Semantic Web 1 (2016) 1–5
IOS Press

A Benchmark Suite for Federated SPARQL Query Processing from Existing Workflows

Antonis Troumpoukis [a] Angelos Charalambidis [a] Giannis Mouchakis [a] Stasinos Konstantopoulos [a]
Daniela Digles [b] Ronald Siebes [c] Victor de Boer [c] Stian Soiland-Reyes [d]

[a] *Institute of Informatics and Telecommunications, NCSR 'Demokritos', Greece*
e-mail: {antru,acharal,gmouchakis,konstant}@iit.demokritos.gr
[b] *Department of Pharmaceutical Chemistry, University of Vienna, Austria e-mail: daniela.digles@univie.ac.at*
[c] *VU University Amsterdam, the Netherlands e-mail: {r.m.siebes,v.de.boer}@vu.nl*
[d] *eScience Lab, The University of Manchester, UK uri: http://orcid.org/0000-0001-9842-9718*

Abstract. This paper presents a new benchmark suite for SPARQL query processors. The benchmark is derived from workflows established by the pharmacology community and exploits the fact that these workflows are not only applied to voluminous data, but they are also equivalent to complex and challenging queries. The value of this queryset is that it realistically represents actual community needs in a challenging domain, testing not only speed and robustness to large data volumes but also all features of modern query processing systems. In addition, the natural partitioning of the data into meaningful datasets makes these workflows ideal for benchmarking *federated* query processors. This emphasis on federated query processing drived complementing the benchmark with an execution engine that can reproduce distributed and federated query processing experiments.

Keywords: Triple store benchmarking, Pharmacology data, Distributed and federated querying.

1. Introduction

Performance benchmarks allow systems to be evaluated and compared, but designing such a benchmark is subject to considerations that are difficult to satisfy simultaneously. For query processing systems in particular, one such consideration is the selection of the data that will be included in the benchmark and the query workload that will be applied to this data. One potential tension is the creation of a *realistic* benchmark that reflects workflows that occur commonly versus the creation of a *generic* and *informative* benchmark that tests as many characteristics of query processing systems as possible. One would expect the former to be useful for selecting what query processing infrastructure to use for specific domains and applications, and the latter to be a generic, multi-dimensional tool for evaluating the technical quality of an infrastructure.

Most well-known benchmarks use natural datasets but define an artificial queryset based on what technical characteristics should be tested [15,13], while some benchmarks also provide synthetic data [9]. However, considering the complex and multi-dimensional nature of modern database systems, fine-tuning generic benchmarks to specific applications can prove to be too difficult and too prone to human bias: a database system can be fine-tuned to perform well on a given dataset and query load, in which case measuring performance on an artificial problem is a lot less informative for deciding what infrastructure to use than measuring performance on a natural problem. A promising compromise could be to design benchmarks that are derived from realistic workflows, but preferring among all possible workflows those that measure many different technical aspects and functionalities of the tested systems. The problem is that realistic workflows are typically based on visual interfaces that are used to fill the parameters of extremely simple query templates. Such queries could be used to measure robustness and reactiveness on large data volumes, but are

1570-0844/16/$35.00 © 2016 – IOS Press and the authors. All rights reserved

not very informative about the ability to optimize complex queries.

The bio-medical domain is one exception where complex queries occur naturally. Within this domain, the Open PHACTS project [17] has put together the datasets and workflows that aim to answer scientific competency questions that were collected to represent standard use cases for drug discovery. The Open PHACTS project has also developed visual tools that allow domain experts to combine query templates in non-trivial ways. Through these tools, domain experts have defined workflows that are equivalent to technically challenging queries, as well as independently motivated and representative of common and frequently used workflows in a challenging domain.

In this article we will first briefly present the data and the drug discovery questions that this data can answer (Section 2). We then proceed to present the translation of the Open PHACTS workflows that target these questions into SPARQL queries, together with an analysis regarding the components and features of federated query processors that these queries test (Section 3). This static analysis is complemented by using the Open PHACTS benchmark to evaluate the state-of-the-art federated SPARQL query processing systems and comparing the results against the results obtained by the same systems over prior benchmark suites (Section 5). Section 5 also describes the experimental setup, including briefly introducing the main features of the KOBE Benchmarking Engine used to execute the experiments, with more technical details about KOBE given in the appendix. The article closes with a comparison with related benchmarks (Section 6) and conclusions (Section 7).

2. The Open PHACTS platform and the '20 questions' approach

The Open PHACTS initiative aims to integrate publicly available data relevant for both academia and the pharmaceutical industry. The core outcome of Open PHACTS is the *Open PHACTS Discovery Platform (ODP)* that provides an easy interface through which researchers can consult the database without being confronted with the complexity of defining efficient Linked Data queries. For the end-user, the platform offers a set of services which are accessible via a RESTful interface. The choice of the services is based on consulting the domain experts among the Open PHACTS project consortium on which questions are most relevant to them when doing their daily research tasks. Through this process, twenty key questions were identified [3] which combine four important pharmacological concepts: compound, target, pathway and disease. Compounds are usually small molecules which can influence targets by activating or inhibiting them (bioactivity). These targets (often proteins) are important for many functions of organisms and are often part of cellular pathways by interacting with other entities (both molecules and targets). Errors in the function of the targets can disturb cellular pathways and lead to diseases. The aim of a drug discovery process is usually to identify a compound which can be used as the active ingredient in drugs. These can be used to restore the correct function of the targets with the aim to cure the disease or at least lower its symptoms. For this, a good understanding of the connections between the four concepts is necessary.

The Open PHACTS Discovery Platform provides an interpretation of these questions as *workflows* that are authored using visual tools. Workflows retrieve data via API calls. These API calls correspond to SPARQL query templates which are instantiated by the parameters of the API call [10]. The platform executes the resulting instantiated queries at an endpoint that serves relevant data [8,5].

Figure 1 depicts the relevant datasets and their interconnections. Each of the datasets adds a different perspective of data which is needed to answer the questions:

- *UniProt* collects sequence and functional data of proteins, providing commonly used identifiers of proteins through their accession codes.
- *ChEMBL* provides bioactivity data, which is of high importance in many of the questions, where literature is curated to collect activity of molecules against targets (often proteins).
- *DrugBank* provides information on drug molecules, such as the approval status for clinical trials.
- *DisGeNET* associates genes and diseases.
- *WikiPathways* is a collection of cellular pathways which can be edited by the scientific community.
- Two ontologies, the *Gene Ontology* for proteins, and *ChEBI* for compounds provide additional annotations of the respective entities.

Some of the datasets specifically focus on mapping entities from different data sources:

- The *OPS Chemical Registry* standardizes molecules from the different data sources in Open PHACTS,

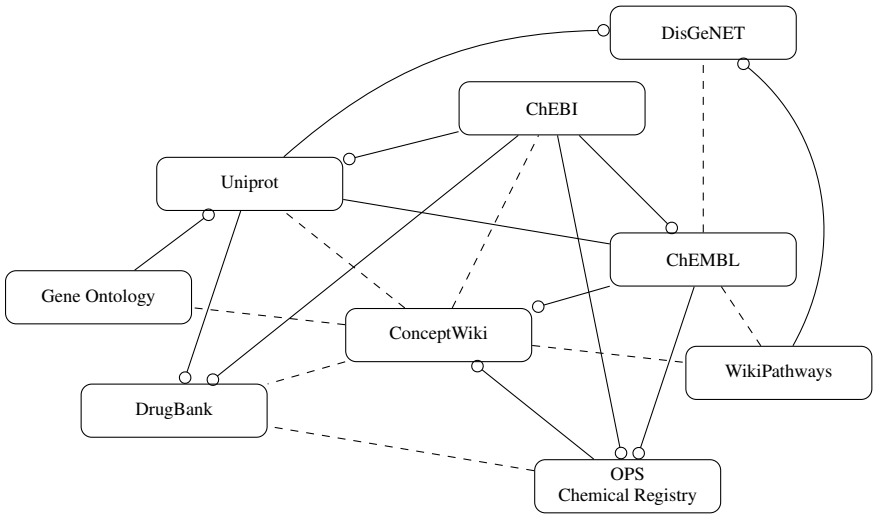

Fig. 1. Open PHACTS datasets and their linkage. Solid lines connect datasets that share common URIs. Solid lines with circle connect datasets that are linked through linksets, with the circle showing which of the two datasets contains the linkset. Dashed lines connect datasets that are linked through the Open PHACTS IMS.

to provide a single identifier if the structures are identical.
- Similarly, *ConceptWiki* contains labels for entities of the different datasets, allowing text searches in the Open PHACTS Discovery Platform.

