc.), a discussion and classification of state-of-the-art approaches, a discussion of statistics, as well as novel experiments and findings.

Our benchmark has been designed to be compatible with the current state-of-the-art in Linked Data query processing. In particular, in our benchmark we use queries defined in SPARQL 1.0, without requiring specific extensions for explicit specification of the endpoint services as proposed by the SPARQL 1.1 federation extensions. To our knowledge, there is currently no public implementation of the SPARQL 1.1 federation extension [1]. Examples of systems expected to support SPARQL 1.1 federation in future releases include SPARQL DQP [5], Sesame [7], and FedX [23]. While in this paper we focus on SPARQL 1.0 features, on our project page we also provide SPARQL 1.1 versions of the benchmark queries, ready to be used as soon as implementations become available.

An analysis and comparison of the structure of different semantic data sets and benchmarks from the Linked Data domain is presented in [8]. It shows that artificial datasets used in benchmarks are typically highly structured, while Linked Data are less structured. They conclude that benchmarks should not solely rely on artificial data but also consider real world datasets. Resuming this discussion, in Section 3.1 we will present an analysis and comparison of our data sets using the methods proposed in [8].

Outline. In the next section, we identify and discuss essential dimensions of federated semantic data processing, which form the groundwork for the benchmark suite presented in this paper. Next, in Section 3 we motivate benchmark design goals and present the suite in detail, namely benchmark datasets (Section 3.1), covering aspects such as their properties and associated statistics, benchmark queries and their properties (Section 3.2), as well as the benchmark driver (Section 3.3). We turn towards a comprehensive evaluation of concrete application scenarios in Section 4 and wrap up with some concluding remarks in Section 5.

2 Heterogeneity of Semantic Data Use Cases

To date, several approaches to semantic data query processing have been studied, comprising both centralized and decentralized settings. Centralized approaches, where all data is periodically crawled and stored locally, come with the merits of high controllability and reliability. While Google and the likes have shown that this is a successful approach for dealing with Web documents that primarily comprise text, it has also been adopted for semantic data, where projects like Factforge[2] collect large amounts of RDF(S) data in centralized stores.

If federation is set up dynamically or the underlying sources exhibit high update rates, though, it may not be affordable to keep imported semantic data up-to-date. Then, decentralized, federated query processing strategies are required, typically implemented on top of public SPARQL endpoints or directly on Linked Data accessible via HTTP requests. In the best case, not only the existence of data sources but also detailed statistics can be assumed and exploited for optimization [14]. A stronger relaxation is to assume only partial knowledge about the sources, e.g. past information that is stored and employed for future runs, thereby obtaining more entry points and hints to explore unknown sources in a goal-directed fashion. In fact, it has been shown that already little knowledge obtained at runtime can be used to adaptively correct the query processing strategy chosen at compile time, to improve performance [16]. In the worst case, engines have to deal with federated scenarios where no data and knowledge are available, so queries have to be processed based on iterative URI lookups [15].

In summary, previous approaches reveal several dimensions along the data and query level, which we use to characterize approaches. They determine the challenges, including the major issues of centralized vs. decentralized processing and knowledge about datasets discussed above. At data level, we identify heterogeneity along the following dimensions:

(D1) Physical Distribution: Federated query processing systems may either access and process global data from the Web, process locally stored data sources, or mix up both paradigms connecting *local* with *global data*.

(D2) Data Access Interface: Semantic data may be accessible through different interfaces. There may be native repositories, SPARQL endpoints, and Linked Data accessible through HTTP requests. These interfaces provide different access paths to the data, ranging from iterators at data level, URI lookups, to expressive queries in different languages.

(D3) Data Source Existence: In particular in Linked Data scenarios, not all sources may be known a priori. Hence, applications may have only few entry points into the data graph, which can be used to iteratively deep-dive by exploring links (see for instance the setting described in [16]).

(D4) Data Statistics: In the best case, advanced statistical information about properties, counts, and distributions in the form of histograms for all data sets are available; in the worst case – in particular if data is not stored locally – only few or no information about the data sources may be given.

[2] `http://ontotext.com/factforge/`

The concrete setting an application faces at data level – i.e., the classification within dimension (D1)–(D4) – implies challenges in data processing and imposes an upper bound on the efficiency in query processing: applications built on top of local repositories exploiting detailed statistical knowledge for query optimization, for instance, are generally faster than applications that rely on federated Linked Data accessible via HTTP lookups, where network delay and incomplete knowledge about data sets impose hard limits on query efficiency.

Apart from the challenges at data level, applications may also face different challenges at query level. Like the dimensions at data level, also those at query level drive the challenges behind semantic data applications and should be covered in a benchmark. In particular, we identify the following dimensions.

(Q1) Query Language: The expressiveness of the query language needed by applications may vary from case to case: while some applications get around with simple conjunctive queries, others may rely on the full expressive power of RDF query languages, such as the de facto standard SPARQL [20, 22].

