Scaling Out Federated Queries for Life Sciences Data in Production

Dieter De Witte[1], Laurens De Vocht[1],
Kenny Knecht[2], Filip Pattyn[2], Hans Constandt[2],
Erik Mannens[1], and Ruben Verborgh[1]

[1] iMinds - IDLab - Ghent University, Ghent, Belgium
`{firstname.lastname}@ugent.be`
`http://datasciencelab.ugent.be/`
[2] Ontoforce, Ghent, Belgium
`{firstname.lastname}@ontoforce.com`
`http://www.ontoforce.com`

Abstract. There exists an abundance of Linked Data storage solutions, but only few meet the requirements of a production environment with interlinked life sciences data. In such environments, a triple store has to support complex SPARQL queries and handle large datasets with hundreds of millions of triples. The Ontoforce platform DISQOVER offers federated search for life sciences, relying on complex federated queries over open life science data. The queries correspond to user actions in its exploratory search interface. Different state-of-the-art approaches for scaling out are compared, both in terms of their ability to execute the queries as in terms of performance. This paper analyzes and discusses the features of the datasets and query mixes. An in-depth analysis is provided showing the features of the most challenging queries.

Keywords: Evaluation, Life Sciences, Big RDF Data, Semantic Web Tools, Distributed Querying

1 Introduction

Life Sciences is one of the successful application domains of semantic technology. The Life Sciences domain is interdisciplinary, which makes interlinking data sources interesting and crucial. The Linked Open Data Cloud contains many RDF data sources related to life sciences, but this comes with a set of challenges: (i) the union of all datasets qualifies as Big Data and therefore puts a strain on the available technologies for querying and (ii) these insights contain information of multiple datasets at once, making the queries federated in nature. These challenges are being addressed by:

- *Vertical scaling* is using an expensive high-end machine with a lot of RAM and CPU power. No additional software development is required in this case.
- A *Compression algorithm* such as HDT [7] can easily compress RDF datasets by a factor of 10–20. This allows for much larger datasets to be handled by a single server.

An alternative is to opt for a distributed architecture:

- *Horizontal scaling* uses multiple - often cheap, low-end - instances in a distributed system. Most enterprise RDF stores support parallelization, but this can imply both a high availability solution (data replication), or a sharded system (data partitions) that can deal with increasingly large datasets.
- *Query federation* [15]: All datasets are hosted by their providers and a federated query engine redirects the relevant parts of each query to the right endpoint and finally combines all the recieved information to solve the query.
- Native *Big Data approaches* typically map SPARQL queries to SQL technologies available in the Hadoop stack: SparkSQL [5] or Impala [13].

1.1 Related Work

Specifically in the Life Sciences domain BioBenchmark Toyama 2012 [16] sheds light on the capabilities of typical single-node RDF storage solutions. In this work 5–10 queries are evaluated against 5 real datasets ranging from 10 million to 8 billion triples. The novelty in our work lies in the fact that we are dealing with a large set of 1,223 federated queries used in a production environment.

Complementary to this work we evaluated 4 RDF databases [6] on a new artificial benchmark named WatDiv, which guarantees diversity both in terms of query properties and dataset properties [1]. Other artificial benchmarks extensively used in the past are the Lehigh Universit Benchmark [9], the Berlin SPARQL Benchmark [3], DBPBM[11], and the SP^{2}Bench [14]. Specifically for NoSQL approaches to RDF Mauroux [4] evaluated the performance for RDF data workloads.

The results of this complementary work motivate the evaluation on a real-world dataset. One big difference with this benchmark as opposed to WatDiv is the federated nature of the data and the queries, the explicit focus on scalability and the fact that the current queryset is rich in SPARQL features, while WatDiv focuses on pure Basic Graph Patterns (BGPs).

The difficulty in selecting and optimizing RDF systems is also being addressed in two European H2020 projects: LDBC [2] and HOBBIT [12]. These projects aim to create a platform to offer industry a unified approach for running benchmarks related to their actual workloads.

1.2 Our Contribution

This paper demonstrates a methodology to evaluate RDF storage solutions on a data and query set of choice. The goal of this work is to evaluate the ability of today's triple stores in terms of scalability with big biomedical data sources and complex real-world queries. The pitfalls in the interpretation of the results are highlighted and suggestions are formulated to circumvent them and draw the right conclusions. Finally, a post-processing approach focusing on re-usability and automation is developed.