Finally, the *Open PHACTS Identity Mapping Service (IMS)* is a service that facilitate the mappings between identifiers of the aforementioned datasets. Originally, Open PHACTS IMS is offered as a service and not as a materialized dataset. However, the IMS is in fact based on a collection of different linksets and therefore we will treat it as a separate dataset that contain solely triples with equivalence predicates such as `skos:exactMatch` and `owl:sameAs`. The assumptions of using the IMS as a dataset will be described in the Section 3. Note how in Figure 1 the IMS is not shown as a dataset but as a type of connection. This is both for visual clarity and also for emphasising that the IMS does not contain any domain knowledge but only linksets.

Table 1 displays some basic statistics for each dataset. Firstly, we display the total number of triples, predicates, distinct subjects, distinct objects and classes. These statistics, known as *VoiD statistics* [1], give a rough idea about the size and the structure of the dataset, and are generally exploited by SPARQL engines in order to perform certain tasks, such as selection and optimization. Moreover, we have computed the *structuredness* for each dataset [6], a metric that shows whether the instances of a class contain triples of all the properties of the class. Typically, artificial datasets are highly structured while natural datasets are less structured. The metric ranges in $[0, 1]$ and a smaller number means that a dataset is less structured. The structuredness of a dataset is considered one of the most important characteristics of a benchmark, which is used especially for drawing comparisons between benchmarks.

3. Deriving Queries from Workflows

The Open PHACTS initiative tried to answer the twenty key questions by developing scientific workflows that access different datasets through API calls to get intermediate results and then transform, filter, and join these results into an answer to the question.

The questions that have been transformed into SPARQL queries and included in the OPFBench, the benchmark described here, are listed in Table 2. A detailed explanation of the questions is available elsewhere [3] and is outside the scope of this article. As an example we will explain Q19, which searches for compounds that hit most specifically the multiple targets in a given pathway (disease). In a biological pathway, sometimes one target can take over the role of another one. So even if a compound could effectively inhibit a target in a disease pathway, there would be no (or only minimal) biological effect in this case, as the function is compensated by another target. So the aim is to find compounds that can inhibit several tar-

Table 1
Dataset statistics

Dataset	#triples	#subjects	#predicates	#objects	#classes	structuredness
Uniprot	1,131,186,434	235,053,262	122	322,660,114	126	0.391
ChEMBL	445,732,880	54,923,033	146	118,629,007	120	0.912
OPS Chemical Registry	241,986,722	38,555,884	18	89,882,844	27	0.896
DisGeNET	17,791,631	1,367,616	77	4,891,477	26	0.877
OPS Identity Mappings	14,431,716	5,254,745	71	10,874,931	27	0.387
WikiPathways	11,781,627	871,000	110	1,467,010	32	0.810
DrugBank	5,478,852	330,274	104	1,917,893	99	0.587
ConceptWiki	4,331,760	3,024,393	4	4,319,478	–	–
Gene Ontology	1,366,494	234,770	45	430,020	7	0.547
ChEBI	1,012,056	113,446	22	651,682	5	0.816
Total	1,875,100,172	339,728,423	719	555,724,456	469	avg. 0.682

Table 2
Multi-domain drug-discovery questions expressed in natural language.

Query	Question expressed in natural language
Q1	Give me all oxidoreductase inhibitors active <100 nM in human and mouse.
Q3	Given a target find me all actives against that target, and find and/or predict the polypharmacology of actives.
Q6	For a specific target family, retrieve all compounds in specific assays.
Q7	For a target, give me all active compounds with the relevant assay data.
Q8	Identify all known protein-protein interaction inhibitors.
Q9	For a given compound, give me the interaction profile with targets.
Q15	Which chemical series have been shown to be active against target X?
Q15b	Which new targets have been associated with disease Y?
Q16	Targets in Parkinson's disease or Alzheimer's disease are activated by which compounds?
Q18	For pathway X, find compounds that agonize targets assayed in only functional assays with potency $< 1\mu M$.
Q19	For the targets in a given pathway, retrieve the compounds that are active with more than one target.

gets of the same pathway.[1] To answer the above question one must access the WikiPathways to retrieve the connection between a disease pathway and a compound and cross link it with the with the ChEMBL and OPS Chemical Registry datasets to retrieve the targets and their chemical representation for a certain compound. Since ChEMBL and WikiPathways are not directly connected, the workflow must rely on the Open PHACTS IMS service for the appropriate linkage between the compound identifiers. We will revisit Q19 later in this article as an example for demonstrating technical points.

Although in principle Open PHACTS workflows are more expressive than relational algebra, the workflows actually defined for answering the pharmacology questions only use basic data processing and are within the expressivity of relational algebra and SPARQL. For OPFBench, the benchmark described here, ten workflows are expressed as single SPARQL queries (Table 2). Similarly to Q19, almost all of the SPARQL queries we have derived from the workflows involve more than one dataset. The characteristics of each query will be presented in Section 4.

There are two reasons why not all of the original 20 questions are included in the OPFBench queries: either no corresponding workflow has been defined, or the corresponding workflow uses dynamic API calls that express relations beyond data access.

Regarding workflow availability, there are questions that do not have a corresponding workflow because they could not be answered from the original datasets or the data was out of scope of the Open PHACTS

[1] It should be noted that this is only one possible interpretation of the original wording of Question 19, the interpretation that is implied by the Open PHACTS workflow developed to answer Question 19.

project. Some datasets (e.g., on patents and pathway interactions) are now included in the Open PHACTS Discovery Platform, but have not been available when the workflows were created. Workflows for some of the missing questions (e.g., Q12 and Q17) are currently being considered under the light of the API calls that have been developed since the inclusion of new datasets. A second reason why a question has no corresponding workflow defined by Open PHACTS is that sometimes multiple questions are deemed to be sufficiently addressed, at least for the time being, by the same workflow. For example, the workflow that answers question #7, also answers question #17 to some extent.

Besides missing workflows, there are also workflows that use external services such as *Identity Mapping Service (IMS)* and the *Similarity and Structural Search Service* of the Open PHACTS Discovery Platform. These services can in principle produce responses dynamically and therefore it is not straightforward how, and if, the responses can be mapped into a materialized dataset. A notable exception is the Identity Mapping Service that is initialized by a collection of known linksets. For the purposes of the benchmark we have used those linksets to create a materialized *OPS Identity Mappings* dataset (Table 1). We also produced and imported the transitive closure of the equivalence triples, namely the triples that use the predicates `skos:exactMatch` and `owl:sameAs`. On the other hand we have excluded the workflows that make use of the similarity and structural search service and we will consider ways of materializing it in the future.

