An Evaluation of SPARQL Federation Engines Over Multiple Endpoints

Muhammad Saleem
AKSW, Uni Leipzig, Germany
saleem@informatik.uni-leipzig.de

Yasar Khan
INSIGHT, NUIG, Ireland
yasar.khan@insight-centre.org

Ali Hasnain
INSIGHT, NUIG, Ireland
ali.hasnain@insight-centre.org

Ivan Ermilov
AKSW, Uni Leipzig, Germany
iermilov@informatik.uni-leipzig.de

Axel-Cyrille Ngonga Ngomo
AKSW, Uni Leipzig, Germany
Uni-Paderborn, Germany
ngonga@informatik.uni-leipzig.de
axel.ngonga@upb.de

ABSTRACT

Due to decentralized and linked architecture underlying Linking Data, running complex queries often require collecting data from multiple RDF datasets. The optimization of the runtime of such queries, called *federated queries*, is of central importance to ensure the scalability of Semantic-Web and Linked-Data-driven applications. This has motivated a considerable body of work on SPARQL query federation. However, previous evaluations of SPARQL query federation engines do not evaluate the performance of these engines pertaining to the different steps involved in the federated query processing. Consequently, it is difficult to pinpoint the components of the federation engines that need to be improved. This work presents an extended summary of the fine-grained evaluation of SPARQL endpoint federation systems performed in [13]. Beside query runtime as an evaluation criterion, we extend the scope of our performance evaluation by considering additional measures which are important but have not been paid much attention to in the previous studies. Our experimental outcomes lead to novel insights for improving current and future SPARQL federation systems.

CCS CONCEPTS

• **General and reference** → *Metrics*; *Evaluation*; *Performance*;

ACM Reference format:
Muhammad Saleem, Yasar Khan, Ali Hasnain, Ivan Ermilov, and Axel-Cyrille Ngonga Ngomo. 2018. An Evaluation of SPARQL Federation Engines Over Multiple Endpoints. In *Proceedings of The Web Conference 2018, Lyon, France, April 2018,* 5 pages.
https://doi.org/

1 INTRODUCTION

The ultimate goal of the research performed in SPARQL query federation to optimize the overall query runtime of the federated engines. However, the query runtime can be affected by a variety of factors such as the efficiency of the source selection, the quality of the query execution plans, and the quality of the implementations. For example, the over-estimation of the number of query-relevant data sources often results in unnecessary intermediate results, generate extra network traffic, and thus can significantly affect the overall query runtime. Current evaluations [1, 3, 8, 12, 14, 19, 20] of SPARQL query federation systems regarded the overall query runtime as their central evaluation criterion. The granularity of such evaluations fails to provide results that allow understanding *why* the query runtimes of systems can differ drastically. Furthermore, as pointed out by [9], the current testbeds [2, 10, 17, 18] for evaluating, comparing, and eventually improving SPARQL query federation systems still have some limitations. Especially, the *partitioning of data* as well as *the SPARQL clauses* used cannot be tailored sufficiently, although they are known to have a direct impact on the behaviour of SPARQL query federation systems.

The aim of this paper is to experimentally evaluate a large number of SPARQL 1.0 query federation systems within a more fine-granular setting in which we can measure the time required to complete different steps of the SPARQL query federation process. To achieve this goal, we conducted a public survey[1] and collected information regarding 14 existing federated system implementations, their key features, and supported SPARQL clauses. Eight of the systems which participated in this survey are publicly available. However, two out of the eight with public implementation do not make use of the SPARQL endpoints and were thus not considered further in this study. In the next step and like in previous evaluations, we compared the remaining six systems [1, 3, 7, 11, 19, 20] with respect to the traditional performance criterion (i.e., the query execution time) using the commonly used benchmark FedBench [17]. In addition, we also compared these six systems with respect to their *answer completeness*, source selection approach in terms of the *total number of sources* they selected, the *total number of SPARQL ASK requests* they used and *source selection time.* For the sake of completeness, we also performed a comparative analysis (based on the survey outcome) of the key functionality of the 14 systems which participated in our survey.