(Q2) Result Completeness: Certain applications may rely on complete results, while others cannot afford it when responsiveness is first priority. In particular in Linked Data scenarios where complete knowledge cannot be assumed (s.t., beginning from some entry points, further sources have to be discovered via online link traversal) not all data sources may be found [15, 16].

(Q3): Ranking: Applications may be interested in queries that enable ranking according to some predefined metrics, or maybe only in top-k results.

3 FedBench: Benchmark Description

In order to support benchmarking of the different scenarios that emerge along all the dimensions, FedBench consists of three components, all of which can be *customized and extended* to fit the desired scenario: (i) multiple datasets, (ii) multiple query sets, and (iii) a comprehensive evaluation framework. We first elaborate on the datasets and statistics (addressing dimension (D4)), then present the queries (addressing dimensions (Q1)–(Q3)), and conclude with a discussion of our evaluation framework, which addresses dimensions (D1)–(D3).

3.1 Benchmark Data

Accounting for the heterogeneity of semantic data use cases, we provide three *data collections*, each consisting of a number of interlinked datasets. The data collections have been selected to represent both real-world and artificial data federations over multiple representative semantic datasets. The collections differ in size, coverage, and types of interlinkage. Two of them are subsets of the Linked Open Data cloud: The first spans different domains of general interest, representing typical scenarios of combining cross-domain data with heterogeneous types of interlinkage; the second contains datasets from the Life Science area, representing a federation scenario in a very domain-specific setting. Additionally, we use a partitioned synthetic data collection, whose advantage lies in the ability to simulate federations of varying size with well-defined characteristics of the data.

General Linked Open Data Collection. This first data collection consists of datasets from different domains: **DBpedia** is a central hub in the Linked Data world, containing structured data extracted from Wikipedia. Many datasets are linked to DBpedia instances. **GeoNames** provides information about geographic entities like countries and cities. **Jamendo** is a music database containing information about artists, records, tracks, publishers and publication dates. **LinkedMDB** exhibits details about films, genres, actors, producers, etc. and connects its instances to the corresponding DBpedia entities. The **New York Times** dataset contains about 10,000 subject headings covering different topics, which are linked with with people, organizations and locations. Finally, the **Semantic Web Dog Food** dataset provides information about Semantic Web conferences of the past years, including paper descriptions, authors, and so on.

Life Science Data Collection. In this collection, we again included the DBpedia subset from the General Linked Open Data dataset as a central hub. **KEGG** (Kyoto Encyclopedia of Genes and Genomes) contains data about chemical compounds and reactions, with a focus on information relevant for geneticists. It is published in a number of separate modules; in the dataset we included the modules KEGG Drug, Enzyme, Reaction and Compound. Next, **ChEBI** (Chemical Entities of Biological Interest) is a dictionary of molecular entities focused on "small" chemical compounds, describing constitutionally or isotopically distinct atoms, molecules, ions, ion pairs, radicals, radical ions, complexes, conformers, etc. **DrugBank** is a bioinformatics and cheminformatics resource that combines detailed drug (i.e. chemical, pharmacological, and pharmaceutical) data with comprehensive drug target (i.e. sequence, structure, and pathway) information. The datasets are linked in different ways: Drugbank is linked with DBpedia via `owl:sameAs` statements, and other datasets are linked via special properties, e.g. Drugbank links to KEGG via the property `keggCompoundId`. KEGG and Drugbank can be joined via identifiers of the CAS database (Chemical Abstract Service). Some links are implicit by the use of common identifiers in literal values, e.g. the `genericName` in Drugbank corresponds to the `title` in ChEBI.

SP^{2}Bench is a synthetic dataset generated by the SP^{2}Bench data generator [21], which mirrors vital characteristics (such as power law distributions or Gaussian curves) encountered in the DBLP bibliographic database. The data generator provides a single dataset, from which we created a collection by clustering by the types occurring in the dataset, finally obtaining 16 sub-datasets (for persons, inproceedings, articles, etc.) that can be deployed independently in a distributed scenario. The data collection consists of 10M triples in total.

Metadata and Statistics about data sources are important for identifying suitable data sources for answering a given query, as well as for query optimization. They can be used to parametrize the benchmark along dimension (D4).