4. Query Characteristics

The queries derived from the scientific workflows require multiple datasets to be accessed in order to compute an answer and therefore it is natural to consider using the selected queryset to evaluate federated SPARQL query processing systems. Obviously this does not exclude the benchmarking of triple stores. In that case all the datasets should be loaded in the same system organized in different graphs. However, we focus our attention to federated SPARQL query processing systems and in this section we discuss the suitability of the proposed queryset for such benchmarking.

The typical flow of a federated query processing system consists of three phases, namely the source selection, the query planning and lastly the query execution. All the phases contribute to the overall efficiency of the query processing system and it would be desirable to have queries that will challenge all those phases. Naturally, different characteristics of a query will stress different phases of the federated query processing system.

The proposed queries vary on complexity, on the number of datasets that are involved and on the SPARQL features needed. Table 3 and 4 collect the different characteristics of each query. Table 3 focuses on the characteristics from a syntactic point of view (e.g. number of triple patterns, types of SPARQL operators etc.) while Table 4 focuses on statistics that have to do with data-driven characteristics, such as the number of expected results or the relevant sources for each query.

4.1. Source selection discussion

In the Open PHACTS Discovery Platform, all the available datasets are loaded into a single triple store and are originally organized in different named graphs. In our setup, we have created one data source for each graph. Since graph annotations directly map to data sources, providing them trivializes the source selection phase. On the other hard, graph annotations are meaningful restrictions: since some triple patterns succeed in more than one graph, dropping graph restrictions produces more results than intended. For this reason, we decided to create two versions for each query, the original one and one without graph annotations for testing source selection. As expected, there are differences in the size of the resultset (ref. #results columns in Table 4).

Every query involves between one and five datasets to compute the result (ref. #sources columns in Table 4, for each version of our queryset). This is relevant for benchmarking if the source selection phase prunes efficiently the irrelevant datasets. Unfortunately most of the predicates used in the queries exist in only one dataset and as a result in most cases they uniquely identify the associated dataset if the source selection exploits such relations [12]. On the other hand, triples with common predicates exist in every dataset (e.g. `rdf:type`) but will not be joinable with any other dataset except the one that will be found. Table 4 shows the number of sources that contain tuples that potentially contribute to the resultset in #potential sources column versus the number of source that really contribute to the resultset. As this difference increases, the significance of correctly identifying the irrelevant sources also increases.

Table 3
Structural characteristics of the OPFBench queryset

Query	#patterns	join vertices					mean join vertex degree	SPARQL Features
		star	chain	sink	hybrid	total		
Q1	8	1	1	1	–	3	3.33	FILTER, UNION
Q3	15	2	2	1	2	7	3	BIND, DISTINCT, FILTER, VALUES
Q6	11	3	–	2	1	6	2.67	OPTIONAL
Q7	16	3	1	1	2	7	3.14	BIND, FILTER
Q8	9	2	1	1	1	5	2.6	FILTER
Q9	12	2	1	1	–	4	3.5	BIND, FILTER, VALUES
Q15	12	1	6	1	–	8	2.25	COUNT, GROUP BY, UNION
Q15b	6	1	1	–	–	2	3	–
Q16	11	2	4	2	–	8	2.25	FILTER, VALUES
Q18	15	2	2	3	1	8	2.75	BIND, FILTER, OPTIONAL
Q19	16	3	2	3	2	10	2.5	COUNT, FILTER, GROUP BY, HAVING, REGEX

Table 4
Data-Driven characteristics of the OPFBench queryset

Query	#potential sources	#sources span	with graphs		without graphs		mean triple pattern selectivity
			#sources	#results	#sources	#results	
Q1	9	9	1	331,600	2	477,540	0.033
Q3	10	9	3	195	3	6,830	0.032
Q6	8	8	4	3,154,375	4	3,154,375	0.064
Q7	8	5	4	161	4	5,508	0.059
Q8	8	7	2	21,881	2	21,881	0.059
Q9	10	6	3	252	4	1,472	0.045
Q15	10	10	3	242	3	242	0.050
Q15b	5	1	1	164	1	164	0.036
Q16	3	3	3	6,317,773	3	6,317,773	0.024
Q18	8	6	3	18,298	4	36,596	0.020
Q19	8	7	4	6,073	4	6,155	0.039

4.2. Query planning discussion

Another characteristic of the proposed queries is that they contain a large number of triple patterns in order to retrieve the required information. The queries contain up to 16 triple patterns per query (ref. #patterns column in Table 3). This is commonly encountered in SPARQL queries in contrast to SQL queries. Traditional join optimization techniques derived from relational databases cannot cope efficiently with a large number of joined relations. Therefore, the large number of triple patterns will challenge the join optimization phase of a federated query processing system. Moreover, the queryset contains queries of various join types. The types of joins are shown in Table 3. A common variable between two or more triple patterns is sometimes referred as a *join vertex*. A join vertex is a *star* if it appears only as a subject, a *sink* if it appears only as an object, a *chain* if it appears only twice, once as a subject and once as an object. If a join vertex appears both as a subject and as an object, but is not a path, then it is referred as a *hybrid* vertex. Finally, the parameter *mean join vertex degree* is the average of the number of all triple patterns that a vertex participates.

Apart from the number of joined triple patterns, another factor of optimization is the optimization of other SPARQL operators such as left outer joins, unions, groupings and orderings. Typically, query planners consider only inner join optimizations since this optimization is considered the most impactful. This assumption does not reflect always the reality and often leads to physical plans that force these operators to be

executed on the side of the federator than on the data stores. Interestingly enough, the realistic queries that have been derived make use of other SPARQL features which are expected to challenge the query planners of the query processing systems. The relevant operators are listed in Table 3.

4.3. Query execution discussion

There are cases where the transfer of large resultsets over the network cannot be circumvented by any valid plan optimization. In these cases, it is the efficiency of the execution engine of the federated query processing system (rather than the quality of the plan) that differentiates systems. OPFBench includes queries that produce large resultsets and challenge query execution: The size of the resultset of each query varies from as small as 164 results to as large as 6M results, as shown in Table 4.

Another relevant point for evaluating federated query execution is their behaviour when a large number of remote endpoints must be accessed. The endpoints that should be accessed in the proposed queryset vary between one to four (out of a total of ten). This is one point where the current queries do not stress the systems enough, but it should be noted that the Open PHACTS queries were authored having in mind the query processing state of the art. Further work will identify use cases (including queries pulling data from many different sources) that are relevant to the domain and challenging for query processing systems. It is tempting to synthesise an artificial query which uses almost all of the 10 endpoints, since this would cover another relevant choke point. However, we decided not to do this because it would defy our main purpose, which is to provide a benchmark that contains queries that are derived from a practical use case, and not to use an artificially constructed queryset.

4.4. Characteristic example query

A characteristic SPARQL query to demonstrate the points discussed in the previous sections is query Q19 that is listed in Listing 1. Initially, the potential sources for this query are nine, but eventually only four datasets contribute to the final answer. The linkage between the datasets is performed through the IMS dataset that corresponds to the `http://ims.openphacts.org` graph. The query is comprised of a large number of joined triple patterns, specifically 16 triple patterns joined in star, chain and sink formations.

One interesting aspect of this query is that it makes use of grouping and of filtering on an aggregate value of this grouping (lines 45–46). Even though the resultset is reasonably small, the grouping operator almost dictates that the grouping must be performed in the federated query processing system due to the fact that the grouping and the aggregating variables belong to different datasets. Thus, the intermediate result must be transferred over the network and the execution engine must cope with its size.

