Eurostars Project

3DFed – Dynamic Data Distribution and Query Federation

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

Deliverable 5.1
A Report on 3DFed Evaluation Based on Linked TCGA Use Case

Dissemination Level	Public
Due Date of Deliverable	March 31, 2024
Actual Submission Date	April 1, 2024
Work Package	WP5, Use Cases
Deliverable	D5.1
Type	Report
Approval Status	Final
Version	1.0
Number of Pages	10

Abstract: In this report, we present the evaluation results based on Linked TCGA dta as well as a fine-grained evaluation of the CostFed engine with state of the art.

The information in this document reflects only the author's views and Eurostars is not liable for any use that may be made of the information contained therein. The information in this document is provided "as is" without guarantee or warranty of any kind, express or implied, including but not limited to the fitness of the information for a particular purpose. The user thereof uses the information at his/ her sole risk and liability.

History

Version	Date	Reason	Revised by
0.1	15/03/2024	Initial Template & Deliverable Structure	Muhammad Saleem
0.2	25/03/2024	Initial draft completed	Muhammad Saleem
1.0	01/04/2024	Finalizing	Muhammad Saleem

Author List

Organization	Name	Contact Information
University of Paderborn	Muhammad Saleem	saleem@informatik.uni-leipzig.de
OpenLink Software	Milos Jovanovik	mjovanovik@openlinksw.com

Contents

1 Introduction

We first briefly describe the benchmark we used in this evaluation. In particular, we focus on the Linked TCGA data and the queries used. It is followed by the performance metrics. The evaluation results and the detailed investigation of the query plans comes in the end.

2 Linked TCGA Benchmark

2.1 Large Data: Linked TCGA

Linked TCGA is the RDF version of Cancer Genome Atlas[1] (TCGA) presented in [6]. This knowledge base contains cancer patient data generated by the TCGA pilot project, started in 2005 by the National Cancer Institute (NCI) and the National Human Genome Research Institute (NHGRI). Currently, Linked TCGA comprises a total of 20.4 billion triples[2] from 9000 cancer patients and 27 different tumour types. For each cancer patient, Linked TCGA contains expression results for the DNA methylation, Expression Exon, Expression Gene, miRNA, Copy Number Variance, Expression Protein, SNP, and the corresponding clinical data.

Given that we aimed to build a 1-billion-triple dataset, we selected 306 patient data randomly to reach the targeted 1 billion triples. The patients distributed evenly across 3 different cancer types, i.e. Cervical (CESC), Lung squamous carcinoma (LUSC) and Cutaneous melanoma (SKCM). The selection of the patients was carried out by consulting domain experts. This data is hosted in three TCGA SPARQL endpoints with all DNA methylation data in the first endpoint, all Expression Exon data in the second endpoint, and the remaining data in the third endpoint. Consequently, we created three different datasets, namely the Linked TCGA-M, Linked TCGA-E, and Linked TCGA-A containing methylation, exon, and all remaining data, respectively. Further statistics about these three datasets can be found in Table .

Table 1: Linked TCGA datasets statistics.

Dataset	#Triples	#Subjects	#Predicates	#Objects	#Classes	#Links	Structuredness
Linked TCGA-M	415,030,327	83,006,609	6	166,106,744	1	-	1
Linked TCGA-E	344,576,146	57,429,904	7	84,403,422	1	-	1
Linked TCGA-A	35,329,868	5,782,962	383	8,329,393	23	251.3k	0.99

2.2 Large Data Queries

The large data queries were designed to test the federation engines for real large data use cases, particularly in life sciences domain. These queries span over large datasets (such as Linked TCGA-E, Linked TCGA-M) and involve processing large intermediate result sets (usually in hundreds of thousands) or lead to large result sets (minimum 80459) and large number of endpoint requests. Consequently, we will see in the evaluation that the query processing time for large data queries exceeds one hour. In order to collect real queries with these characteristics, we contacted different domain experts and obtained a total of 8 large data queries to be included in our benchmark. Further details given in table .