Table 1. Basic Statistics of Datasets[1]

Collection	Dataset	version	#triples	#subj.	#pred.	#obj.	#types	#links	strct.
Cross Domain	DBpedia subset[2]	3.5.1	43.6M	9.50M	1063	13.6M	248	61.5k	0.19
	NY Times	2010-01-13	335k	21.7k	36	192k	2	31.7k	0.73
	LinkedMDB	2010-01-19	6.15M	694k	222	2.05M	53	63.1k	0.73
	Jamendo	2010-11-25	1.05M	336k	26	441k	11	1.7k	0.96
	GeoNames	2010-10-06	108M	7.48M	26	35.8M	1	118k	0.52
	SW Dog Food	2010-11-25	104k	12.0k	118	37.5k	103	1.6k	0.43
Life Sciences	DBpedia subset[2]	3.5.1	43.6M	9.50M	1063	13.6M	248	61.5k	0.19
	KEGG	2010-11-25	1.09M	34.3k	21	939k	4	30k	0.92
	Drugbank	2010-11-25	767k	19.7k	119	276k	8	9.5k	0.72
	ChEBI	2010-11-25	7.33M	50.5k	28	772k	1	-	0.34
SP^{2}Bench	SP^{2}Bench 10M	v1.01	10M	1.7M	77	5.4M	12	-	0.76

[1] All datasets are available at `http://code.google.com/p/fbench/`.
[2] Includes the ontology, infobox types plus mapped properties, titles, article categories with labels, Geo coordinates, images, SKOS categories, and links to New York Times and Linked Geo Data.

Some Linked Data sources provide basic VoiD [3] statistics such as *number of triples*, *distinct subjects, predicates, objects*, and information about the vocabulary and links to other sources. Table 1 surveys such basic statistics for our datasets. These and other statistics (such as *predicate and type frequency*, *histograms*, *full pattern indexes* obtained by counting all combinations of values in triple patterns, *full join indexes* obtained by counting all join combinations, and *link statistics*) can be exploited by engines in the optimization process.

Duan et al. [8] introduced the notion of structuredness, which indicates whether the instances in a dataset have only a few or all attributes of their types set. They show that artificial datasets are typically highly structured and “real” datasets are less structured. As shown in the last column of Table 1, the structuredness (range $[0, 1]$) varies for our datasets, e.g. DBpedia has a low structuredness value whereas Jamendo and KEGG are highly structured.

3.2 Benchmark Queries

There are two reasonable options for the design of benchmark queries [10]: language-specific vs. use case driven design. The query sets we propose cover both dimensions. We choose SPARQL as a query language: It is known to be relationally complete, allowing us to encode a broad range of queries with varying complexity, from simple conjunctive queries to complex requests involving e.g. negation [4, 20, 22]. We restrict ourselves on general characteristics, pointing to the FedBench project page for a complete listing and description.

Life Science (LS) and Cross Domain Queries (CD). These two query sets implement realistic, real-life use cases on top of the cross-domain and life science data collection, respectively. Their focus is on federation-specific aspects, in particular (1) number of data sources involved, (2) join complexity, (3) types

Example, Life Science Query 4: For all drugs in DBpedia, find all drugs they interact with, along with an explanation of the interaction.

```
SELECT ?Drug ?IntDrug ?IntEffect WHERE {
   ?Drug rdf:type dbpedia-owl:Drug .
   ?y owl:sameAs ?Drug .
   ?Int drugbank:interactionDrug1 ?y .
   ?Int drugbank:interactionDrug2 ?IntDrug .
   ?Int drugbank:text ?IntEffect . }
```

This query includes a star-shaped sub pattern of drugs which is connected via `owl:sameAs` link to `DBpedia` drug entities.

Example, Cross Domain Query 5: Find the director and the genre of movies directed by Italians.

```
SELECT ?film ?director ?genre WHERE {
   ?film dbpedia-owl:director ?director.
   ?director dbpedia-owl:nationality dbpedia:Italy .
   ?x owl:sameAs ?film .
   ?x linkedMDB:genre ?genre . }
```

A chain-like query for finding film entities (in `LinkedMDB` and in `DBpedia`) linked via `owl:sameAs` and restricted on genre and director.

Example, Linked Data Query 4: Find authors of papers at the ESWC 2010 conference who were also involved in the conference organization.

```
SELECT * WHERE {
  ?role swc:isRoleAt <http://data.semanticweb.org/conference/eswc/2010> .
  ?role swc:heldBy ?p .
  ?paper swrc:author> ?p .
  ?paper swc:isPartOf ?proceedings .
  ?proceedings swc:relatedToEvent  <http://data.semanticweb.org/conference/eswc/2010> }
```

Fig. 1. Selected Benchmark Queries

of links used to join sources, and (4) varying query (and intermediate) result size. Figure 1 exemplarily discusses three queries taken from these query sets.

SP^{2}Bench Queries (SP). Next, we reuse the queries from the SP^{2}Bench SPARQL performance benchmark, which were designed to test a variety of SPARQL constructs and operator constellations, but also cover characteristics like data access patterns, result size, and different join selectivities. Some of the SP^{2}Bench queries have high complexity, implementing advanced language constructs such as negation and double negation. They are intended to be run on top of the distributed SP^{2}Bench dataset described in Section 3.1. A thorough discussion of the queries and their properties can be found in [21].

