Eurostars Project

3DFed – Dynamic Data Distribution and Query Federation

Project Number: E!114681 **Start Date of Project:** 2021/04/01 **Duration:** 36 months

Deliverable 3.2
Final Report on the Automatic Data Distribution

Dissemination Level	Public
Due Date of Deliverable	January 31, 2023
Actual Submission Date	January 31, 2023
Work Package	WP3, Automatic Data Distribution & Dynamic Exchange
Deliverable	D3.2
Type	Report
Approval Status	Final
Version	1.0
Number of Pages	10

Abstract: In this deliverable, the two workload-aware RDF graph partitioning techniques PCM and PCG proposed in the previous deliverable (D3.1: Initial Report on the Automatic Data Distribution) are evaluated using various real-world data and query benchmarks. The overall results indicate the superiority of the proposed techniques over the previous techniques in terms of better query runtime performance, number of timeout queries, overall rank score, and number of distinct sources selected.

3DFed Project by Eurostars.

History

Version	Date	Reason	Revised by
0.1	01/12/2022	Initial Template & Deliverable Structure	Mohammad Sajjadi
0.2	09/01/2023	Initial Draft	Asal Alikhani & Muhammad Saleem
0.3	23/01/2023	Issued for review	Mohammad Sajjadi
0.4	27/01/2023	Review	Mirko Spasić
1.0	31/01/2023	Final Submission	Mohammad Sajjadi

Author List

Organization	Name	Contact Information
elevait GmbH & Co. KG	Asal Alikhani	asal.alikhani@elevait.de
University of Paderborn	Muhammad Saleem	saleem@informatik.uni-leipzig.de
elevait GmbH & Co. KG	Martin Voigt	martin.voigt@elevait.de
University of Paderborn	Axel-Cyrille Ngonga Ngomo	axel.ngonga@upb.de
elevait GmbH & Co. KG	Mohammad Sajjadi	mohammad.sajjadi@elevait.de
OpenLink Software	Mirko Spasić	mspasic@openlinksw.com

Contents

1 Introduction

In the initial step of T3.1, two novel RDF graph partitioning techniques, PCM and PCG, were proposed by using different clustering techniques that exploit the predicates co-occurrences in the querying workload and were documented in the previous deliverable (D3.1: Initial Report on the Automatic Data Distribution). In Task T3.2, the impact of the query workload on the accuracy of the data distribution with respect to various measurements is evaluated and the final results are documented in this deliverable. PCM and PCG are evaluated against state-of-the-art graph distribution techniques in terms of query runtime performance, number of timeout queries, overall rank score, and number of distinct sources selected. We performed extensive experiments based on various real-world datasets and query evaluation environments.

The evaluation results show the superiority of the proposed techniques (PCM and PCG) compared to other partitioning techniques in terms of better query runtime performance. The source codes, datasets, and instructions for reproducing the complete results are publicly available on GitHub[1].

2 Evaluation

In this section we summarize the evaluation carried out. Section 2.1 describes the setup of the evaluation, including the datasets used, benchmark queries, workloads, and partition details. Performance measures are presented, and the hardware and software specifications of the machines on which the experiments are performed are given. In Section 2.2 we examine the observed results of our experiments.

2.1 Evaluation Setup

We have exactly reused the evaluation setup discussed in [1]. The reasons for choosing this evaluation setup are twofold:

- Since our proposed techniques require query workloads, we wanted to use real-world query workloads (i.e., collected from public SPARQL endpoints of real-world RDF datasets), and real-world RDF benchmarks.
- We wanted our results to be comparable to the results presented in [1].

Datasets. As in [1], for partitioning, we used two real-world datasets:

- DBpedia 3.5.1
- Semantic Web Dog Food (SWDF).

Some statistics of these datasets are shown in Table 1.