To provide a quantitative analysis of the effect of *data partitioning* on the systems at hand, we extended both FedBench [17] and SP^{2}Bench [18] by distributing the data upon which they rely.

[1]Survey: http://goo.gl/iXvKVT, Results: http://goo.gl/CNW5UC

To this end, we used the slice generation tool[2] described in [16]. This tool allows creating any number of subsets of a given dataset (called *slices*) while controlling the number of slices, the amount of overlap between the slices as well as the size distribution of these slices. The resulting slices were distributed across various data sources (SPARQL endpoints) to simulate a highly federated environment. In our experiments, we made use of both FedBench [17] and SP^{2}Bench [18] queries to ensure that we *cover the majority of the SPARQL query types and clauses*.

Our main contributions are summarized as follows: (1) We present the results of a public survey which allows us to provide a crisp overview of categories of SPARQL federation systems as well as provide their implementation details, features, and supported SPARQL clauses. (2) We present (to the best of our knowledge) the most comprehensive experimental evaluation of open-source SPARQL federations systems in terms of their source selection and overall query runtime using in two different evaluation setups. (3) We extend both FedBench and SP^{2}Bench to mirror highly distributed data environments and test SPARQL endpoint federation systems for their parallel processing capabilities. (4) We provide a detailed discussion of experimental results and reveal novel insights for improving existing and future federation systems.

2 EVALUATION RESULTS AND DISCUSSION

2.1 Survey results

Based on our survey results[3], existing SPARQL query federation approaches can be divided into three main categories:

1. Query federation over multiple SPARQL endpoints: In this setting, RDF data is made available via SPARQL endpoints. The federation engine makes use of endpoint URLs to federate sub-queries and collect results back for integration. Fedex [19], LHD [20], SPLENDID [3], DAW [16], HiBISCuS [15], ANAPSID [1], DARQ [11] and others implement this form of federation over multiple SPARQL endpoints.

2. Query federation over Linked Data: This type of approaches relies on the Linked Data principles[4] for query execution. The set of data sources which can contribute results into the final query resultset is determined by using URI lookups during the query execution itself. Query federation over Linked Data does not require the data providers to publish their data as SPARQL endpoints. Instead, the only requirement is that the RDF data follows the Linked Data principles. LDQPS [5], and SIHJoin [6] implement federation over linked data (LDF).

3. Query federation on top of Distributed Hash Tables: This type of federation approaches stores RDF data on top of Distributed Hash Tables (DHTs) and use DHT indexing to federate SPARQL queries over multiple RDF nodes. Atlas [4] implements DHT federation.

Each of the above main category can be further divided into three sub-categories:

[2]https://code.google.com/p/fed-eval/wiki/SliceGenerator
[3]Available at http://goo.gl/CNW5UC
[4]http://www.w3.org/DesignIssues/LinkedData.html

(a) Catalog/index-assisted solutions: These approaches utilize dataset summaries that have been collected in a pre-processing stage. DARQ is an example of such solutions.

(b) Catalog/index-free solutions: In these approaches, the query federation is performed without using any stored data summaries. The data source statistics can be collected on-the-fly before the query federation starts. FedX is an example of such solutions.

(c) Hybrid solutions: In these approaches, some of the data source statistics are pre-stored while some are collected on-the-fly, e.g., using SPARQL ASK queries. SPLENDID is an example of such solutions.

Table 1 summarizes the survey outcome w.r.t. different features supported by systems.

2.2 SlicedBench

As pointed out in [9] the data partitioning can affect the overall performance of SPARQL query federation engines. To quantify this effect, we created 10 slices of each of the FedBench's datasets and distributed this data across 10 local virtuoso SPARQL endpoints (one slice per SPARQL endpoint). Thus, every SPARQL endpoint contained one slice from each of the 10 datasets. This creates a highly fragmented data environment.

2.3 Efficiency of Source Selection

We define efficient source selection in terms of: (1) the total number of triple pattern-wise sources selected (#T), (2) the total number of SPARQL ASK requests (#AR) used to obtain (1), and (3) the source selection time (SST). Table 2 shows the results of these three metrics for the selected approaches. Note that CD represents the cross domain, LS represents the life sciences, and LD represents the Linked Data queries of the FedBench.