Table 2. Query Characteristics. Operators: **A**nd ("."), **U**nion, **F**ilter, **O**ptional; Modifiers: **D**istinct, **L**imit, **Of**fset, **Or**derBy; Structure: **S**tar, **C**hain, **H**ybrid

Life Science (LS)					
	Op.	*Mod.*	*Struct.*	*#Res.*	*#Src*
1	U	-	-	1159	2
2	AU	-	-	333	4
3	A	-	H	9054	2
4	A	-	H	3	2
5	A	-	H	393	3
6	A	-	H	28	3
7	AFO	-	H	144	3
Cross Domain (CD)					
1	AU	-	S	90	2
2	A	-	S	1	2
3	A	-	H	2	5
4	A	-	C	1	5
5	A	-	C	2	5
6	A	-	C	11	4
7	A	-	C	1	5

SP^{2}Bench (SP)					
	Op.	*Mod.*	*Struct.*	*#Res.*	*#Src*
1	A	-	S	1	11
2	AO	Or	S	>500k	12
3a	AF	-	S	>300k	16
3b	AF	-	S	2209	16
3c	AF	-	S	0	16
4	AF	D	C	>40M	14
5a	AF	D	C	>300k	14
5b	AF	D	C	>300k	14
6	AFO	-	H	>700k	16
7	AFO	D	H	>2k	14
8	AFU	D	H	493	16
9	AU	D	-	4	16
10	-	-	-	656	12
11	-	LOfOr	-	10	8

Linked Data (LD)				
	Op.	*Mod.*	*Struct.*	*#Res.*
1	A	-	C	309
2	A	-	C	185
3	A	-	C	162
4	A	-	C	50
5	A	-	S	10
6	A	-	H	11
7	A	-	S	1024
8	A	-	H	22
9	A	-	C	1
10	A	-	C	3
11	A	-	S	239

Linked Data Queries (LD). Today's Linked Data engines typically focus on basic graph patterns (Conjunctive Queries). Therefore, all LD queries are basic graph pattern queries, designed to deliver results when processing Linked Data in an exploration-based way (cf. the bottom-up strategy described in Section 4.1).

Queries LD1–LD4 use the SW Dog Food dataset to extract information about conference and associated people. LD1–LD3 all contain a single URI to be used as a starting-point for exploration-based query processing, whereas LD4 has two URIs that could be used to speed up processing by starting the exploration from multiple points in the Linked Data graph (cf. Figure 1). The other queries operate on DBpedia, LinkedMDB, NewYork Times, and the Life Science collection. In summary, the Linked Data queries vary in a broad range of characteristics, such as number of sources involved, number of query results, and query structure.

Query Characteristics. Table 2 surveys the query properties, showing that they vastly vary in their characteristics. We indicate the SPARQL operators used inside the query (*Op.*), the solution modifiers that were used additionally (*Sol.*), categorize the query structure (*Struct.*), roughly distinguishing different join combinations – like subject-subject or subject-object joins – leading to different query structures commonly referred to as star-shaped, chain, or hybrid queries, and indicate the number of results (*#Res.*) on the associated datasets (we provide an estimated lower bound when the precise number is unknown). In addition, we denote the number of datasets that potentially contribute to the result (*#Src*) , i.e. those that match at least one triple pattern in the query. Note that the number of data sets used for evaluation depends on the evaluation strategy (e.g., an engine may substitute variable bindings at runtime and, in turn, some endpoints would no longer yield results for it), or intermediate results delivered by endpoints may be irrelevant for the final outcome. We observe that the (LS) queries typically address 2–3 sources, the (CD) queries up to 5 sources, while the (SP) queries have intermediate results in up to 16 sources (where, however, typically only few sources contribute to the result).

3.3 Benchmark Evaluation Framework

To help users executing FedBench in a standardized way and support parametrization along the dimensions from Section 2, we have developed a Java benchmark driver, which is available in Open Source. It provides an integrated execution engine for the different scenarios and is highly configurable. Using the Sesame[3] API as a mediator, it offers support for querying local repositories, SPARQL endpoints, and Linked Data in a federated setting. Systems that are not built upon Sesame can easily be integrated by implementing the generic Sesame interfaces.

The driver comes with predefined configurations for the benchmark scenarios that will be discussed in our experimental results. Custom scenarios can be created intuitively by writing config files that define properties for data configuration and other benchmark settings (query sets, number of runs, timeout, output mediator, etc). In particular, one can specify the types of repositories

[3] `http://www.openrdf.org/`

(e.g., native vs. SPARQL endpoints), automatically load datasets into repositories (while measuring loading time), and execute arbitrary queries while simulating real-world conditions like execution via HTTP with a customizable network delay. Combining this flexibility with the predefined data and query sets thus allows the user to customize the benchmark along the dimensions relevant for the setting under consideration. Designed with the goal to position the benchmark as an ongoing community effort, the underlying evaluation framework is Open Source and has been designed with extensibility in mind at different levels. In particular, it is easy to specify complex evaluation settings by means of simple configuration files (i.e., without code modifications), plug in new systems, implement new metrics, evaluate and visualize results, etc.