2 Benchmark dataset and queries

2.1 The DISQOVER platform

Ontoforce has designed a semantic search platform DISQOVER which integrates over 110 Life Sciences data sources. Examples of these data are PubMed, ClinicalTrials.gov, NCBI Gene, National Drug Code, MedDRA, DrugBank, MeSH, etc. DisQover comes with an interactive UI which allows the exploration of this huge dataset without sacrificing low query latency. The combination of query federation and interactivity is achieved by combining the use of triple store with an indexing system in which different RDF datasets are combined into well chosen aggregates by making use of an ETL preprocessing pipeline. To keep up with the growing set of data sources in their product Ontoforce requires RDF engines which are (i) fully SPARQL 1.1 compliant, (ii) are sufficiently mature to operate in a production environment and (iii) allow their offering to further scale out, thus requiring RDF solutions which support compression or horizontal scalability.

2.2 Ontoforce Data Analysis

Table 1 contains global statistics about the benchmark dataset. The dataset is only a fraction of the complete dataset used in DISQOVER which contains over 8 billion triples. It contains the graphs which are most relevant to the queries in the operation mix. As shown in our prior work with WatDiv [6], datasets exceeding 1 billion triples can pose serious challenges, to even state-of-the-art enterprise RDF databases given modest hardware.

For every graph an additional graph is present containing the inferred triples. The five largest graphs make up approximately 80% of the dataset size. These graphs correspond to PubMed, ChEMBL, NCBI-Gene, DisGeNET and EPO, with the PubMed already making up 60% of the data. Since the dataset is under nondisclosure we provide some additional statistics in Fig. 1 to shed some light on the data and help researchers create artificial benchmarks with similar dataset properties.

Property	Value
Number of distinct subjects	0.137B
Number of distinct predicates	1,782
Number of distinct objects	0.287B
Number of distinct classes	2,434

Property	Value
Number of triples	2.4B
Number of graphs	107
Compressed (nq.gz) dataset	25GB

Table 1. The combined statistics show the high demands of the datasets used by the DISQOVER platform.

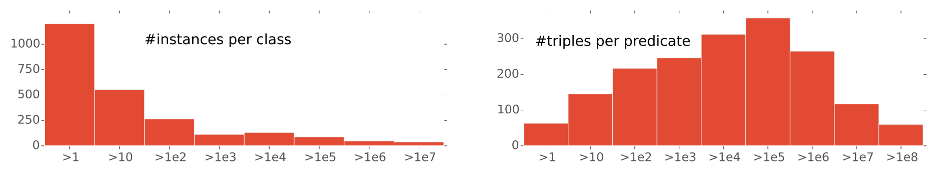

Fig. 1. The number of instances per class (left) follows a long tail distribution: the bulk of 2,000 classes has 1–100 instances, while the long tail contains 50 classes with 500K instances each. The median of the predicate distribution (right) occurs at 5,000 triples, while the outliers reveal that 50 predicates have over 10M triples each.

2.3 Ontoforce Query analysis

The benchmark query mix consists of 1,223 queries which are both complex and diverse in term of SPARQL features. The queries are automatically generated by the UI in the context of faceted browsing [8] making aggregation and filter operations very common. The actual query formulation is not optimized towards performance [10].

To provide the reader with some insights in the queries we used SPARQL.js parser[3], which converts SPARQL queries into JSON objects. This JSON structure is then used to generate a feature vector per query. We distinguish between features related to the complexity of the query structure and features which correspond to SPARQL keyword counts. In Figure 2, a series of features and their occurrence distribution is shown which shed further light on the complexity of the SPARQL patterns.

1. properties of the JSON tree representation such as the number of levels (depth), the number of nodes (keys) and the length of the file (jsonLines);
2. properties of the query graph structure such as the amount of queries, BGPs and the total amount of triple patterns;
3. the type of triple patterns.

The large amount of queries with over 10 triple patterns is noteworthy, while the Watdiv query templates[4] contain 3 up to 10 patterns maximally. The prevalence of unbound triples reveals how the DISQOVER queries are built: starting from general queries where additional selectivity is introduced by ad hoc introduction of `FILTER` statements. Most of these `FILTER ?x = <...>` queries can be manually removed by replacing the corresponding `?x` variable in the triple patterns. Other `FILTER` operations contain an `IN` operator followed by a long series of possible values for a variable, which could be rewritten as complex unions. Half of the queries are `COUNT DISTINCT` queries, furthermore most keywords are present, their effect on query runtimes will be studied in Fig. 5.