Overall, ANAPSID is the most efficient approach in terms of total number of triple pattern-wise sources selected. LHD, DARQ, ADERIS do not make use of the SPARQL ASK requests during source selection. FedX (100% cached) is the fastest in terms of source selection time. It is important to note that FedX(100% cached) means that the complete source selection is performed by using only cache, i.e., no SPARQL ASK request is used. This the best-case scenario for FedX and very rare in practical cases.

2.4 Answer Completeness

Two or more engines are only comparable to each other if they provide the same result set for a given query. Table 3 shows the queries and federated engines for which we did not receive the complete results. As an overall answer completeness evaluation, only FedX is always able to retrieve complete results. It is important to note that these results are directly connected to the answer completeness results presented in survey Table 1; which shows only FedX is able to provide complete results among the selected systems.

2.5 Query Runtime

The comparison of the overall performance of each approach is summarised in Figure 1, where we show the average query execution time for the queries in CD, LS, LD, and SP^{2}Bench sub-groups. As an

Table 1: Survey outcome: System's features (R.C. = Results Completeness, P.R.R. = Partial Results Retrieval, N.B.O. = No Blocking Operator, A.Q.P. = Adaptive Query Processing, D.D. = Duplicate Detection, P.B.Q.P = Policy-based Query Planning, Q.R.E. = Query Runtime Estimation, Top-K.Q.P = Top-K query processing)

Systems	R.C.	P.R.R.	N.B.O / A.Q.P.	D. D.	P.B.Q.P	Provenance	Q.R.E	Top-K.Q.P
FedX	✓	✗	✗	✗	✗	✗	✗	✗
LHD	✗	✗	✗	✗	✗	✗	✗	✗
SPLENDID	✗	✗	✗	✗	✗	✗	✗	✗
FedSearch	✓	✗	✗	✗	✗	✗	✗	✗
GRANATUM	✗	✗	✗	✗	partial	partial	✗	✗
Avalanche	✗	✓	✓	partial	✗	✗	✗	✗
DAW	✗	✓	based on underlying system	✓	✗	✗	✗	✗
ANAPSID	✗	✗	✓	✗	✗	✗	✗	✗
ADERIS	✗	✗	✓	✗	✗	✗	✗	✗
DARQ	✗	✗	✗	✗	✗	✗	✗	✗
LDQPS	✗	✗	✓	✗	✗	✗	✗	✗
SIHJoin	✗	✗	✓	✗	✗	✗	✗	✗
WoDQA	✓	✗	✗	✗	✗	✗	✗	✗
Atlas	✗	✗	✗	partial	✗	✗	✗	✗

Table 2: Comparison of the efficiency of source selection in terms of total triple pattern-wise sources selected #T, total number of SPARQL ASK requests #AR, and source selection time SST in msec. SST* represents the source selection time for FedX(100% cached i.e. #A =0 for all queries). For ANAPSID, SST represents the query decomposition time. (T/A = Total/Avg., where Total is for #T, #AR, and Avg. is SST, SST*)

	FedX				SPLENDID			LHD			DARQ			ANAPSID			ADERIS		
Query	#T	#AR	SST	SST*	#T	#AR	SST	#T	#AR	SST	#T	#AR	SST	#T	#AR	SST	#T	#AR	SST
FedBench																			
CD	78	252	151	7	78	44	256	112	0	6	84	0	7	36	43	186	84	0	6
LS	56	297	147	8	56	33	236	105	0	6	77	0	12	44	63	477	70	0	5
LD	108	369	139	8	108	19	221	119	0	6	122	0	7	54	37	804	119	0	5
T/A	242	918	146	8	242	96	237	336	0	6	283	0	9	134	143	489	273	0	5
SlicedBench																			
CD	182	280	284	9	182	110	470	225	0	7	195	0	8	163	228	3183	195	0	7
LS	125	330	207	7	125	100	308	163	0	7	133	0	8	102	143	2897	117	0	7
LD	243	410	323	9	243	140	557	324	0	8	324	0	7	183	270	2908	324	0	8
SP2B	521	660	212	9	521	180	738	593	0	8	420	0	9	244	265	RE	173	0	6
T/A	1071	1680	256	8	1071	530	518	1305	0	7	1072	0	8	692	906	2996	809	0	7