To standardize the output format, the driver ships two default mediators for writing results in CSV and RDF; for the latter we have implemented an Information Workbench [12] module to visualize benchmark results automatically.

4 Evaluation

The central goal of our experimental evaluation is to demonstrate the usefulness of our benchmark suite. Thus, in order to show that our framework is a useful tool to assess strengths and weaknesses in a variety of semantic data use cases, we investigate different scenarios that vary in the dimensions sketched in Section 2, in particular in data distribution, access interfaces, and query complexity. For space limitations, aspects (Q2) *Result Completeness* and (Q3) *Ranking* are not covered; further, there are currently no systems that improve their behavior when an increasing amount of statistics are provided, so (D4) *Data Statistics* could only be assessed by comparing systems that make use of different statistics. All experiments described in the following are supported by our benchmark driver out-of-the-box and were realized by setting up simple benchmark driver config files specifying data and query sets, setup information, etc. We refer the interested reader to [23] for additional results on other systems, such as DARQ and FedX. We start with a description of the scenarios, then discuss the benchmark environment, and conclude with a discussion of the evaluation results.

4.1 Description of Scenarios

(A) RDF Databases vs. SPARQL Endpoints. This first set of scenarios was chosen to demonstrate the capabilities of FedBench to compare federation approaches for data stored in local RDF databases or accessible via SPARQL endpoints. In particular, they were designed to test how dimensions (D1) *Physical Distribution* of data and (D2) *Data Access* Interfaces affect query evaluation while the remaining dimensions are fixed across the scenarios, namely

(A1) centralized processing, where all data is held in a local, central store, vs.
(A2) local federation, where we use a federation of local repository endpoints, all of which are linked to each other in a federation layer, vs.

(A3) a federation of SPARQL endpoints, also linked to each other in a common local federation layer, pursuing the goal to test the overhead that is imposed by the SPARQL requests exchanged over the associated HTTP layer.

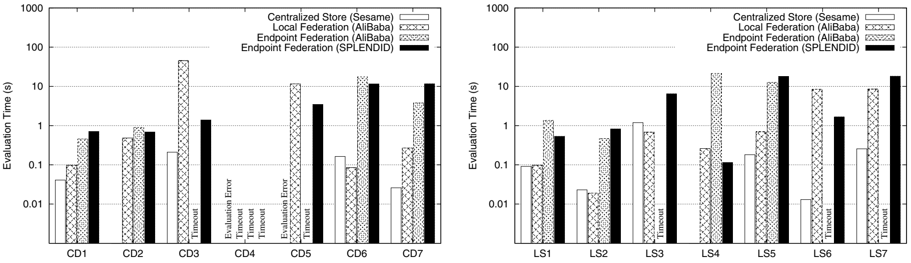
For all scenarios, we carried out experiments with the Sesame 2.3.2 engine using AliBaba version 2.0 beta 3, a federation layer for the Sesame framework which links integrated federation members together. In addition, we carried out experiments with the SPLENDID federation system from [9]. In contrast to the AliBaba federation, the latter uses statistical data about predicates and types to select appropriate data sources for answering triple patterns, which offers a wide range of optimization opportunities. Patterns that need to be sent to the same source are grouped and the join order between them is optimized with a dynamic programming approach, using e.g. the number of distinct subjects and objects per predicate to estimate the cardinality of intermediate results. The evaluation strategy relies on hash joins to allow for parallel execution and to reduce the number of HTTP requests, instead of sending individual result bindings to endpoints in a nested-loop join.

(B) Linked Data. Complementing the previous scenarios, we also evaluated a Linked Data scenario, where the data is distributed among a large number of sources that can only be retrieved using URI lookup and queries are evaluated on the combined graph formed by the union of these sources. Hence, in this setting the focus is on the knowledge about (D3) *Data Source Existence*. When all relevant sources are known, all of them are retrieved using their URIs before executing the query on the retrieved data (*top-down*) [14]. Another scheme that does not require a priori knowledge about data sources is an exploration-based approach [15]. Here, the query is assumed to contain at least one constant that is a URI. This URI is then used for retrieving the first source, and new sources are iteratively discovered in a bottom-up fashion starting with links found in that source (*bottom-up*). The (*mixed*) approach in [16] combines bottom-up and top-down to discover new sources at runtime as well as leverage known sources.

All three approaches in scenario (B) were evaluated based on the Linked Data query set (LD) using the prototype system from [16], which implements a stream-based query processing engine based on symmetric hash join operators. Note that all three approaches yield complete results (by design of the queries).