[3] `https://www.npmjs.com/package/sparqljs`
[4] `http://dsg.uwaterloo.ca/watdiv/basic-testing.shtml`

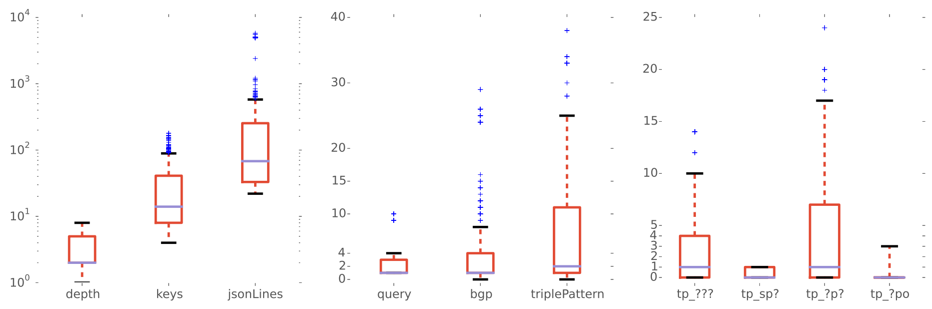

Fig. 2. The range of depths (left) indicate many queries are nested, jsonLines shows that some queries are very long (mostly caused by FILTER IN). 25% of the queries contain more than 10 triple patterns (center), but most of these are completely unbound (`???`), or only have a predicate (`?p?`) (right).

3 Methodology

The selected RDF storage solutions should be capable of serving in a production environment with Life Sciences data. Important criteria here are the maturity of the solutions and the ability to handle Big Data while offering full SPARQL 1.1 compliance. The following setups were used in the benchmarks, we also introduce a shorthand notation for the different benchmark runs:

– *Single-node references*: Virtuoso 7.2.41 (**V1**) and Blazegraph 2.0.0 (**Bla1**) single-node setups.
– *Vertical Scalability*: is analyzed by scaling down the reference Virtuoso node to a 32GB machine (**V1_32**).
– *Compression*: Jena Fuseki is used as a SPARQL endpoint on top of compressed HDT files - one per graph (**Fu1**).
– *Query federation*: **V1** was used for individual endpoints and a separate instance ran FluidOps FedX 3.2 with a Virtuoso Adapter. Two setups were tested: one with a single endpoint to measure the overhead of the federation software (**Fl1**) and one with 3 endpoints (**Fl3**).
– *Horizontal scaling*: Virtuoso's enterprise offering includes support for a sharded cluster, which we tested in a 3 node setup (**V3**).

The hardware for the benchmarks was selected out of the AWS on-demand instance offer as follows[5]:

– `r3.2xlarge` (8 vCPU, 61 GB RAM) for the triple stores and for FedX.
– `c3.2xlarge` (8 vCPU, 15 GB RAM) to run the benchmark client.
– `r3.xlarge` (4 vCPU, 30 GB RAM) for the scaled down **V_32**.

Multiple independent simulations were run to improve the confidence in our results for some set-ups, in the notation these are distinguished by using a simulation id, for example **V3_0**.

[5] `https://aws.amazon.com/ec2/instance-types/`

A PAGO (Pay-as-you go) license was used for Virtuoso[6]. Blazegraph[7] was installed manually with the quad index[8] enabled. All stores were configured to make use of the available memory and CPU. Fuseki was configured to keep half of the HDT graphs in memory, which was the maximum feasible given the system memory. SPARQL Query Benchmarker[9] was configured to run 1 single-threaded warmup run and one multithreaded run with 5 concurrent threads. Each thread runs an operation mix with all queries in a randomized order, the query timeout was set to 20 minutes.

4 Results

The dataset and query properties indicate that the DISQOVER dataset will challenge existing RDF databases. In the results section (i) we investigate the ability of the different solutions to ingest the big dataset and to survive a multi-threaded stress test with complex queries; by diving into the logs (ii) we show which errors occur and if the queries are solved correctly; and finally (iii) we analyse the features of the most time consuming queries.