[1] http://cancergenome.nih.gov/
[2] http://tcga.deri.ie/

Table 2: Linked TCGA query characteristics. (**#TP** = total number of triple patterns in a query, **#RS** = distinct number of relevant source. The values inside brackets show the total number of distinct sources used in the SPARQL 1.1 version (using SPARQL SERVICE clause) of each of the benchmark queries. the minimum number of distinct sources required to get the complete result set, **#Results** = total number of results), **#JV** = total number join vertices, **MJVD** = mean join vertices degree, **MTPS** = mean triple pattern selectivity, **MBRTPS** = Mean BGP-restricted triple pattern selectivity, **MJRTPS** = mean join-restricted triple pattern selectivity, **UN** = UNION, **OP** = OPTIONAL, **DI** = DISTINCT, **FI** = FILTER, **LI** = LIMIT, **OB** = ORDER BY, **RE** = Regex, **NA** = not applicable since there is no join node in the query, **-** = no SPARQL clause used. **Avg.** = the average values across the individual queries categories, i.e., simple, complex, and large data.

Query	Join Vertices	#TP	#RS	#Results	#JV	MJVD	MTPS	MBRTPS	MJRTPS	Clauses
L1	4 Path	6	3(2)	227192	4	2	0.192	0.48437	0.00001	UN
L2	1 Path,1 Hybrid	6	3(2)	152899	2	3.5	0.286	0.15652	0.00098	DI, FI
L3	2 Path,1 Hybrid	7	3(2)	257158	3	3	0.245	0.07255	0.07205	FI, OB
L4	2 Path,2 Hybrid	8	4(2)	397204	4	2.5	0.305	0.38605	0.00008	UN, FI, RE
L5	1 Star,1 Path,1 Sink,2 Hybrid	11	4(3)	190575	5	3	0.485	0.39364	0.00367	FI
L6	1 Star,1 Path,1 Sink,2 Hybrid	10	4(2)	282154	5	2.8	0.349	0.23553	0.00298	FI, DI
L7	2 Path,1 Hybrid	5	13(2)	80460	3	2.33	0.200	0.26498	0.00007	DI, FI
L8	2 Path,2 Hybrid	8	3(2)	306705	4	2.5	0.278	0.33376	0.00001	UN, FI

2.3 Performance Metrics

Previous works [4, 5, 7] suggest that the following six metrics are important to evaluate the performance of federation engines: (1) the total number of triple pattern-wise (TPW) sources selected during the source selection, (2) the total number of SPARQL ASK requests submitted to perform (1), (3) the completeness (recall) and correctness (precision) of the query result set retrieved, (4) the average source selection time, (5) the average query execution time, In addition, we also show the results of the data sources index/data summaries generation time and index compression ratio (i.e., index to dataset ratio). However, they are not applicable to index-free approaches such as FedX [7]. Previous work [5] show that an overestimation of triple pattern-wise sources selected can greatly increase the overall query execution time. This is because extra network traffic is generated and unnecessary intermediate results are retrieved, which are excluded after performing all the joins between query triple patterns. The time consumed by the SPARQL ASK queries during the source selection is directly added into the source selection time, which in turn is added into the overall query execution time.

3 Evaluation

In this section, we evaluate state-of-the-art SPARQL query federation systems by using both SPARQL 1.0 and SPARQL 1.1 versions of LargeRDFBench queries. We first describe our experimental setup in detail. Then, we present our evaluation results. All data used in this evaluation can be found on the benchmark homepage.

3.1 Experimental Setup

We used Virtuso triplestore for loading the benchmark datasets. To avoid server bottlenecks, we started the two largest endpoints (i.e., Linked TCGA-E and Linked TCGA-M) in machines with high processing capabilities. To

minimise the network latency we used a dedicated local network. We conducted our experiments on local copies of Virtuoso (version 7.1) SPARQL endpoints with number of buffers 1360000, maximum dirty buffers 1000000, number of server threads 20, result set maximum rows 100,000,000,000 and maximum SPARQL endpoint query execution time of 6000,000,000 seconds.