5. Experiments

In this section we apply our benchmark to three state-of-the-art federated SPARQL query processors in order to understand whether, and how, the benchmark challenges the state of the art.

We selected the *FedX*, *SPLENDID*, and *Semagrow* systems for this comparison. *FedX* [16] relies on an extremely fast query executor and only applies minimal, heuristic optimizations to prepare the query plan. Although this approach often produces suboptimal plans, it introduces minimal overhead to the query execution process and is very efficient in situations where more sophisticated optimizations have little impact. *SPLENDID* [7] features a slower query executor but employs a more sophisticated (but time consuming) planning algorithm to recover the planning overhead and to counter the disadvantage of its slower executor. FedX is typically benchmarked as being faster, as easier queries do not give SPLENDID an opportunity to recover the planning overheads and queries that fetch voluminous results challenge SPLENDID's query executor, leaving SPLENDID a small winning margin when smart optimization can greatly reduce the effort need to process a query [14]. Finally, *Semagrow* combines sophisticated planning with a very efficient non-blocking asynchronous stream processing executor that clearly outperforms SPLENDID and marginally outperforms FedX on the FedBench benchmark [4].

5.1. KOBE Benchmarking Engine

In order to execute the experiments on all federated querying engines, we have generalized the driver released by FedBench [15] into a general purpose benchmarking driver that can be configured for different datasets, query loads, and querying scenarios. Furthermore, we have developed a *federation composer* that

```
PREFIX  chembl:   <http://rdf.ebi.ac.uk/terms/chembl#>
PREFIX  cheminf:  <http://semanticscience.org/resource/>
PREFIX  dc:       <http://purl.org/dc/elements/1.1/>
PREFIX  dcterms:  <http://purl.org/dc/terms/>
PREFIX  rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX  skos:     <http://www.w3.org/2004/02/skos/core#>
PREFIX  wp:       <http://vocabularies.wikipathways.org/wp#>

SELECT ?smiles (COUNT(DISTINCT ?target) AS ?count)
{
  GRAPH <http://www.wikipathways.org> {
    ?rev    dc:identifier    <http://identifiers.org/wikipathways/WP382> .
    ?rev    dc:title         ?title .
    ?gene_product_internal   dcterms:isPartOf   ?rev .
    ?gene_product_internal   rdf:type           ?type .
    ?gene_product_internal   dc:identifier      ?gene_product .
    FILTER (?type = wp:GeneProduct || ?type = wp:Protein) .
    FILTER (!REGEX(?gene_product,"/DataNode/noIdentifier")) .
  }

  GRAPH <http://ims.openphacts.org/> {
    ?item   skos:relatedMatch   ?gene_product.
  }

  GRAPH <http://www.ebi.ac.uk/chembl> {
    ?targetComp   chembl:targetCmptXref       ?item .
    ?target       chembl:hasTargetComponent   ?targetComp .
    ?target       dcterms:title               ?target_name_chembl .
    ?target       chembl:organismName         ?target_organism .
    ?assay        chembl:hasTarget            ?target .
    ?assay        chembl:hasActivity          ?act .
    ?act          chembl:hasMolecule          ?compound .
    ?act          chembl:pChembl              ?pChembl .
    FILTER (?pChembl > 5).
  }

  GRAPH <http://ims.openphacts.org/> {
    ?ocrs_compound   skos:exactMatch   ?compound .
  }

  GRAPH <http://ops.rsc.org> {
    ?ocrs_compound   cheminf:CHEMINF_000018   ?smiles .
  }
}
GROUP BY ?smiles
HAVING(COUNT(DISTINCT ?target) > 1)
```

Listing 1: SPARQL query that answers Q19

automates the process of deploying the SPARQL endpoints that made up the federation on computer clusters. We are making this toolkit, the *KOBE Benchmarking Engine*,[2] publicly available as it can be more generally useful besides the experiments described here.

For these experiments, we have configured KOBE to apply the FedBench, BigRDFBench [13], and OPFBench benchmarks on all of Semagrow, FedX, SPLENDID, and, for the purposes of comparison, on a stand-alone Virtuoso database. The federations are configured to deploy each dataset as a separate Virtuoso endpoint; the stand-alone Virtuoso database has each dataset loaded as a different graph in the same one-node Virtuoso end-point. Deployment is done on a cluster with 4 computation nodes, each having a 4-core Intel(R) Xeon(R) CPU E31220 at 3.10GHz and 30 GB RAM.

The performance of FedX, Semagrow, SPLENDID Federations and the Virtuoso Store for each version of our queryset is shown in Table 5 for the version queryset that includes the graph annotations and in Table 6 for the version without graph annotations. We display the cold (first run) and the average of two subsequent runs (hot runs) for the Virtuoso Store, and one single cold run for each of the federation engines.

In many cases, a runtime error (denoted as r/e in the tables) has prevented some federators from producing any results. Such errors may have occurred in the query decomposition phase (for example, a case in which the federator could not process the input query mostly due to non-supported SPARQL operations), or in the query execution phase (for example, a case in which a huge resultset from an endpoint has caused a choke point in the query execution engine of the federator). Finally, some query evaluations were not completed due to a timeout (denoted as t/o in the tables). Query evaluation timeout is set at 1 hour. In the following paragraphs, we will discuss those errors in a more detailed way.

5.2. Supported SPARQL Operators

Even before the source selection stage, the federation engine has to parse the input query and decide whether the SPARQL operators that are present in the query can be evaluated by the system. Our queries contain many SPARQL operators, but not all of them are supported by most state-of-the-art systems. For example, all runtime errors for FedX that appear in Table 5 and all errors except Q1 in Table 6 are due to unsupported SPARQL operations. In the case of Semagrow, such an error occurs only in Q18 (in both versions of the queryset). In the case of SPLENDID, such errors occur in queries Q3, Q9 and Q16 in the version of our queryset without graph annotations (Table 6). We observed that the most important of the unsupported SPARQL operations is the VALUES operator, which cannot be processed by FedX and SPLENDID federations. In many cases it is possible that a query that contain the VALUES keyword can be transformed into an equivalent one which does not contain the VALUES keyword. However, this transformation can be problematic in some situations; for example Q3 contains a VALUES clause with six bindings, and therefore an equivalent transformation would contain six unions and the resulting query would be very large, and probably difficult to be processed by a federation engine. Since most state-of-the-art federation engines implement bind join as the most common implementation of the join operation between triple patterns, it would be helpful if the state-of-the-art started to embrace some easy to implement SPARQL 1.1 operators, such as the VALUES one.

5.3. Source Selection

Usually, the first stage of an evaluation of a federated query is the source selection, in which a federation engine selects the appropriate sources that contain data for each triple pattern. In Table 7 we illustrate the number of the sources that are accessed from each query execution plan. In most cases the federation engines fail to prune the unwanted sources for triple patterns that contain common predicates, such as `rdf:type` and `skos:exactMatch`. This can be seen by comparing the sources that are selected by the federators with those that contain the actual data, which are shown in Table 4. For example, consider the plan that is yielded by FedX for query Q1 (Listing 2). In this plan, the pattern in line 11 is evaluated in almost all sources, while in fact it only contains joinable data in ChEMBL and in Uniprot.

Regarding to the version of the queryset that contains graph annotations, FedX ignores these annotations, and therefore the plans and the execution are the same in both cases. SPLENDID performs ASK queries with syntax errors at the source selection phase and therefore fail to produce a plan in all cases, hence all runtime errors that occur in the queryset with graphs

[2] See Appendix A for more technical details.

Table 5
Query execution times (msec) and number of results for queries with GRAPH annotations.