4.1 Database Survival during Ingest- and Stress-Testing

A benchmark consists of a data ingest phase and a query execution phase. For **V1** 3 concurrent bulk loaders were used which finished after 4h11m. Scaling down to **V1_32** also impacted the bulk loader which now required 11h15m. In the clustered setup **V3**, 9 bulk loaders were run and every graph was stored only on one of the nodes, this process finished in 4h35m. The **Flu** simulations used Virtuoso as SPARQL endpoint with the data manually distributed, with a single node hosting the PubMed dataset (60%) and the other two nodes each containing approximately 20% of the data. **Bla1** took 7d12h38m to complete the ingestion. For **Fu1** the data needs to be compressed to an HDT file per graph. A high memory machine (256GB RAM) was used to complete this task which was only possible when converting the N-Quads to Turtle format before running the HDT in memory algorithm which took 15h23m to complete. The time to build all index files required by each HDT file, about 1h, must be added to the total for Fuseki. The benchmarker software only generates runtime information upon completion of a run. Therefore the benchmarker logs were analysed to find the last successful query per query thread for every simulation, the results of which can be seen in Fig. 3.

Some complications arose in between the loading and the benchmark phase with some of the Virtuoso runs. For **V1_32** the store went offline on 3 separate occasions after initiating the benchmark client. In the third occasion we

[6] `https://aws.amazon.com/marketplace/pp/B011VMCZ8K/ref=srh_res_product_title?ie=UTF8&sr=0-5&qid=1455494788712`

[7] `https://www.blazegraph.com/product/`

[8] `https://wiki.blazegraph.com/wiki/index.php/Configuring_Blazegraph#Quads_Mode`

[9] `http://sourceforge.net/p/sparql-query-bm/wiki/Introduction/`

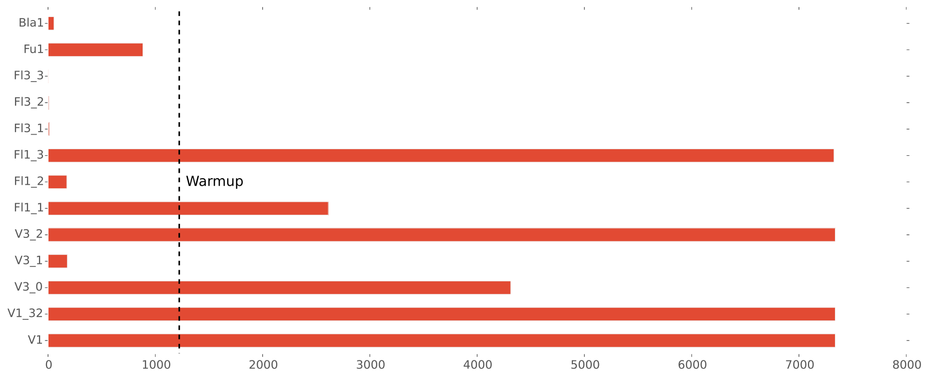

Fig. 3. Only the Virtuoso instances managed to succesfully complete the benchmark ($6 \times 1,223$ queries), with the exception of 2 V3 simulations. Fu1 crashed after having completed most of the warmup run ($1 \times 1,223$), Bla1 was shut down after a week while still in the warmup run. Fl3 could hardly complete any queries.

waited for the transaction logs to be fully replayed before starting the benchmark. The PAGO instance has a limit of 10 user sessions which must be used for both querying and bulk loading. For this reason the Virtuoso simulations were restarted after the ingest phase with the HTTP threads increased to 10 (except for **V3_2**, 1 bulkloader). This restart did not immediately succeed.

4.2 Error Frequency and Types

The query event stream generated from the benchmarker logs contains information about the type of errors, the number of results and the query runtime. Upon comparing the number of results for every query we discovered that this can be very different between simulations. Careful analysis showed that the benchmarker results file is incorrect in case a query is not always successful. Query failures – with 0 results – for example modify the average number of results, while omitting these cases is more intuitive. The query event stream allowed us to filter out the successful queries per thread and compare the number of results per query. The latter was always identical between the threads of a single database but in between systems the results can differ, an important observation which is often overlooked by focusing exclusively on the runtimes.