Benchmark Queries (test queries). As in [1], we used four sets of real-world SPARQL benchmark queries (300 queries each):

- *SWDF BGP-only* is the SWDF benchmark containing only single BGP queries; the other SPARQL features such as `OPTIONAL`, `UNION` etc. are not used,
- *SWDF fully-featured* is the SWDF benchmark containing fully-featured (multiple BGPs, aggregates, functions etc.) SPARQL queries,

[1] `https://github.com/dice-group/workload-aware-rdf-partitioning`

Table 1: The total number of triples, distinct subjects, predicates, and objects within the used datasets.

	Triples	Subjects	Predicates	Objects
DBpedia	232,536,510	18,425,128	39,672	65,184,193
SWDF	304,583	36,879	185	95,501

- *DBpedia BGP-only* is the DBpedia benchmark only containing single BGP queries, and
- *DBpedia fully-featured* benchmark queries contain not only single BGPs but may also include additional constructs.

These benchmarks are generated by using FEASIBLE [3], a real benchmark generation framework, out of the query logs.

Workloads (train queries). We used a query workload of 3000 queries for both DBpedia and SWDF, which are selected from real-world query logs of these datasets. The reason for choosing 3000 was according to the *10-fold cross validation*[2], which suggests choosing 10% test queries and 90% training queries.

Partitioning Environments. As in [1], we used two different partitioning environments to evaluate our techniques:

- a clustered or distributed RDF storage environment, where the given dataset is distributed among n data nodes of a clustered triple store, and
- a purely federated environment, where the dataset is distributed among multiple SPARQL endpoints that are physically separated from each other, and a federation engine is used to perform the query processing task.

We used *Koral* [2] distributed RDF engines for the first type of partitioning environment. We chose *Koral* due to its flexibility in choosing different partitioning methods for data distribution among data nodes. Moreover, it has already been used in [1].

For the second type of partitioning environment, we used *FedX* [5] and *SemaGrow*. The reason for choosing these two engines was their different query planning strategies: *FedX* implements an index-free and heuristic-based query planner, while *SemaGrow* implements an index-assisted and cost-based query planner. Both were also used in [1]. It is important to note that Koral does not support many of the SPARQL features used in the fully-featured SPARQL benchmarks. Therefore, we used BGP-only queries in our Koral-based evaluation.

Number of Partitions. Following [1] and [4], we generated 10 partitions of the selected datasets. Therefore, 10 slaves were created in Koral, each responsible for one partition. Similarly, we used 10 Linux-based Openlink Virtuoso 7.1[3] SPARQL endpoints, each storing one partition. The selected federation engines physically federate the given SPARQL query over these endpoints.

Selected RDF Graph Partitioning Techniques. We selected state-of-the-art techniques for partitioning RDF graph based on the following criteria:

- open source and configurable,

[2]`https://machinelearningmastery.com/k-fold-cross-validation/`
[3]`https://virtuoso.openlinksw.com/`

- working for RDF data,
- scaleable to medium-large datasets, such as DBpedia in our case,
- take the RDF dataset and/or workload as input and give the required number of RDF chunks as output, and
- do not require online services such as cloud or configuring online datasets.

Based on these criteria, we selected the following ten RDF graph partitioning techniques to consider in the evaluation results: Horizontal, Subject-Based, Predicate-Based, Hierarchical, Recursive-Bisection, TCV-Min, Min-Edgecut, Partout, PCG, PCM. Note that the workload-aware technique Partout worked only for SWDF datasets; for DBpedia, it was unable to partition the dataset in 3 days[4].

Performance Measures. As in [1], we used five performance metrics: partitions generation time, Queries per Second (QpS) [1, 3], overall rank score, partitioning imbalance, and the total number of sources selected for the complete benchmark execution in a purely federated environment. We used a three minutes timeout for execution of each query [3, 1].