	Virtuoso			FedX		Semagrow		SPLENDID	
Query	cold	hot	#results	cold	#results	cold	#results	cold	#results
Q1	38,645	22,325	331,600	1,681,137	477,540	66,580	331,600	r/e	–
Q3	17,983	874	195	r/e	–	r/e	–	r/e	–
Q6	479,388	392,762	1,048,576	r/e	–	r/e	–	r/e	–
Q7	11,273	988	161	r/e	–	t/o	–	r/e	–
Q8	3,383	2,706	21,881	135,165	21,881	21,997	21,881	r/e	–
Q9	2,135	828	252	r/e	–	266,626	252	r/e	–
Q15	6,251	733	242	r/e	–	18,047	242	r/e	–
Q15b	3,290	81	164	3,801	164	7,823	164	r/e	–
Q16	117,649	49,753	1,048,576	r/e	–	r/e	–	r/e	–
Q18	7,343	1,791	18,298	r/e	–	r/e	–	r/e	–
Q19	28,598	18,738	6,073	r/e	–	r/e	–	r/e	–

Table 6
Query execution times (msec) and number of results for queries without GRAPH annotations.

	Virtuoso			FedX		Semagrow		SPLENDID	
Query	cold	hot	#results	cold	#results	cold	#results	cold	#results
Q1	65,434	30,647	477,540	r/e	–	r/e	–	r/e	–
Q3	62,775	5,381	6,830	r/e	–	r/e	–	r/e	–
Q6	347,219	244,828	1,048,576	r/e	–	r/e	–	r/e	–
Q7	45,105	4,647	5,508	r/e	–	t/o	–	r/e	–
Q8	4,873	2,920	21,881	105,496	21,881	229,365	21,881	r/e	–
Q9	3,853	1,127	1,472	r/e	–	238,248	1,472	r/e	–
Q15	8,022	1,132	242	r/e	–	r/e	–	r/e	–
Q15b	5,070	245	164	2,956	164	7,506	164	r/e	–
Q16	57,276	43,658	1,048,576	r/e	–	r/e	–	r/e	–
Q18	6,468	2,770	36,596	r/e	–	r/e	–	r/e	–
Q19	174,747	19,879	6,155	r/e	–	r/e	–	r/e	–

for the SPLENDID case are due to source selection errors. In contrast, Semagrow takes graph annotations into consideration and manages to perform a more complete pruning procedure in the case of the queryset with graphs.

We have excluded in Table 7 the SPLENDID federator since it fails to produce any results for each one of our queries. Except from taking the graph annotations into consideration, the state-of-the-art should progress by developing more elaborate source selectors.

5.4. Query Planning

Another challenging factor of this benchmark is that at least half of the queries contain up to 16 statement patterns and only 3 queries contain less than 10. The set of all possible plans is exponential the total number of predicates. As a result, a dynamic-programming based decomposer (such as the decomposition component of Semagrow and SPLENDID) which perform an exhaustive search over the set of all possible execution plans, will need much time to provide a resulting plan. In Table 7 we illustrate the query planning time for each query for the Semagrow and FedX federators. Notice that in the case of Semagrow, exhaustive search results in a very high planning time and in many cases it even uses more than half of the total query evaluation time (eg. Q8, Q9, Q19 etc). In Q3, the query evaluation resulted in a timeout before even a plan was produced. FedX, on the other hand uses a greedy planner, and the high number of triple patterns of our queries does not affect the query planning time as much. How-

Table 7
Source Selection and Planning for FedX and Semagrow federations.

	with GRAPH annotations				without GRAPH annotations			
	Semagrow		FedX		Semagrow		FedX	
Query	planning time (in ms)	#sources	planning time (in ms)	#sources	planning time (in ms)	#sources	planning time (in ms)	#sources
Q1	7,061	1	676	9	13,537	9	r/e	–
Q3	811,000	3	r/e	–	747,428	10	r/e	–
Q6	33,549	4	r/e	–	33,185	8	r/e	–
Q7	t/o	–	r/e	–	t/o	–	r/e	–
Q8	6,406	3	428	8	24,132	8	2,511	8
Q9	259,729	3	r/e	–	220,535	9	r/e	–
Q15	15,014	3	r/e	–	11,596	10	r/e	–
Q15b	2,787	1	265	5	2,687	5	178	5
Q16	54,595	3	r/e	–	59,348	3	r/e	–
Q18	r/e	–	r/e	–	r/e	–	r/e	–
Q19	1,394,556	4	r/e	–	2,127,269	8	r/e	–

```
SELECT ?target ?compound
WHERE {
  { ?item rdf:type chembl:Activity .
    ?item chembl:hasMolecule ?compound .
    ?item chembl:pChembl ?pChembl .
    FILTER (?pChembl > 7) .
    ?item chembl:hasAssay ?assay_uri .
    ?assay_uri chembl:hasTarget ?target .
  } @ChEMBL
  .
  { ?target rdf:type ?target_type .
  } @Uniprot,ChEMBL,OPS-rc,DisGeNET,
     OPS-ims,WikiPathways,DrugBank,ChEBI
  .
  { ?target
      chembl:organismName "Homo sapiens"
  } @ChEMBL
  UNION
  { ?target
      chembl:organismName "Mus musculus"
  } @ChEMBL .
}
```

Listing 2: FedX execution plan for Q1

ever, a greedy planner may yield a not so efficient plan. For example, the query execution time (that is the time taken to execute the plan i.e. the total query evaluation time minus the query planning time) of Q8 is much lower in the case of Semagrow, even where the total query evaluation time is similar in both federation engines. In the case of SPLENDID, the planning phase resulted to runtime errors in queries Q1, Q7, Q8, in the queryset without graphs.

5.5. Query Execution

The query execution step is the final stage of a federated query evaluation, in which a query execution plan produced by the planner is executed. The remaining runtime errors that are not covered in the previous paragraphs (i.e. Q1 in Table 6 for the FedX case, Q3-Q7 and Q16-Q17 in both tables and Q1 and Q15 in Table 6 for Semagrow and also Q16, Q15, Q15b, Q18 and Q19 in Table 6 for SPLENDID) are due to errors in the query execution phase. Moreover, in all cases where the returned results are less than expected, an error on the execution phase has occurred, but the query execution component manages to recover from the error and continues with the evaluation of the query. In order to categorize those errors in the query execution phase, we can divide them in two families; firstly, an overflow or a similar error that occurs in the internal buffers of the federator itself (e.g. Q6 in Semagrow, or Q15, Q15b in SPLENDID); or secondly, a runtime error or an exception thrown by the source endpoint, which is a Virtuoso 40001 Error or a HTTP 404 Not Found Error, which is a known Virtuoso issue in the case of a heavy query load. Although the first family of errors seem to be of an engineering nature, the latter, though is a more complex issue. Since most federation

engines try to minimize query evaluation time, query executors tend to issue the queries in the source endpoints as quicker as possible, and this in bigger scale queries will definitely result in response failures. As a result, the state-of-the-art should progress by developing more reactive query execution strategies in order to prevent these errors.

5.6. Completeness of the Result Set

In order to draw further performance comparisons between two federators, we must make sure that both systems return the same results. However, we notice that no system managed to return all results in all queries of our benchmark. In the case of Virtuoso Store which contains all datasets (and that we included in our tables for reference reasons) the correct number of results was returned in all cases except Q6 and Q16. In these situations, a maximum of 1,048,576 results are returned. This happens because in Virtuoso the size of the maximum results per query is configurable but it cannot exceed a maximum of 1M results. SPLENDID couldn't evaluate any of our queries. FedX managed to evaluate only three of the queries, but since it ignores GRAPH annotations, it did not return the correct number of results in the version of the queryset with graph annotations. Semagrow manages to return the correct number of results in 5 queries from the queryset with graph annotations and in 3 queries from the queryset without graph annotations.