All experiments (i.e., the federation engines themselves) were run on a separate Linux machine with a 2.70GHz i7 processor, 8 GB RAM and 500 GB hard disk. We used the default Java Virtual Machine (JVM) initial memory allocation pool (Xms) size of 1024MB and the maximum memory allocation pool (Xmx) size of 4096MB. Each query was executed 10 times and results were averaged. The query timeout was set to 2.5 hours (9×10^6 ms). Furthermore, the query runtime results were statistically analyzed using Wilcoxon Signed Rank Test (WSRT), a non-parametric statistical hypothesis test used when comparing two related samples. We chose this test because it is parameter-free and, unlike a t-test, it does not assume a particular error distribution in the data. For all the significance tests, we set the p-value to 0.05.

Federated Query Engines

We compared five SPARQL endpoint federation engines (versions available as of October 2015) – FedX [7], SPLENDID [2], ANAPSID [2], FedX+HiBISCuS [5], CostFed – on all of the 8 benchmark queries. Note that HiBISCuS [5] is only a source selection approach and FedX+HiBISCuS is the HiBISCuS extensions of the FedX federation engines. To the best of our knowledge, the five systems we chose are the most state-of-the-art SPARQL endpoint federation engines [5]. Of all the systems, only ANAPSID and CostFed perform *join-aware* Triple Pattern-Wise Source Selection (TPWSS). The goal of the *join-aware* TPWSS is to select those data sources that actually *contribute* to the final result set of the query. This is because it is possible that a source contributes to the triple pattern but its results may be excluded after performing a join with the results of another triple pattern.

FedX [7] is an index-free SPARQL query federation system which completely relies on SPARQL ASK queries and a cache (which store the most recent ASK request) to perform TPWSS. This query is forwarded to all of the data sources and those sources which pass the SPARQL ASK test are selected. The result of each SPARQL ASK test is then stored in cache to be used in future. Thus before sending a SPARQL ASK request to a particular data source, a cache lookup is performed. A bind (vectored evaluation in nested loop) join is used for the integration of sub-queries results. We consider two setups for FedX. We evaluated both FedX(cold) and FedX(100%) setups of FedX. The former setup displays the characteristics of FedX with its cache empty and the latter means that cache contains all the information necessary for TPWSS. Thus in later setup, no SPARQL ASK request is used for TPWSS. Consequently, the former setup represents the worst case and the later setup represents the best case scenario.

SPLENDID [2] is an index-assisted approach which makes use of VoiD descriptions as index along with SPARQL ASK queries to perform the TPWSS. A SPARQL ASK query is used when either predicate is unbound (e.g., $< s >$?p $< o >$) or any of the subject (e.g., $< s >< p >$?o) or object (e.g., ?s $< p >< o >$) of the triple pattern is bound. Both bind and hash joins are used for integrating the sub-queries result and a dynamic programming strategy [8] is used to optimize the join order of SPARQL basic graph patterns.

ANAPSID [1] is an index-assisted adaptive query engine that adapts its query execution schedulers to the data availability and runtime status of SPARQL endpoints. ANAPSID performs a heuristic-based source selection presented in its extension [3]. The Adaptive Group Join (based on the Symmetric Hash Join and Xjoin operators) and Adaptive Dependent Join (adjoin) which extends the dependent join operator are used for integrating the sub-queries result.

CostFed source selection (based on HibIScUS[5]) is the index-assisted hyper graph based triple pattern-wise source selection approach for SPARQL endpoint federation systems. It intelligently makes use of the hypergraph

representation of SPARQL queries and URI's authorities[3] to perform TPWSS. The query planner is based on cost estimation for different joins.