The rank score of the partitioning technique is defined as follows [1]:

Definition 1 (Rank Score) *Let t be the total number of partitioning techniques and b be the total number of benchmark executions used for the evaluation. Let $1 \leq r \leq t$ denote the rank number and let $O_p(r)$ denote the occurrence of a partitioning technique p placed at rank r. The rank score of the partitioning technique p is defined as follows:*

$$s := \sum_{r=1}^{t} \frac{O_p(r) \times (t-r)}{b(t-1)}, 0 \leq s \leq 1$$

In our evaluation, we have a total of ten partitioning techniques (i.e., t = 10 for SWDF, and 9 for DBpedia) and a total of 10 benchmarks executions (i.e., b = 10, 4 benchmarks by FedX + 4 benchmarks by SemaGrow + 2 benchmarks by Koral).

The partitioning imbalance in the size of the generated partitions is defined as follows [1]:

Definition 2 (Partitioning Imbalance) *Let n be the total number of partitions generated by a partitioning technique and let $P_1, P_2, \ldots P_n$ be the set of these partitions, ordered by increasing size of the number of triples. The imbalance in partitions is defined as the Gini coefficient:*

$$b := \frac{2\sum\limits_{i=1}^{n}(i \times |P_i|)}{(n-1) \times \sum\limits_{j=1}^{n}|P_j|} - \frac{n+1}{n-1}, 0 \leq b \leq 1$$

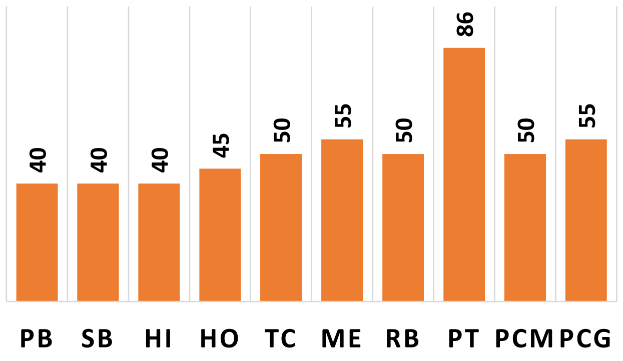
Hardware and Software Specifications. The hardware and software configuration for our techniques is the same as in [1], i.e., all our experiments are executed on an Ubuntu-based machine with Intel Xeon 2.10 GHz, 64 cores, and 512 GB of RAM. We conducted our experiments on local copies of Virtuoso (version 7.1) SPARQL endpoints. We used default configurations for FedX, SemaGrow and Koral (except that the slaves in Koral were changed from 2 to 10).

[4]We have discussed this issue with the authors of the Partout system.

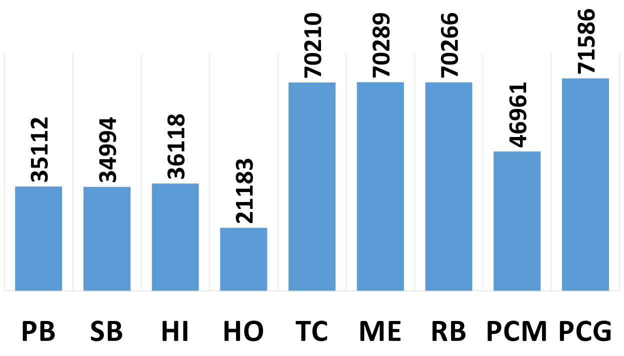
2.2 Evaluation Results

Note that the results of Partout (PT) are shown only for SWDF, as it was unable to partition the DBpedia dataset.

Partition Generation Time. Figure 1 shows a comparison of the total time it took to generate the required 10 partitions for both datasets used in our evaluation. PT took the most time, followed by PCG, Min-Edgecut, Recursive-Bisection, TCV-Min, PCM, Hierarchical, Predicate-Based, Subject-Based and Horizontal, respectively. The rest of the discussion is focused on PCG, as it is the best performing method proposed in our deliverable.