5.7. Query Evaluation Time

The most obvious and common way to compare two querying systems is the overall query evaluation time (with the fastest system being the best one), even though as we saw previously, this approach may lead to various problems (such as response failures). In general, we notice that for Virtuoso and Semagrow the queryset without graphs is evaluated faster than those that the graph annotations are contained. This difference in query execution time is due to the difference in the cardinality of the resultset and the difference in the number of the relevant sources/graphs. In FedX though, the query execution is similar in both cases since FedX ignores the graph annotations.

Also, as we noted in the previous subsection, a comparison between two systems makes sense only if they return the same number of results for the test situation. The only queries for which FedX, Semagrow and Virtuoso return the same results are Q8 and Q15b. Since the query execution plan of Q15b for all systems involves a small number of sources and results (the execution plans of the federators make use of almost only the DisGeNet source), we notice that all systems perform in a similar evaluation time. This is not the case for Q8 though, which uses more datasets, and as a result the execution times differ. Semagrow uses a more efficient plan than FedX, and as a result Semagrow is faster by an order of magnitude, but it is much slower that the standalone Virtuoso. The differences between a federation engine and a standalone can be huge (notice the difference between Virtuoso and Semagrow in Q9).

6. Related Benchmarks

FedBench [15] is a popular suite for benchmarking federated SPARQL query processing systems. It is comprised from three data collections, two of them using real datasets and focused on domain-specific queries and one collection that contains synthetic data. The first, named *Cross-Domain* collection refers to datasets of general interest and federate 6 datasets including DBpedia, Geonames and LinkedMDB; the second, called *Life-Science* considers queries that combine data from datasets from the drug discovery domain, such as ChEBI, Drugbank and KEGG. The queries proposed are considered to be typical scenarios for combining those datasets and are selected in such a way as to measure basic query characteristics of a federation engine, but are not produced from a realistic workflows that are typical for that domain. However, the complexity of queries is low using mainly inner joins of triple patterns.

BigRDFBench [13] extends FedBench by introducing additional large-scale real datasets to the federation and by proposing more complex queries that make use of various SPARQL operators. The benchmark splits the queries into two collections. The *Complex* collection which contains queries of increased complexity and the *Big Data* collection which contains queries that require processing of large intermediate results. The total federation consists of 13 datasets that contain in total one billion triples.

In the remaining section we will compare our benchmark with FedBench and BigRDFBench benchmarks. Table 8 draws a comparison between the datasets of the benchmarks, Table 10 compares them in terms of query characteristics while Table 9 collects the available SPARQL features. According to previous works

Table 8
Comparison of SPARQL features.

	FedBench	BigRDFBench	OPFBench
#datasets	10	13	10
#triples	164,816,689	1,003,960,176	1,875,100,172
#subjects	18,143,974	165,785,212	339,728,423
#predicates	1,659	2,160	719
#objects	54,129,688	326,209,517	555,724,456
#classes	431	459	469
struct.	0.616	0.658	0.682

Table 9
Comparison of SPARQL features.

	FedBench	BigRDFBench	OPFBench
BIND	–	–	1/11
COUNT	–	–	2/11
DISTINCT	–	9/32	1/11
FILTER	1/14	10/32	9/11
GROUP BY	–	–	2/11
LIMIT	–	4/32	–
OPTIONAL	1/14	8/32	12/11
ORDER BY	–	3/32	–
HAVING	–	–	1/11
REGEX	–	1/32	1/11
UNION	3/14	6/32	2/11
VALUES	–	–	3/11

Table 10
Comparison of benchmarks.

	FedBench	BigRDFBench	OPFBench
#patterns	4.3	6.7	11.9
#join vertices	2.6	3.4	6.2
star	49%	38%	32%
chain	32%	36%	31%
hybrid	5%	14%	24%
sink	14%	12%	13%
mean join vertex degree	2.1	2.6	2.8
#sources span	3.3	3.6	6.5
#results	907	59,754	233,162
mean triple pattern selectivity	0.057	0.102	0.042

[2,13], these characteristics are the most important ones in order to draw quantitative comparisons between benchmarks.

Regarding to the dataset statistics, we notice that FedBench contains 165M triples, BigRDFBench 1B triples, while our benchmark contains more than 1.8B triples. Despite the larger number of triples in our benchmark datasets, the total number of predicates is lower than FedBench. The number of classes are similar in all three dataset collections. We also notice that the average structuredness is slightly higher in our case. Even though Linked Data in general are less structured [15], it seems that the datasets used in the pharmaceutical domain are more structured and contain a relatively high number of triples and entities, however organized via a relatively small number of predicates.

All queries provided by the in all three benchmarks are SELECT queries, typically conjunctively joining multiple triple patterns. All three benchmarks also use the UNION operator in about 1/5 of the queries. FedBench queries are more simple with respect to the SPARQL operators that are featured. Except from the queries that use unions, there exists only one query that uses a filtering operation and a left join operation. On the contrary, BigRDFBench and OPFBench feature more SPARQL operators. There are differences between the operators offered in each query-set. OPFBench uses COUNT, GROUP BY, HAVING, BIND and VALUES operators while this is not the case of BigRDFBench. In contrast, LIMIT and ORDER BY operators are used by BigRDFBench and not by OPFBench. From the common operators, BigRDFBench queries make less use of the DISTINCT operator, while OPFBench make much use of the filtering operation (more than 80% of the queries use this operator).

In Table 10 we present a comparison of the query characteristics. We present the average of each characteristic shown in Section 3 (Table 3 and Table 4) per query for OPFBench, FedBench and BigRDFBench. We notice that our queries feature more triple patterns, join vertices (almost twice the size). Also, our query-set focus more in hybrid vertices, and the average resultset cardinality is greater by more than one order of magnitude from the other two benchmarks.

The *sources span* of a query is the number of data sources that contain at least one triple that matches any

of the query's patterns.[3] Sources span indicates the effect that non-trivial query selection can have on eliminating sources to be considered: A higher span will make more visible the effect of sophisticated source selection that eliminates sources that superficially appear to have relevant data. The #sources span column in Table 4 gives the sources span for each query in OPFBench. As can been seen in Table 10, OPFBench queries have a considerably higher average span than FedBench and BigRDFBench.

FedBench and BigRDFBench are using real-world datasets but are using an artificial queryset, while OPFBench uses not only real-world datasets but also a realistic queryset. Furthermore, the size of the OPFBench datasets is larger, and the queries are more difficult in terms of their syntactic characteristics, resultset cardinality, and SPARQL operators.

7. Conclusion

In this article we present *OPFBench*, a new SPARQL query processing benchmark that comprises data from the pharmacology domain and a queryset that is derived from workflows established by the pharmacology community. The natural segmentation of the data into ten datasets, each containing information about different kinds of entities or from a different perspective or linksets between the former, OPFBench is particularly suited for benchmarking federated querying of heterogeneous data.

Although pharmacology, and the life sciences in general, have been used in prominent SPARQL benchmarks before (such as FedBench and BigRDFBench), synthesised querysets were used. OPFBench is the first benchmark where the query set is challenging and based on actual practice, giving us a more realistic perspective of the features and capabilities that matter the most for data analysis practitioners; and especially in the life sciences, a challenging domain for data management and analysis technologies.