4.2 Setup and Evaluation Metrics

All experiments were carried out on an Integrated Lights-Out 2 (ILO2) HP server ProLiant DL360 G6 with 2000MHz 4Core CPU and 128KB L1 Cache, 1024KB L2 Cache, 4096KB L3 Cache, 32GB 1333MHz RAM, and a 160GB SCSI hard drive. They were run on top of a 64bit Windows 2008 Server operating system and executed with a 64bit Java VM 1.6.0_22 (all tested systems were Java-based). In the centralized setting (A1) we reserved a maximum of $28GB$ RAM to the VM, while in the distributed settings we assigned $20GB$ to the server process and $1GB$ to the client processes (i.e., the individual endpoints). Note that we run



	CD1	CD2	CD3	CD4	CD5	CD6	CD7	LS1	LS2	LS3	LS4	LS5	LS6	LS7
Endpoint Federation (AliBaba)	27	22	93k	(372k)	(117k)	6k	2k	13	61	(410)	21k	17k	(130)	(876)
Endpoint Federation (SPLENDID)	22	9	20	(9)	15	2465	492	2	49	9	10	4778	322	4889

Fig. 2. Scenario (A): Evaluation Time (top) and Number of Requests to Endpoints (bottom) for the Cross Domain and Life Science Queries

all tests against local SPARQL endpoints, one for each federation member, to avoid unpredictable effects like network delay and high load of remote sources.

All query mixes in each setup have been run three times with a timeout of ten minutes per query. We report on the average total time over all runs. Queries that failed (e.g. with a system-internal exception) or delivered an unsound or incomplete result are indicated by “Evaluation Error”. To exclude the influence of cold start effects, for each setup we ran a “ramp-up” query mix prior to query evaluation. We executed each query set separately and in order, counting the number of results (but not materializing them on disk). In addition to the evaluation time, we counted the number of requests sent to the individual endpoints (which is supported by our benchmark driver out-of-the-box).

For the Linked Data scenario, a CumulusRDF[4] Linked Data server was used to deploy the dataset on the local network. Both the server process and the Linked Data server were started with a maximum of 10GB RAM reserved. To simulate internet conditions, an artificial delay of 750ms was introduced, which resembles typical response times of current Linked Data servers.

4.3 Experimental Results

Figure 2 summarizes our results for the cross domain (CD) and life science (LS) queries in scenarios (A1)–(A3). The two plots visualize evaluation time, while the table at the bottom shows the number of requests sent by the systems to the federation members (numbers in parentheses are lower bounds for queries that failed due to timeout). Comparing Sesame and AliBaba first, we observe that the centralized approach in almost all cases outperforms the local federation approach, for 5 out of 14 queries even by an order of one magnitude or more (the time for the centralized store for queries CD2 and LS4 was about 1ms, which is not visible in the diagram). We observe an additional performance overhead for the AliBaba SPARQL Endpoint federation approach, which delivered results

[4] `http://code.google.com/p/cumulusrdf/`

only for 8 out of the 14 queries. Upon closer investigation, we could identify several reasons for the poor performance of the AliBaba federation approaches:

- Due to lack of statistics, effective join order optimization is often impossible, resulting in a high number of triples being exchange in the federation.
- Also caused by the lack of statistics, the AliBaba federation layer iteratively sends triple patterns to all federation members, to obtain candidate bindings for free variables. These bindings are then instantiated in the remaining part of the query, subsequently sending instantiated triple patterns to the federation members in a nested-loop fashion. This often results in a very high number of requests to the federation members (cf. the table at the bottom), which cause the high evaluation time. Given that AliBaba's strategy is identical in the local and SPARQL Endpoint federation scenario, the results indicate an enormous overhead imposed by the HTTP layer in the SPARQL Endpoint federation, typically in the order of one magnitude or more.
- Sesame's ability to deal with parallelization is limited. In the SPARQL Endpoint scenario, where our driver simulates endpoints by servlets that process incoming HTTP requests, we experimented with different degrees of parallelization. When instantiating more than 5–10 worker threads for answering the queries (each having its own repository connection), we could observe a performance drop down, manifesting in high waiting times for the worker threads, probably caused by Sesame's internal locking concept.

For the SPLENDID federation, we can observe that the number of HTTP requests is significantly lower: in contrast to AliBaba, which evaluates the query starting with a single triple pattern and iteratively substitutes results in subsequent patterns, SPLENDID also generates execution plans which send the patterns independently to relevant endpoints and join them together locally, at the server. Therefore, SPLENDID still returns results where AliBaba's naive nested-loop join strategy times out. For queries CD3, CD5, LS4 and LS6 it even beats the local federation.