(a) SWDF partitioning time



(b) DBpedia partitioning time

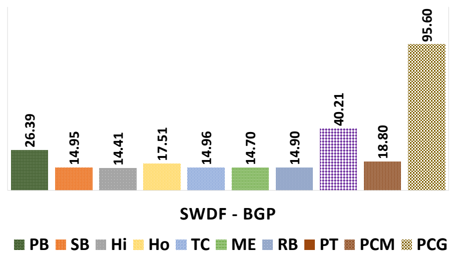
Figure 1: Time in seconds required to create 10 partitions. (PB = Predicate-Based, SB = Subject-Based, HI = Hierarchical, HO = Horizontal, TC = TCV-Min, ME = Min-Edgecut, RB = Recursive Bisection, PT = Partout)

Query per Second (QpS). Query per Second (QpS) is important for measuring query runtime performance with respect to different partitioning techniques. The idea is to find out how many queries are executed by a technique in one second. The higher the QpS value, the better the query runtime performance. Figure 2 shows a comparison of the QpS values of the selected partitioning techniques for each of the four benchmarks and three different query execution engines. Since Koral only supports BGP-only queries, we used SWDF-BGP and DBpedia-BGP benchmarks. For each timeout query, we added an extra 180 seconds to the total benchmark execution time. The results show that the proposed PCG method significantly outperforms the other partitioning methods in the majority of benchmark executions. In particular, PCG ranked first or second in 7/10 benchmark executions.

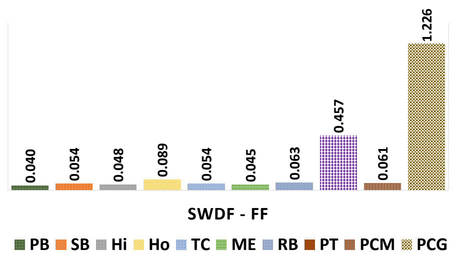
Rank Scores. It is difficult to determine an overall winner in terms of query runtime performance from the QpS results. The rank score shows the overall ranking of a particular method compared to other selected methods in the completed benchmark executions. The rank score is a value between 0 and 1, where 1 represents the highest ranking. Figure 3a represents the computed rank scores for each partitioning technique according to Definition 1. The overall results show that PCG has the highest ranked score, followed by PT, PCM, TCV-Min, Predicate-Based, Horizontal, Recursive-Bisection, Subject-Based, Hierarchical, and Min-Edgecut, respectively.

Partitioning Imbalance. Figure 3b shows the values of partitioning imbalance (Definition 2) of the partitions generated by the selected partitioning techniques. Horizontal partioning results in the smallest partitioning imbalance, followed by Hierarchical, Subject-Based, PCM, PCG, Min-Edgecut, Recursive-Bisection, TCV-Min, Partout, and Predicate-Based, respectively.

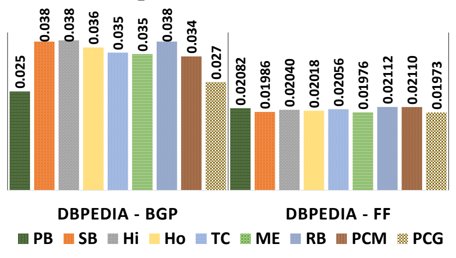
Number of Sources Selected. The number of sources (in our case SPARQL endpoints) selected by the federation engine to execute a given SPARQL query is an important performance metric for federated SPARQL querying engines [4]. The smaller the number of selected sources, the lower the communication cost, and hence the better the query runtime performance [1, 4]. Figure 4 shows the total number of distinct sources selected by FedX and SemaGrow. For SWDF, PT selects the smallest sources, followed by PCG and PCM. As an overall (1200 queries) source selection evaluation, PCG selects the smallest number of sources, followed by PCM, Predicate-Based,