3.2 Experimental Results

Index Construction Time and Compression Ratio

Table 3 shows a comparison of the index/data summaries construction time and the compression ratio[4] of the selected approaches. A high compression ratio is essential for fast index lookups during source selection and query planning. FedX does not rely on an index and makes use of a combination of SPARQL ASK queries and caching to perform the whole of the source selection steps it requires to answer a query. Therefore, these two metrics are not applicable for FedX. As pointed out in [5], ANAPSID only stores the set of distinct predicates corresponding to each data source. Therefore, its index generation time and compression ratio are better than that of CostFed and SPLENDID on our benchmark.

Table 3: Comparison of index construction time, compression ratio, and support for index update. (NA = Not Applicable).

	FedX	SPLENDID	ANAPSID	CostFed
Index Gen. Time(min)	NA	190	6	92
Compression Ratio(%)	NA	99.998	99.999	99.998
Index update?	NA	✗	✗	✓

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 4 shows the results of these three metrics for the selected approaches. The optimal number of sources were calculated by looking manually into the intermediate results for relevant sources and selecting those sources which contribute to the final result set.

Overall, ANAPSID is the most efficient approach in terms of total TPW sources selected, CostFed is the most efficient in terms of total number of SPARQL ASK requests used, and FedX (100% cached) is the fastest in terms of source selection time (see Table 4). 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. Still, FedX (100% cached) clearly overestimates the set of capable sources by more than half to the optimal (474 in FedX vs. 229 optimal). FedX (100% cached) is clearly outperformed by ANAPSID and CostFed. FedX (100% cached)'s poorer performance is due to FedX only performing TPWSS while both CostFed and ANAPSID perform *join-aware* TPWSS. As mentioned before, such overestimation of sources can be very costly because of the extra network traffic and irrelevant intermediate results retrieval. The effect of such overestimation is consequently even more critical while dealing with large data queries. For large data queries CostFed is not able to skip many sources. This is because the approach makes use of the different URI authorities to perform source pruning [5]. However, most of the large data queries come from Linked TCGA with single URI authority (i.e., `tcga.deri.ie`). Hence, HiBISCuS tends to

[3]URI syntax: `http://tools.ietf.org/html/rfc3986`
[4]Compression ratio = 100*(1 - index size/total data dump size)

Table 4: Comparison of the 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.

	FedX				SPLENDID			ANAPSID			CostFed			Optimal
Query	#T	#AR	SST	SST*	#T	#AR	SST	#T	#AR	SST	#T	#AR	SST	#T
L1	14	78	282	**5**	14	52	720	**6**	10	260	14	**0**	124	6
L2	10	78	279	**7**	10	13	230	**6**	5	142	10	**0**	94	6
L3	10	91	314	**9**	10	26	314	**7**	5	146	11	**0**	99	7
L4	18	104	321	**7**	18	**0**	198	**8**	8	338	16	**0**	80	8
L5	21	143	400	**5**	21	26	277	**12**	31	10255	20	**0**	130	11
L6	20	130	419	**4**	20	26	298	**10**	52	13173	18	**0**	160	10
L7	20	65	320	**6**	20	13	240	**6**	7	1822	9	**0**	270	5
L8	20	104	366	**7**	20	52	700	**9**	17	404	20	**0**	170	8

overestimate the number of sources in this case. On the other hand, ANAPSID makes use of SPARQL ASK requests combined with SSGM (Star Shaped Group Multiple Endpoints) [3] to skip a large number of sources. However, SPARQL ASK queries are expensive compared to local index lookups, as performed in HiBISCuS.