Figure 3 summarizes our results in the SP^{2}Bench scenario. We observe that even the centralized Sesame store has severe problems answering the more complex queries, which is in line with previous investigations from [21].[5] Except for the outlier query SP1, the trends are quite similar to those observed in the (LS) and (CD) scenario, with the centralized scenario being superior to the local federation, which is again superior to the SPARQL Endpoint federation. Query SP1 is a simple query that asks for a specific journal in the data set. It can be answered efficiently on top of the local federation, because the federation is split up by type and the journals are distributed across only two federation members, so the system benefits from the parallelization of requests. In summary, the SP^{2}Bench experiments show that, for more complex queries and federation among a larger number of federation members (as indicated in Table 2), current federation approaches are still far from being applicable in practice.

[5] In the experiments from [21] Sesame was provided with all possible index combinations, whereas in these experiments we use only the standard indices. This explains why in our setting Sesame behaves slightly worse than in the experiments from [21].

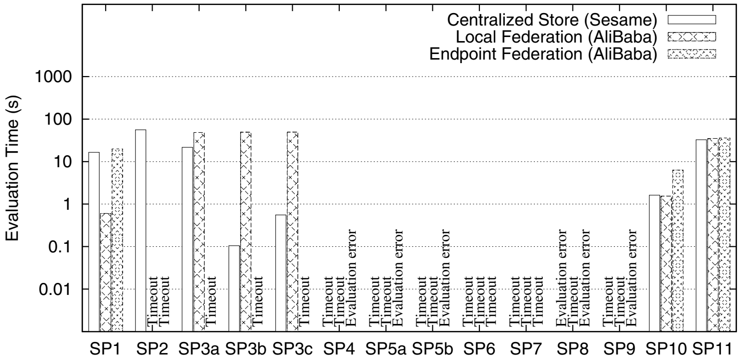
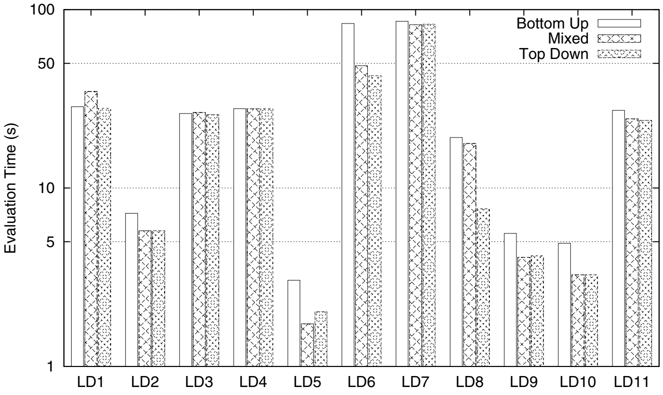


Fig. 3. Scenario (A): Results of SP^{2}Bench Queries



Query	#Srcs.
LD1	317
LD2	67
LD3	317
LD4	342
LD5	39
LD6	598
LD7	1219
LD8	217
LD9	59
LD10	44
LD11	366

Fig. 4. Scenario (B): Query Evaluation Times (left) and Number of Sources Dereferenced for Answering the Query (right)

Figure 4 visualizes our results for the Linked Data scenario (B). Regarding overall query time, the bottom-up and mixed approaches behave similarly: both perform run-time discovery and the mixed strategy cannot use its partial knowledge to restrict sources, but only to load relevant sources earlier. This leads to earlier result reporting, but is not reflected in the overall query time. In some cases the mixed approach is even slightly worse than bottom-up, due to the overhead imposed by using local source indices. The top-down approach, though, is able to restrict the number of sources to be retrieved, leading to better query times in many cases. For example, for query LD8 the query time for bottom up evaluation is 19.1s, while top-down requires 4.2s, an improvement of 75%, made possible by the lower number of sources retrieved in the top-down scenario.

Overall, the top-down approach uses its centralized-complete knowledge to identify relevant sources and exclude non-relevant sources. In a dynamic scenario, though, such as Linked Data, it may be infeasible to keep local indexes up-to-date, so exploration-based approaches like bottom-up or the mixed approach, which do not rely on complete knowledge, may be more suitable.

5 Conclusion

As witnessed by the evaluation, our benchmark is flexible enough to cover a wide range of semantic data application processing strategies and use cases, ranging from centralized processing over federation to pure Linked Data processing. Clearly, our experimental analysis is not (and does not intend to be) complete with respect to covering all existing systems and solutions – yet we have provided a flexible benchmark suite that can be used and extended by others to evaluate alternative approaches, systems, and scenarios.

Extensions we are planning to address in future work particularly include queries targeted at the new SPARQL features that will be published in the coming SPARQL 1.1 release such as aggregation, nested subqueries, and built-in support for federation. Further, given that the current data sets and queries focus on instance data and query answering over ground RDF graphs, extensions for testing reasoning capabilities (e.g., over RDFS and OWL data) in a distributed setting are left as future work. With our flexible framework, though, it is straightforward to incorporate such scenarios with little effort, and we invite the community to contribute to FedBench with own data and query sets.