To understand what insights can be gained by this new benchmark and how these can drive research in federated query processing, we have analysed OPFBench both statically and also by using it to test the current state of the art in federated query processing. We have made several interesting observations that point to future development in federated query processing. From the static analysis, for example, we see that the ORDER BY/LIMIT construction is not used in the Open PHACTS workflows which prefer receiving all matching results. On the other hand, the GROUP BY/COUNT construction is used to compute aggregates in two OPFBench queries (Q15 and Q19), and performance on this construction is not tested by either FedBench or BigRDFBench (Table 9). The relevance of this finding for future development is corroborated by the experimental analysis, where all three federated query processors failed on Q19 and FedX and SPLENDID failed on Q15, demonstrating the challenge presented by the need to aggregate voluminous intermediate results even if the final resultset is relatively small (Tables 5 and 6).

Through these examples, we show that the conclusions drawn by benchmarking on OPFBench are both technically interesting and not previously observed by benchmarking on FedBench and BigRDFBench. Naturally, one cannot argue that life sciences benchmarks (a challenging data analysis domain as it may be) suffice to prioritize research plans or to compare systems as fit for applications in any domain. What we do argue for, is that using real workflows has shown that attention on robustly computing the GROUP BY operator should take priority over ORDER BY, at least for pharmacology applications.

To be able safely generalize about the impact of research on SPARQL query optimization and execution, we need a battery of benchmarks derived from different domains and applications. Especially those domains where, like pharmacology, the need for complex, non-trivial queries exists and where the culture and tools are in place to have domain experts define challenging queries as a matter of their daily work, without having these queries manually optimized by database experts. And we also need the mechanisms that allow us to easily and reproducibly execute such complex experiments, involving multiple engines and multiple benchmark suites. The KOBE Benchmarking Engine, the second contribution of the work described here, moves towards this direction by generalizing a benchmark executor into an open source, generic, configurable tool for deploying SPARQL endpoint federations and applying a query load to them.

Our future work on OPFBench and KOBE will move in two directions. Firstly, we will follow closely ongoing research by the Open PHACTS Foundation on using data on patent and pathway interactions to answer the questions that were not addressed by the currently released workflows. Besides the general bene-

[3] Excluding from the coubt sources that contribute to the result for only one triple pattern.

fits to the benchmark from the increased size of the data and the queryset, we also expect that this will address joining data from a large number of different endpoints. The current queryset needs to involve at most four datasets out of the ten datasets in the suit, and the queries on patent and pathway interactions are expected to involve more datasets.

The second direction of future work on OPFBench and KOBE will allow testing engines on realistic *query loads* instead of sets of isolated queries. This kind of benchmarking is even more specifically tied to individual applications and less indicative of overall quality, but there are several features than can only be tested by query loads and not query sets, such as caching and prefetching. Again, ideally we will need to collect a varied collection of query loads to be able to safely generalize. And, based on the experiences and conclusions above, we advocate that if we can collect such a varied collection of *real* query loads, we will phenomena and choke points that will not be present in synthetic query loads, since the latter only test what we already suspect to be an issue.

Acknowledgements

The work described here has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644564. For more details see the BigDataEurope Website at `https://www.big-data-europe.eu`

We wish to acknowledge the Open PHACTS Foundation, the charitable organisation responsible for the Open PHACTS Discovery Platform, without which this work would not have been possible.

Appendix

A. The KOBE Benchmarking Engine

The *KOBE Benchmarking Engine (KOBE)* automates the process of deploying the SPARQL endpoints that made up a federation, applying the query load, and collecting the results. KOBE is an open-source software that can be configured to different datasets, query loads, querying scenarios, and federated query processing systems.[4]

KOBE is organized around datasets and the SPARQL endpoints that expose them, federations of such endpoints, and experiments. In order to provide an easy way to redistribute in multiple platforms we have packaged all these KOBE components as Docker images [11]. This enables us to provide a highly configurable benchmarking environment where different datasets and federation engines can be easily deployed and different benchmark query loads and scenarios can be easily applied to any deployment.

A.1. Datasource endpoints and federation engines

Data can be served from any public or local SPARQL endpoint, but KOBE also includes Docker images for all datasets in the OPFBench, FedBench, and BigRDFBench suites. These images have Debian 8.5 and Virtuoso 7 pre-loaded and each image also executes the commands needed to download from a public location one dataset dump, carry out any necessary pre-processing (e.g., convert from RDF/XML to N-TRIPLES), and bulk-load the dump into its Virtuoso instance.

The federation engines are also deployed in separate Docker containers. Each federation engine has its own configuration mechanism for declaring the data sources that it federates, and KOBE provides Docker images for each of Semagrow, FedX and SPLENDID pre-configured to federate the data sources needed for OPFBench and FedBench. It should be noted that Semagrow and SPLENDID also need dataset statistics as part of their configuration. These statistics are provided for the OPFBench and FedBench datasets, but can also be easily prepared for any new dataset using our scraper that computes it from RDF dumps.[5] Using the current images as templates and the scraper to prepare the configuartion files, it is straightforward for the experimenter to prepare new Docker images for Semagrow and SPLENDID federations of any mixture of datasets.

A.2. Workload Generation

After defining the federation that will be benchmarked, the next step is to configure the our benchmarking engine to generate the query load for an experiment. An experiment can be configured with the use of simple configuration files for benchmark settings (query sets, number of runs per query, execu-

[4] See `https://github.com/semagrow/kobe`

[5] See `https://github.com/semagrow/sevod-scraper`

tion timeout). The configuration files for executing the OPFBench and FedBench suites are included in the current KOBE distribution.

The driver can connect to the specific federation engine via the SPARQL protocol, and all the federation engines can access the data sources via the SPARQL protocol as well. At each step of the experiment, all queries from the given queryset are executed one time and in subsequent runs, and then this step is repeated a desired number of times. This process allows us to distinguish between the performance of cold runs and hot runs, and therefore to exclude (if wanted) the effect of cold starts in our measurements. This distinction may be useful since in many situations the execution time of the first stage of the experiment is much larger than the following stages, usually due to cashing and metadata loading.

The output of the experiment is written on a CSV file which contains information about each query. An example from the output file is the following:

```
Query;run1;run2;run3;avg;numResults;minRes;maxRes;
SQ1;793;148;117;352;1159;1159;1159;
SQ2;707;472;383;520;333;333;333;
```

In this example, we have executed a workload which executes SQ2 after SQ1 six times. For each query, we display the query execution time for each run, the average execution time for all runs, and the minimum, the maximum and the average number of results that were returned by the federation engine. In the case that a federation engine failed to process the request, or some other error occurred, a negative number is inserted in the corresponding line of the CSV file. The error itself is displayed in the log file.

A.3. Setting up an experiment

Setting up and executing an experiment is a complex process that requires a number of datasource endpoints, a number of federations and a process where a driver connects to a federation and executes a given queryset. However, since all components are deployed using Docker containers, in order to deploy a benchmarking environment, one has to create an appropriate docker-compose.yml file. A minimal example could be the following:

```
version: '2'
services:
 datasource:
   image: semagrow/virtuoso:bench-7.1
   container_name: datasource
   volumes_from:
    - datasource-data
   environment:
    - DOWNLOAD_URL=https://path/to/download/dump.tar
 datasource-data:
   image: busybox
   volumes:
    - /data

 semagrow:
   image: semagrow/semagrow
   container_name: semagrow
   volumes:
    - /path/to/semagrow/conf:/etc/default/semagrow

 eval:
   image: semagrow/kobe-evaluator
   container_name: eval
   volumes:
    - /path/to/query/set:/etc/querySet
   environment:
    - ENDPOINT=http://semagrow:8080/SemaGrow/sparql
    - TIMEOUT=3600000
```

This is a minimal federation that contains of a Semagrow federation that consists of only a single datasource. The datasource is configured with a URL from where the data are going to be downloaded. Semagrow federator is configured by a configuration file which is placed in /path/to/semagrow/conf directory of the host machine. The minimal configuration for Semagrow for the specific scenario is the following metadata.ttl file:

```
@prefix void:<http://rdfs.org/ns/void#> .
@prefix rdf:<http://www.w3.org/1999/02/22-rdf-syntax-ns#>.

_:DatasetRoot rdf:type void:Dataset .

_:Dataset1 rdf:type void:Dataset ;
  void:subset _:DatasetRoot ;
  void:sparqlEndpoint <http://datasource:8890/sparql>.
```

Finally, the evaluator engine is configured to issue in Semagrow all queries contained in /path/to/query/set directory of the host machine once with a timeout of 3600000 milliseconds. Each query must be contained in its separate file. Finally, in order to run the experiments, the experiment should evoke the appropriate docker-compose commands.