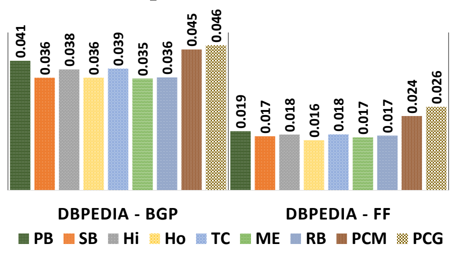
(a) QpS FedX SWDF-BGP



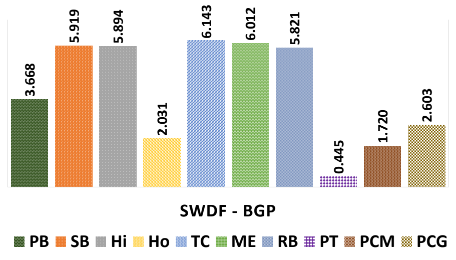
(b) QpS FedX SWDF-FF



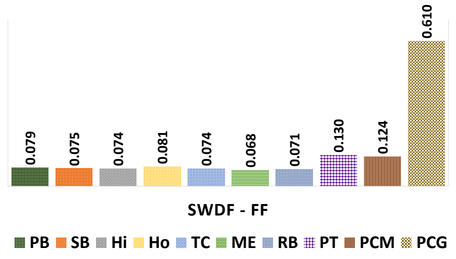
(c) QpS FedX DBpedia



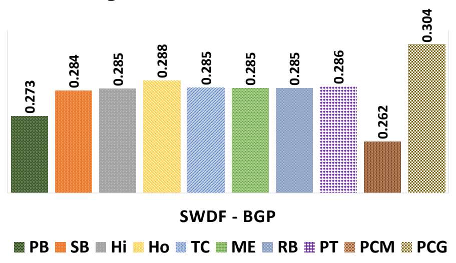
(d) QpS SemaGrow DBpedia



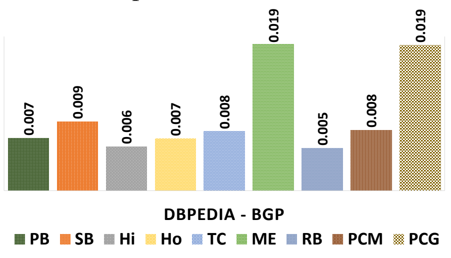
(e) QpS SemaGrow SWDF-BGP



(f) QpS SemaGrow SWDF-FF



(g) QpS Koral SWDF-BGP

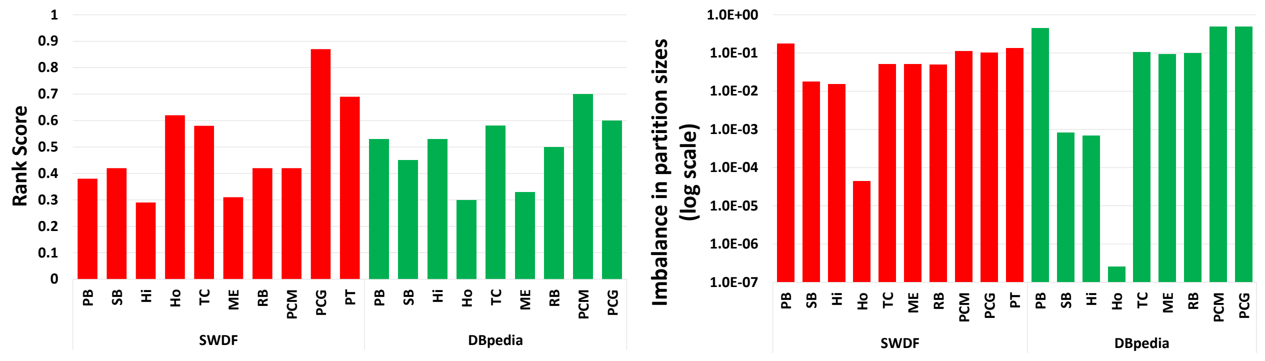


(h) QpS Koral DBpedia-BGP

Figure 2: QpS (for all four benchmarks) including timeouts. (SW = Semantic Web Dog Food, DB = DBpedia, BGP = Basic Graph Pattern, FF = Fully Featured, PB = Predicate-Based, SB = Subject-Based, Hi= Hierarchical, Ho = Horizontal, TC = TCV-Min, ME = Min-Edgecut, RB = Recursive Bisection, PT = Partout)

Min-Edgecut, TCV-Min, Recursive-Bisection, Subject-Based, Hierarchical and Horizontal, respectively.