Finally, our evaluation has revealed severe deficiencies of today's federation approaches, which underline the practicability of FedBench. As one of our major findings, data statistics – which are explicitly included in our benchmark suite – play a central role in efficient federated data processing: as indicated by our results, they are crucial in optimization to minimize the size of results shipped across federation members and requests exchanged within the federation.

Acknowledgments. Research reported in this paper was partially supported by the German BMBF in the project CollabCloud. `http://collabcloud.de/`

References

1. SPARQL 1.1 Federation Extensions. W3C Working Draft (June 1, 2010), `http://www.w3.org/TR/sparql11-federated-query/`
2. Abadi, D.J., Marcus, A., Madden, S.R., Hollenbach, K.: Scalable Semantic Web Data Management Using Vertical Partitioning. In: VLDB (2007)
3. Alexander, K., Hausenblas, M.: Describing Linked Datasets – On the Design and Usage of voiD. In: Linked Data on the Web Workshop (2009)
4. Angles, R., Gutierrez, C.: The Expressive Power of SPARQL. In: Sheth, A.P., Staab, S., Dean, M., Paolucci, M., Maynard, D., Finin, T., Thirunarayan, K. (eds.) ISWC 2008. LNCS, vol. 5318, pp. 114–129. Springer, Heidelberg (2008)
5. Buil-Aranda, C., Arenas, M., Corcho, O.: Semantics and Optimization of the SPARQL 1.1 Federation Extension. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 201. LNCS, vol. 6644, pp. 1–15. Springer, Heidelberg (2011)

6. Bizer, C., Schultz, A.: The Berlin SPARQL Benchmark. Int. J. Semantic Web Inf. Syst. 5(2), 1–24 (2009)
7. Broekstra, J., Kampman, A., van Harmelen, F.: Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, pp. 54–68. Springer, Heidelberg (2002)
8. Duan, S., Kementsietsidis, A., Srinivas, K., Udrea, O.: Apples and Oranges: A Comparison of RDF Benchmarks and Real RDF Datasets. In: SIGMOD (2011)
9. Görlitz, O., Staab, S.: Federated Data Management and Query Optimization for Linked Open Data. In: New Directions in Web Data Management (2011)
10. Gray, J.: Database and Transaction Processing Performance Handbook. In: The Benchmark Handbook (1993)
11. Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. J. Web Sem. 3(2-3), 158–182 (2005)
12. Haase, P., Eberhart, A., Godelet, S., Mathäß, T., Tran, T., Ladwig, G., Wagner, A.: The Information Workbench - Interacting with the Web of Data. Technical report, fluid Operations & AIFB Karlsruhe (2009)
13. Haase, P., Mathäß, T., Ziller, M.: An Evaluation of Approaches to Federated Query Processing over Linked Data. In: I-SEMANTICS (2010)
14. Harth, A., Hose, K., Karnstedt, M., Polleres, A., Sattler, K.-U., Umbrich, J.: Data Summaries for On-Demand Queries over Linked Data. In: WWW (2010)
15. 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)
16. 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)
17. Ladwig, G., Tran, T.: SIHJoin: Querying Remote and Local Linked Data. In: Antoniou, G., Grobelnik, M., Simperl, E., Parsia, B., Plexousakis, D., De Leenheer, P., Pan, J. (eds.) ESWC 2011, Part I. LNCS, vol. 6643, pp. 139–153. Springer, Heidelberg (2011)
18. Magkanaraki, A., Alexaki, S., Christophides, V., Plexousakis, D.: Benchmarking RDF Schemas for the Semantic Web. In: Horrocks, I., Hendler, J. (eds.) ISWC 2002. LNCS, vol. 2342, p. 132. Springer, Heidelberg (2002)
19. Neumann, T., Weikum, G.: Rdf-3X: a RISC-style engine for RDF. PVLDB 1(1) (2008)
20. Pérez, J., Arenas, M., Gutierrez, C.: Semantics and Complexity of SPARQL. ACM Trans. Database Syst. 34(3) (2009)
21. Schmidt, M., Hornung, T., Lausen, G., Pinkel, C.: SP2Bench: A SPARQL Performance Benchmark. In: ICDE, pp. 222–233 (2009)
22. Schmidt, M., Meier, M., Lausen, G.: Foundations of SPARQL Query Optimization. In: ICDT, pp. 4–33 (2010)
23. Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: Optimization Techniques for Federated Query Processing on Linked Data. In: Aroyo, L., et al. (eds.) ISWC 2011, Part I. LNCS, vol. 7031, pp. 601–616. Springer, Heidelberg (2011)
24. Weiss, C., Karras, P., Bernstein, A.: Hexastore: Sextuple Indexing for Semantic Web Data Management. PVLDB 1(1), 1008–1019 (2008)