Table 3: The queries for which some system's did not retrieve complete results. The values inside bracket shows the actual results size. "-" means the results completeness cannot be determined due to query execution timed out. Incomplete results are highlighted in bold

	CD1(90)	CD7(1)	LS1(1159)	LS2(333)	LS3(9054)	LS5(393)	LD1(309)	LD3(162)	LD9(1)	SP2B-3a(27789)	SP2B-6(>70k)
SPLENDID	90	1	1159	333	9054	393	**308**	**159**	1	-	-
LHD	**77**	1	**0**	**322**	**0**	**0**	309	162	1	-	-
ANAPSID	90	**0**	1159	333	9054	393	309	162	1	**0**	**0**
ADERIS	**77**	1	1159	333	9054	393	309	162	1	-	-
DARQ	90	1	1159	333	9054	393	309	162	**0**	-	-
FedX	90	1	1159	333	9054	393	309	162	1	27789	-

overall performance evaluation based on FedBench, FedX(cached) outperformed FedX(first run) on all of the 25 queries. FedX(first run) in turn outperformed LHD on 17 out of 22 commonly supported queries (LHD retrieve zero results for three queries). LHD is better than SPLENDID in 13 out of 22 comparable queries. SPLENDID outperformed ANAPSID in 15 out of 24 queries while ANAPSID outperforms DARQ in 16 out of 22 commonly supported queries. For SlicedBench, FedX(cached) outperformed FedX(first run) in 29 out of 36 comparable queries. In turn FedX(first run) outperformed LHD in 17 out of 24 queries. LHD is better than SPLENDID in 17

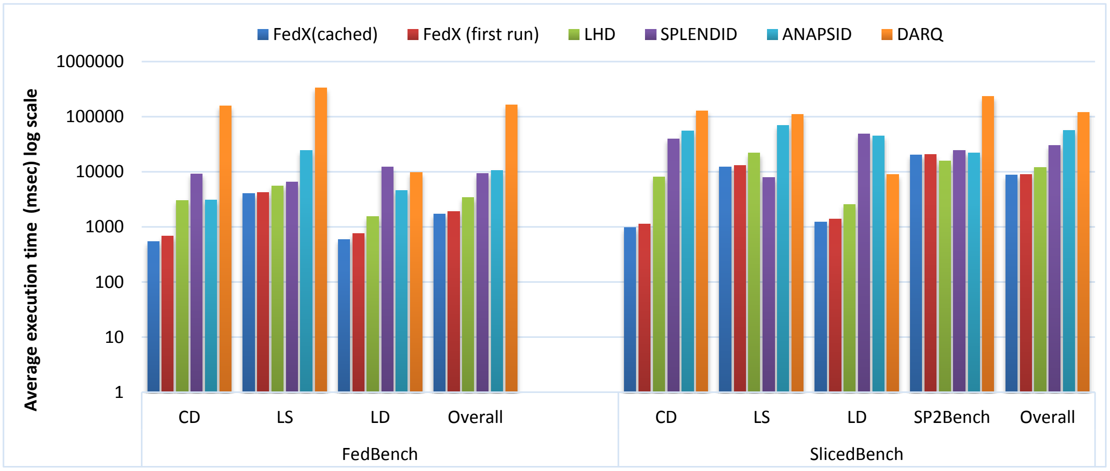
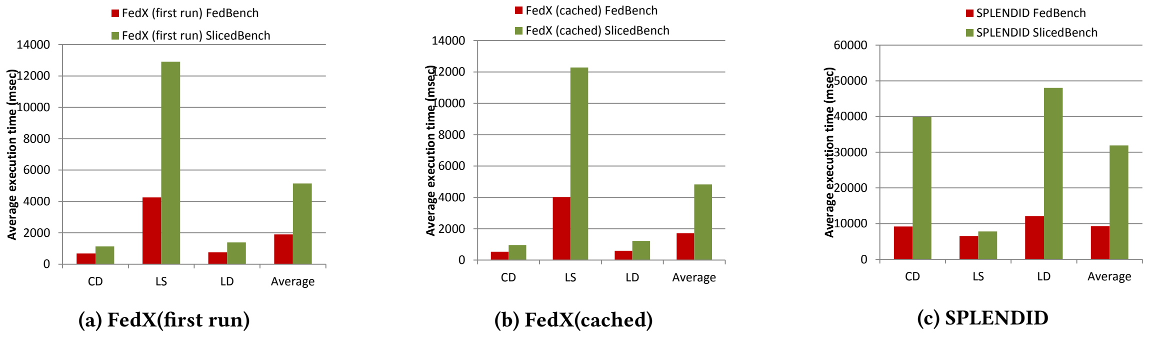