Key observation. The results show that PCG significantly outperformed the other selected techniques for SWDF benchmarks (Figure 2a, Figure 2b, Figure 2e, Figure 2f, Figure 2g) in comparison to DBpedia benchmarks (Figure 2c, Figure 2d, Figure 2h). The average QpS of PCG is 20.07 for SWDF benchmarks, which is 3.30 times faster than the second best performing partitioning method. On the other hand, the average QpS of PCG is 0.028 for DBpedia benchmarks which is only 1.06 times faster than the second-best performing partitioning method. A detailed examination of the query workload and RDF datasets shows that the query workload used for our SWDF evaluation already covered 63.7% of the total 185 predicates used in the SWDF dataset. Thus, more predicates were correctly grouped into the desired partitions. On the other hand, the DBpedia dataset contains a total of 39672 distinct predicates, of which only 0.55% were covered by the used workload. As a result, a



(a) Rank scores (b) Partitioning imbalance

Figure 3: Rank scores and partitioning imbalance of the partitioning techniques. (PB = Predicate-Based, SB= Subject-Based, Hi= Hierarchical, Ho = Horizontal, TC = TCV-Min, ME Min-Edgecut, RB = Recursive Bisection, PT = Partout)

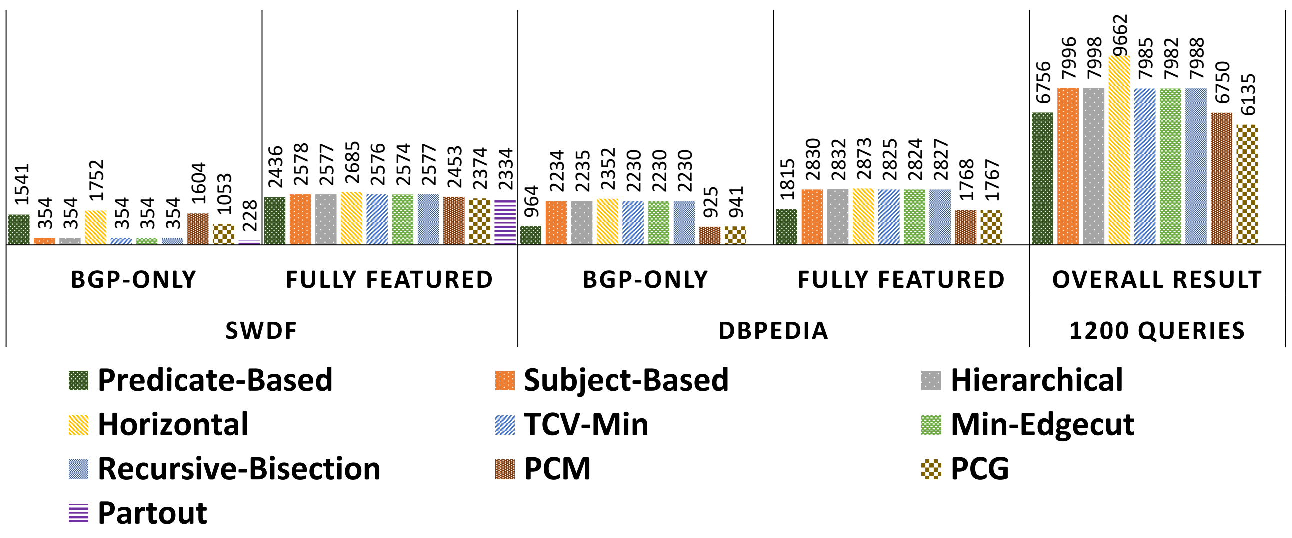


Figure 4: Total distinct sources selected

majority of the predicates were grouped into a separate partition for unused predicates. Consequently, only a small portion of the predicates were correctly mapped into correct partitions. In conclusion, the complexity of dataset and quality and size of workload can have a significant impact on the quality of partitioning achieved by the methods proposed in our work.