Since this process of the setup can be tedious for the experimenter, apart from the configurations of known benchmark scenarios, we have implemented a tool that aims to provide a quick and easy way to set up the

needed configuration for the specific benchmark experiment. By following the provided step by step procedure one can create a docker-compose.yml file that is used to deploy a benchmark using Docker containers.

References

[1] Alexander, K., Cyganiak, R., Hausenblas, M., Zhao, J.: Describing linked datasets. In: Bizer, C., Heath, T., Berners-Lee, T., Idehen, K. (eds.) Proceedings of the WWW2009 Workshop on Linked Data on the Web, LDOW 2009, Madrid, Spain, April 20, 2009. CEUR Workshop Proceedings, vol. 538. CEUR-WS.org (2009), http://ceur-ws.org/Vol-538/ldow2009_paper20.pdf

[2] Aluç, G., Hartig, O., Özsu, M.T., Daudjee, K.: Diversified stress testing of RDF data management systems. In: Mika, P., Tudorache, T., Bernstein, A., Welty, C., Knoblock, C.A., Vrandecic, D., Groth, P.T., Noy, N.F., Janowicz, K., Goble, C.A. (eds.) The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Riva del Garda, Italy, October 19-23, 2014. Proceedings, Part I. Lecture Notes in Computer Science, vol. 8796, pp. 197–212. Springer (2014), http://dx.doi.org/10.1007/978-3-319-11964-9_13

[3] Azzaoui, K., Jacoby, E., Senger, S., Cuadrado Rodríguez, E., Loza, M., Zdrazil, B., Pinto, M., Williams, A.J., de la Torre, V., Mestres, J., Pastor, M., Taboureau, O., Rarey, M., Chichester, C., Pettifer, S., Blomberg, N., Harland, L., Williams-Jones, B., Ecker, G.F.: Scientific competency questions as the basis for semantically enriched open pharmacological space development. Drug Discovery Today 18(17–18), 843–852 (2013), http://www.sciencedirect.com/science/article/pii/S1359644613001542

[4] Charalambidis, A., Troumpoukis, A., Konstantopoulos, S.: SemaGrow: Optimizing federated SPARQL queries. In: Proceedings of the 11th International Conference on Semantic Systems (SEMANTiCS 2015), Vienna, Austria, 16–17 September 2015 (2015)

[5] Chichester, C., Digles, D., Siebes, R., Loizou, A., Groth, P., Harland, L.: Drug discovery FAQs: Workflows for answering multidomain drug discovery questions. Drug Discovery Today 20(4), 399–405 (2015), http://www.sciencedirect.com/science/article/pii/S1359644614004437

[6] Duan, S., Kementsietsidis, A., Srinivas, K., Udrea, O.: Apples and oranges: a comparison of RDF benchmarks and real RDF datasets. In: Sellis, T.K., Miller, R.J., Kementsietsidis, A., Velegrakis, Y. (eds.) Proceedings of the ACM SIGMOD International Conference on Management of Data, SIGMOD 2011, Athens, Greece, June 12-16, 2011. pp. 145–156. ACM (2011), http://doi.acm.org/10.1145/1989323.1989340

[7] Görlitz, O., Staab, S.: SPLENDID: SPARQL endpoint federation exploiting VOID descriptions. In: Proceedings of the 2nd International Workshop on Consuming Linked Data (COLD 2011), Bonn, Germany, October 23, 2011. CEUR Workshop Proceedings, vol. 782 (2011)

[8] Groth, P., Loizou, A., Gray, A.J., Goble, C., Harland, L., Pettifer, S.: API-centric linked data integration: The Open PHACTS Discovery Platform case study. Web Semantics: Science, Services and Agents on the World Wide Web 29, 12–18 (2014), http://www.sciencedirect.com/science/article/pii/S1570826814000195

[9] Guo, Y., Pan, Z., Heflin, J.: LUBM: A benchmark for OWL knowledge base systems. Journal of Web Semantics 3(2-3), 158–182 (2005), http://dx.doi.org/10.1016/j.websem.2005.06.005

[10] Loizou, A., Angles, R., Groth, P.: On the formulation of performant SPARQL queries. Web Semantics: Science, Services and Agents on the World Wide Web 31, 1–26 (Mar 2015)

[11] Merkel, D.: Docker: Lightweight Linux containers for consistent development and deployment. Linux Journal 2014(239) (2014), http://dl.acm.org/citation.cfm?id=2600239.2600241

[12] Ozkan, E.C., Saleem, M., Dogdu, E., Ngonga Ngomo, A.C.: UPSP: Unique predicate-based source selection for SPARQL endpoint federation. In: Demidova, E., Dietze, S., Szymanski, J., Breslin, J.G. (eds.) Proceedings of the 3rd International Workshop on Dataset Profiling and Federated Search for Linked Data (PROFILES 2016) co-located with the 13th ESWC 2016 Conference, Anissaras, Greece, May 30, 2016. CEUR Workshop Proceedings, vol. 1597. CEUR-WS.org (2016), http://ceur-ws.org/Vol-1597/PROFILES2016_paper4.pdf

[13] Saleem, M., Hasnain, A., Ngonga Ngomo, A.C.: BigRDFBench: A billion triples benchmark for SPARQL endpoint federation http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.667.3600

[14] Saleem, M., Khan, Y., Hasnain, A., Ermilov, I., Ngomo, A.N.: A fine-grained evaluation of SPARQL endpoint federation systems. Semantic Web 7(5), 493–518 (2016), http://dx.doi.org/10.3233/SW-150186

[15] Schmidt, M., Görlitz, O., Haase, P., Ladwig, G., Schwarte, A., Tran, T.: FedBench: A benchmark suite for federated semantic data query processing. In: Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.) Proceedings of the 10th International Semantic Web Conference (ISWC 2011), Bonn, Germany, October 23-27, 2011, Part I. pp. 585–600. Springer, Berlin/Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-25073-6_37

[16] Schwarte, A., Haase, P., Hose, K., Schenkel, R., Schmidt, M.: FedX: A federation layer for distributed query processing on Linked Open Data. In: Proceedings of the 8th Extended Semantic Web Conference (ESWC 2011), Heraklion, Crete, Greece, May 29 – June 2, 2011. Lecture Notes in Computer Science, vol. 6644, pp. 481–486. Springer (2011)

[17] Williams, A.J., Harland, L., Groth, P., Pettifer, S., Chichester, C., Willighagen, E.L., Evelo, C.T., Blomberg, N., Ecker, G., Goble, C., Mons, B.: Open PHACTS: Semantic interoperability for drug discovery. Drug Discovery Today 17(21–22), 1188–98 (2012), http://dx.doi.org/10.1016/j.drudis.2012.05.016