Figure 1: Overall performance evaluation (ms)



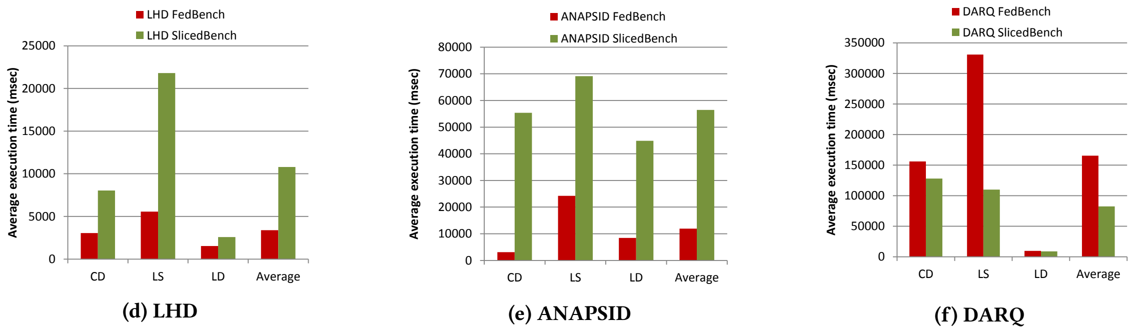


Figure 2: Effect of the data partitioning

out of 24 comparable queries. SPLENDID outperformed ANAPSID in 17 out of 26 which in turn outperformed DARQ in 12 out of 20 commonly supported queries. No results were retrieved for majority of the queries in case of ADERIS, hence not included to this section. All of the above improvements are significant based on Wilcoxon signed ranked test with significance level set to 0.05.

2.6 Effect of the data partitioning

The SlicedBench extension of FedBench can also be utilized to test the capability of parallel execution of queries in SPARQL endpoint federation system. Figure 2 shows the effect of data partitioning on the selected federation engines. The performance of FedX(cached) and DARQ is improved with partitioning while the performance of FedX(first run), SPLENDID, ANAPSID, and LHD is reduced. As an overall evaluation result, FedX(first run)'s performance is reduced by 214%, FedX(cached)'s is reduced 199%, SPLENDID's is reduced by 227%, LHD's is reduced by 293%, ANAPSID's is reduced by 382%, and interestingly DARQ's is improved by 36%. This results suggest that FedX is the best system in terms of parallel execution of queries, followed by SPLENDID, LHD, and ANAPSID.

REFERENCES

[1] Maribel Acosta, Maria-Esther Vidal, Tomas Lampo, Julio Castillo, and Edna Ruckhaus. 2011. ANAPSID: An Adaptive Query Processing Engine for SPARQL Endpoints. In *The Semantic Web âĂŞ ISWC 2011*, Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Noy, and Eva Blomqvist (Eds.). Lecture Notes in Computer Science, Vol. 7031. Springer Berlin Heidelberg, 18–34. https://doi.org/10.1007/978-3-642-25073-6_2

[2] Christian Bizer and Andreas Schultz. 2009. The Berlin SPARQL Benchmark. In *International Journal on Semantic Web and Information Systems (IJSWIS)*, Vol. 5. IGI Global, 1–24.