In Fig. 4 we show both the error frequency and the correctness. Correctness is defined as having the maximal number of results compared to other simulations. We used **V1** in pairwise comparisons and with the exception of 2 queries (**Flu1**) the results are always correct. DISQOVER results can be obtained by using both a triple store or an indexing system as the backend, the results are consistent in case of **V1** and the alternate system. For the two incorrect results **V1** returned 1,048,576 which seems to be an upper boundary, while **V1** was explicitly configured not to limit the results. **Fu1** returns more results for the

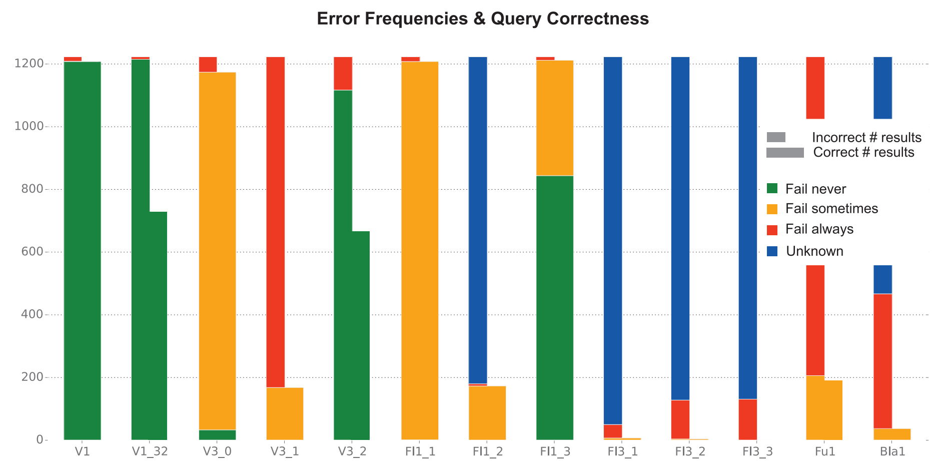

Fig. 4. Error frequency per query (color) and query correctness (bar width) for the different benchmark runs. **V1_32** and **V3_2** seem to be very successful at first sight but for half of the successful(!) queries it is shown that the number of results is incorrect (0). **V3_0** is thus the most reliable V3 simulation.

queries targeting a specific graph but upon closer inspection this reveals an error in the HDT graph implementation. Incorrectly, these queries query the default union graph.

4.3 In-depth Analysis based on Query Features

In this section we study the properties of the queries with respect to errors and runtimes. Three important observations can be made:

- The problematic queries for **V1** and **V3** are in general queries which are complicated along all query features, i.e. have a high frequency of the different SPARQL keywords. The feature values are in general 1 to 2 standard deviations above the average query.
- The queries successfully solved by **Fu1** and **Bla1** are queries which are less complicated than the average query, they correspond to BGP queries with hardly any SPARQL operators.
- A runtime comparison cannot be considered reliable. For example for the **V1_32** simulation 80% of the correct queries are `COUNT DISTINCT` queries for which we cannot verify whether the actual counts are correct since they were not logged by SPARQL query benchmarker. (only the number of results per query are counted)

In Fig. 5 we sorted all queries by descending execution time, a second axis shows the actual runtimes which drop from 15 minutes 10 seconds in the first 100 queries. The runtime of a single multi-threaded query mix is 4h04m. As

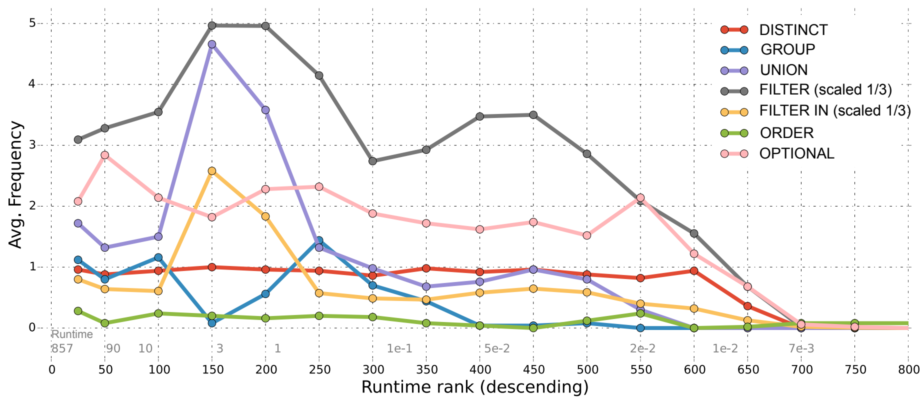

Fig. 5. Occurrence of SPARQL keywords per query (averaged). The extremum for query 100 to 150 is indicates that queries with a lot of FILTERs and UNIONs are among the most challenging ones, yet the left hand side contains queries with a higher number of OPTIONAL and GROUP keywords and the resultset sorted.

mentioned in section 3, this plot gives an idea about the occurrence frequency of certain SPARQL keywords.