Number of Timeout Queries. Table 2 shows the number of timeout queries for each of the benchmarks and for each query execution engine. PCG has the fewest timeout queries (231 queries) followed by; Min-Edgecut (344 queries), PCM (397 queries), Subject-Based (422 queries), TCV-Min (455 queries), Predicate-Based (485 queries), Horizontal (498 queries), Hierarchical (544 queries), and Recursive-Bisection (556 queries), respectively.

3 Conclusion and Future Work

In D3.1, two RDF graph partitioning techniques were proposed:

Table 2: Timeout queries using FedX, SemaGrow and Koral

	FedX				SemaGrow				Koral		Total
	SWDF		DBpedia		SWDF		DBpedia		SWDF	DBpedia	
Partitioning	BGP	FF	BGP	FF	BGP	FF	BGP	FF	BGP	BGP	
Predicate-Based	0	35	32	73	0	20	35	81	0	209	485
Subject-Based	0	24	29	69	0	20	35	83	0	162	422
Hierarchical	0	28	28	70	0	20	33	79	0	286	544
Horizontal	0	12	31	73	0	19	34	83	0	246	498
TCV-Min	0	24	35	70	0	20	33	85	0	188	455
Min-Edgecut	0	30	35	74	0	22	34	84	0	65	344
Recursive-Bisection	0	19	32	70	0	21	35	81	0	298	556
Partout	0	1			0	1			0		2
PCM	0	20	40	68	0	11	20	51	0	187	397
PCG	0	0	52	70	0	2	16	32	0	59	231

- predicates co-occurrence-based partitioning using a greedy algorithm (PCG), and
- predicates co-occurrence-based partitioning using extended Markov clustering (PCM).

Both of these techniques make use of clustering algorithms to first cluster all the predicates used in the input query workload. Partitions are then created according to the clusters such that all triples pertaining to predicates in a given cluster are distributed into the same partition. The details of these techniques are documented in the previous deliverable (D3.1: Initial Report on the Automatic Data Distribution), and are evaluated in this deliverable using various real-world data and query benchmarks; the details are documented in this deliverable.

Our overall results indicate the superiority of our proposed techniques compared to the other techniques, in terms of better query runtime performance, number of timeout queries, overall rank score, and number of distinct sources selected. It was found that the quality and size of the workload is a key to obtain better results by the proposed methods. Our proposed techniques naturally lead to predicate-based indexing used in existing state-of-the-art RDF engines. Moreover, the created partitions can be easily managed in terms of index updates or dynamic shuffling of data among multiple data nodes of a clustered triple store.
In the future, we plan to measure the impact of query workloads on the accuracy of data distribution. In addition, it is highly possible that the initial distribution of data is suboptimal and therefore dynamic shuffling of data is required. To this end, proposing a self-updating, dynamic data distribution mechanism based on experienced workload is recommended.

References

[1] Akhter et al. An empirical evaluation of rdf graph partitioning techniques. In *European Knowledge Acquisition Workshop*, 2018.

[2] Janke et al. Koral: A glass box profiling system for individual components of distributed rdf stores. 2017.

[3] Saleem et al. Feasible: A feature-based sparql benchmark generation framework. In *International Semantic Web Conference*, 2015.

[4] Saleem et al. A fine-grained evaluation of sparql endpoint federation systems. 2016.

[5] Schwarte et al. Fedx: Optimization techniques for federated query processing on linked data. 2011.