[3] Olaf Görlitz and Steffen Staab. 2011. SPLENDID: SPARQL Endpoint Federation Exploiting VoID Descriptions. In *O. Hartig, A. Harth, and J. F. Sequeda, editors, 2nd International Workshop on Consuming Linked Data (COLD 2011) in CEUR Workshop Proceedings*, Vol. 782.

[4] Zoi Kaoudi, Manolis Koubarakis, Kostis Kyzirakos, Iris Miliaraki, Matoula Magiridou, and Antonios Papadakis-Pesaresi. 2010. Atlas: Storing, Updating and Querying RDF(S) Data on Top of DHTs. In *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 8. Elsevier, 271–277. https://doi.org/10.1016/j.websem.2010.07.001

[5] GÃijnter Ladwig and Thanh Tran. 2010. Linked Data Query Processing Strategies. In *The Semantic Web âĂŞ ISWC 2010*, PeterF. Patel-Schneider, Yue Pan, Pascal Hitzler, Peter Mika, Lei Zhang, JeffZ. Pan, Ian Horrocks, and Birte Glimm (Eds.). Lecture Notes in Computer Science, Vol. 6496. Springer Berlin Heidelberg, 453–469. https://doi.org/10.1007/978-3-642-17746-0_29

[6] GÃijnter Ladwig and Thanh Tran. 2011. SIHJoin: Querying Remote and Local Linked Data. In *The Semantic Web: Research and Applications*, Grigoris Antoniou, Marko Grobelnik, Elena Simperl, Bijan Parsia, Dimitris Plexousakis, Pieter De Leenheer, and Jeff Pan (Eds.). Lecture Notes in Computer Science, Vol. 6643. Springer Berlin Heidelberg, 139–153. https://doi.org/10.1007/978-3-642-21034-1_10

[7] Steven Lynden, Isao Kojima, Akiyoshi Matono, and Yusuke Tanimura. 2011. ADERIS: An Adaptive Query Processor for Joining Federated SPARQL Endpoints. In *R. Meersman, T. Dillon, P. Herrero, A. Kumar, M. Reichert, L. Qing, B.-C. Ooi, E. Damiani, D.C. Schmidt, J. White, M. Hauswirth, P. Hitzler, M. Mohania, editors, On the Move to Meaningful Internet Systems (OTM2011), Part II. LNCS.* Vol. 7045. Springer Heidelberg, 808–817.

[8] Gabriela Montoya, Maria-Esther Vidal, and Maribel Acosta. 2012. A Heuristic-Based Approach for Planning Federated SPARQL Queries. In *J. F. Sequeda, A. Harth, and O. Hartig, editors, 3rd International Workshop on Consuming Linked Data (COLD 2012) in CEUR Workshop Proceedings*, Vol. 905.

[9] Gabriela Montoya, Maria-Esther Vidal, Oscar Corcho, Edna Ruckhaus, and Carlos Buil-Aranda. 2012. Benchmarking Federated SPARQL Query Engines: Are Existing Testbeds Enough? In *P. Cudre Mauroux, J. Heflin, E. Sirin, T. Tudorache, J. Euzenat, M. Hauswirth, J.X. Parreira, J. Hendler, G. Schreiber, A. Bernstein, E. Blomqvist, editors, The Semantic Web – ISWC 2012, Part II. LNCS.* Vol. 7650. Springer Heidelberg, 313–324.

[10] Mohamed Morsey, Jens Lehmann, Sören Auer, and Axel-Cyrille Ngonga Ngomo. 2011. DBpedia SPARQL Benchmark - Performance Assessment with Real Queries on Real Data. In *Aroyo, L., Welty, C., Alani, H., Taylor, J., Bernstein, A., Kagal, L., Noy, N., Blomqvist, E. (eds.), International Semantic Web Conference (ISWC2011), Part I. LNCS*, Vol. 7031. Springer Heidelberg, 454–469.