We took slices of 25 or 50 queries and calculated for each the average feature vector the values of which are plotted. This immediately shows that the `COUNT DISTINCT` queries are the most challenging. The frequency of `FILTER` operators is a bad indicator for complexity, **V1**'s query optimizer most likely eliminates most of these, `FILTER IN` is a better indicator. The combination of `OPTIONAL`, `GROUP` and `ORDER` poses the biggest challenge. Note that query features alone cannot fully predict runtime complexity, the graph structure of the data also plays an important role.

5 Conclusions and Future Work

Complex SPARQL queries in combination with a big RDF dataset are a real challenge for most RDF solutions. In depth analysis of the query results leads to the recommendation of adding more diagnostics to ascertain proper operation of RDF stores, an upgrade and extension of the SPARQL benchmarker to better deal with errors and counts is desirable. Scientific claims about the runtimes of the different engines are premature but current evaluation does show that sharded parallelisation is the most promising for truly big RDF.

6 Acknowledgments

The research activities were funded by VLAIO (the Agency for Innovation and Entrepreneurship in Flanders) in an R&D project with Ontoforce, Ghent University and imec (iMinds). iLab.t provided the required high memory infrastructure.

References

1. G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee. Diversified stress testing of RDF data management systems. In *The Semantic Web–ISWC 2014*, pages 197–212. Springer, 2014.
2. R. Angles, P. Boncz, J. Larriba-Pey, I. Fundulaki, T. Neumann, O. Erling, P. Neubauer, et al. The linked data benchmark council: A graph and rdf industry benchmarking effort. *SIGMOD Rec.*, 43(1):27–31, May 2014.
3. C. Bizer and A. Schultz. The Berlin SPARQL Benchmark. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(2):1–24, 2009.
4. P. Cudré-Mauroux, I. Enchev, S. Fundatureanu, P. Groth, A. Haque, A. Harth, Keppmann, et al. NoSQL databases for RDF: an empirical evaluation. In *The Semantic Web–ISWC 2013*, pages 310–325. Springer, 2013.
5. O. Curé, H. Naacke, M. A. Baazizi, and B. Amann. On the evaluation of rdf distribution algorithms implemented over apache spark. 2015.
6. D. De Witte, L. De Vocht, R. Verborgh, E. Mannens, et al. Big linked data etl benchmark on cloud commodity hardware. In *Proceedings of the International Workshop on Semantic Big Data*, page 12. ACM, 2016.
7. J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres, and M. Arias. Binary RDF representation for publication and exchange (HDT). *Journal of Web Semantics*, 19:22–41, 2013.
8. S. Ferré, A. Hermann, and M. Ducassé. Semantic Faceted Search: Safe and Expressive Navigation in RDF Graphs. Research Report PI 1964, Jan. 2011.
9. Y. Guo, Z. Pan, and J. Heflin. LUBM: A benchmark for OWL knowledge base systems. *Web Semantics: Science, Services and Agents on the World Wide Web*, 3(2):158–182, 2005.
10. A. Loizou, R. Angles, and P. Groth. On the formulation of performant {SPARQL} queries. *Web Semantics: Science, Services and Agents on the World Wide Web*, 31:1 – 26, 2015.
11. M. Morsey, J. Lehmann, S. Auer, and A.-C. Ngonga Ngomo. DBpedia SPARQL benchmark–performance assessment with real queries on real data. *The Semantic Web–ISWC 2011*, pages 454–469, 2011.
12. A.-C. N. Ngomo and M. Röder. Hobbit: Holistic benchmarking for big linked data.
13. A. Schätzle, M. Przyjaciel-Zablocki, A. Neu, and G. Lausen. Sempala: Interactive SPARQL Query Processing on Hadoop. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Proceedings, Part I*, pages 164–179, 2014.
14. M. Schmidt, T. Hornung, G. Lausen, and C. Pinkel. SPˆ 2Bench: a SPARQL performance benchmark. In *Data Engineering, 2009. ICDE'09. IEEE 25th International Conference on*, pages 222–233. IEEE, 2009.
15. A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. FedX: Optimization Techniques for Federated Query Processing on Linked Data. In *The Semantic Web - ISWC 2011, Proceedings, Part I*, pages 601–616, 2011.
16. H. Wu, T. Fujiwara, Y. Yamamoto, J. Bolleman, and A. Yamaguchi. BioBenchmark Toyama 2012: an evaluation of the performance of triple stores on biological data. *Journal of biomedical semantics*, 5(1):1, 2014.