[11] Bastian Quilitz and Ulf Leser. 2008. Querying Distributed RDF Data Sources with SPARQL. In *The Semantic Web: Research and Applications*, Sean Bechhofer, Manfred Hauswirth, JÃűrg Hoffmann, and Manolis Koubarakis (Eds.). Lecture Notes in Computer Science, Vol. 5021. Springer Berlin Heidelberg, 524–538. https://doi.org/10.1007/978-3-540-68234-9_39

[12] Nur Aini Rakhmawati, Jürgen Umbrich, Marcel Karnstedt, Ali Hasnain, and Michael Hausenblas. 2013. Querying over Federated SPARQL Endpoints - A State of the Art Survey. In *CoRR*, Vol. abs/1306.1723. http://arxiv.org/abs/1306.1723

[13] Muhammad Saleem, Yasar Khan, Ali Hasnain, Ivan Ermilov, and Axel-Cyrille Ngonga Ngomo. 2016. A fine-grained evaluation of SPARQL endpoint federation systems. *Semantic Web* 7, 5 (2016), 493–518.

[14] Muhammad Saleem, R. Kamdar Maulik, Iqbal Aftab, Sampath Shanmukha, Helena F. Deus, and Axel-Cyrille Ngonga Ngomo. 2013. Fostering Serendipity through Big Linked Data. In *Semantic Web Challenge at International Semantic Web Conference.*

[15] Muhammad Saleem and Axel-Cyrille Ngonga Ngomo. 2014. HiBISCuS: Hypergraph-Based Source Selection for SPARQL Endpoint Federation. In *The Semantic Web: Trends and Challenges*, Valentina Presutti, Claudia dâĂŹAmato, Fabien Gandon, Mathieu dâĂŹAquin, Steffen Staab, and Anna Tordai (Eds.). Lecture Notes in Computer Science, Vol. 8465. Springer International Publishing, 176–191. https://doi.org/10.1007/978-3-319-07443-6_13

[16] Muhammad Saleem, Axel-Cyrille Ngonga Ngomo, Josiane Xavier Parreira, HelenaF. Deus, and Manfred Hauswirth. 2013. DAW: Duplicate-AWare Federated Query Processing over the Web of Data. In *The Semantic Web âĂŞ ISWC 2013*, Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, JosianeXavier Parreira, Lora Aroyo, Natasha Noy, Chris Welty, and Krzysztof Janowicz (Eds.). Lecture Notes in Computer Science, Vol. 8218. Springer Berlin Heidelberg, 574–590. https://doi.org/10.1007/978-3-642-41335-3_36

[17] Michael Schmidt, Olaf GÃűrlitz, Peter Haase, GÃijnter Ladwig, Andreas Schwarte, and Thanh Tran. 2011. FedBench: A Benchmark Suite for Federated Semantic Data Query Processing. In *The Semantic Web âĂŞ ISWC 2011*, Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Noy, and Eva Blomqvist (Eds.). Lecture Notes in Computer Science, Vol. 7031. Springer Berlin Heidelberg, 585–600. https://doi.org/10.1007/978-3-642-25073-6_37

[18] Michael Schmidt, Thomas Hornung, Georg Lausen, and Christoph Pinkel. 2009. SPˆ 2Bench: A SPARQL Performance Benchmark. In *Proceedings of the 25th International Conference on Data Engineering ICDE*. IEEE, 222–233.

[19] Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. 2011. FedX: Optimization Techniques for Federated Query Processing on Linked Data. In *The Semantic Web âĂŞ ISWC 2011*, Lora Aroyo, Chris Welty, Harith Alani, Jamie Taylor, Abraham Bernstein, Lalana Kagal, Natasha Noy, and Eva Blomqvist (Eds.). Lecture Notes in Computer Science, Vol. 7031. Springer Berlin Heidelberg, 601–616. https://doi.org/10.1007/978-3-642-25073-6_38

[20] Xin Wang, Thanassis Tiropanis, and Hugh C. Davis. 2013. LHD: Optimising Linked Data Query Processing Using Parallelisation. In *C. Bizer, T. Heath, T. Berners-Lee, M. Hausenblas, S. Auer, editors, Proceedings of the WWW2013 Workshop on Linked Data on the Web in CEUR Workshop Proceedings*, Vol. 996. http://eprints.soton.ac.uk/350719/