Completeness and Correctness of Result Sets

Two systems can only be compared to each other if they provide the same results for a given query execution. Table 5 shows the federation engines and the corresponding Linked TCGA queries for which complete and correct results were not retrieved by at least one of the systems. It is important to note that the correct results for all benchmark queries were obtained by loading all the data sources into a single virtuoso triple store and executing the query (a no more federated query) over it. We have not included L8 since every system either timed out or resulted in runtime error, hence the results completeness and correctness cannot be determined in this case. CostFed clearly outperforms other systems in this evaluation. Interestingly, none of the systems is able to provide complete and correct results. The incomplete results generated by federation systems can be due to a number of reasons, e.g., their join implementation, the type of network [4], the use of an outdated index or cache or even endpoint restrictions on the maximum result set sizes. However, in our evaluation we always used an up-to-date index and cache, there was no restriction on SPARQL endpoints maximum result set sizes, and a dedicated local network. Thus, the sole reason (to the best of our knowledge) for the systems at hand not providing complete/correct result is the existence of flaws in the implementation of joins or various SPARQL constructs such as `FILTER`, `REGEX`, etc. For example (as discussed further in the next section), FedX possibly give zero results for L2, L3, and L5 due to a flaw in the `FILTER` implementation.

Query Execution Time

The query execution time has often been used as the key metric to compare federation engines. Table 6 show the query execution time of the selected approaches for Linked TCGA queries. The query execution time was

Table 5: Result set completeness and correctness: Systems with incomplete precision and recall. (**RE** = Runtime error, **TO** = Time out)

System	FedX			SPLENDID			ANAPSID			FedX+HiBISCuS			CostFed		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
L1	TO	TO	TO	1	0.03	0.06	1	0.16	0.28	TO	TO	TO	1	1	1
L2	0	0	0	TO	TO	TO	TO	TO	TO	0	0	0	1	1	1
L3	0	0	0	TO	TO	TO	TO	TO	TO	0	0	0	1	1	1
L4	TO	TO	TO	0	0	0	0	0	0	1	0.48	0.65	1	1	1
L5	TO	TO	TO	RE	RE	RE	TO	TO	TO	0	0	0	RE	RE	RE
L6	TO	TO	TO	RE	RE	RE	TO	TO	TO	0	0	0	RE	RE	RE
L7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

Table 6: Runtimes (in ms) on Linked TCGA queries with all Virtuoso endpoints. The values inside the brackets show the percentage of the actual query results obtained.(**TO** = Time out after 2.5 hour, **RE** = runtime error).

Qr.	FedX (cold)	FedX (warm)	SPLENDID	ANAPSID	FedX+HiBISCuS	CostFed
L1	TO (7.2 %)	TO (7.2 %)	123735 (2.73 %)	19672 (15.76 %)	TO (7.2 %)	1237000 (100 %)
L2	35 (0 %)	35 (0 %)	45473 (1.8 %)	TO (0 %)	76 (0 %)	454709 (100 %)
L3	27 (0 %)	27 (0 %)	4877696 (100 %)	TO (0 %)	47 (0 %)	4877991 (100 %)
L4	TO (0.08 %)	TO (0.08 %)	7535531 (0 %)	8775598 (0 %)	62595 (48.34 %)	7535200 (100 %)
L5	TO (0 %)	TO (0 %)	RE (0 %)	TO (0 %)	TO (0 %)	RE (0 %)
L6	TO (0 %)	TO (0 %)	RE (0 %)	TO (0 %)	6127090 (0 %)	RE (0 %)
L7	122633 (100 %)	122500 (100 %)	114456 (100 %)	105447 (100 %)	119449 (100 %)	114400 (100 %)
L8	TO (0.01 %)	TO (0.01 %)	TO (0.05 %)	TO (0.05 %)	TO (0.01 %)	TO (0.05 %)

calculated once all the results were retrieved from the result set iterator. Overall, our engine is able to retrieve complete results for 5 out of 8 queries within the timeout limit. Other systems are only able to retreive complete results for 1 or maximum 2 queries.

The most important finding for *large data queries* is that no system can produce complete results for all of the 8 queries. This shows that the current *implementation* of query planning strategies (i.e., bushy trees in ANAPSID, left-deep trees in FedX, and dynamic programming [8] in SPLENDID) and join techniques (i.e., adaptive group and dependent join in ANAPSID, bind and nested loop in FedX, and bind, hash in SPLENDID) in the selected systems is still not mature enough to deal with large data. In addition, we have found that queries terminating within the timeout limit and returning zero results might possibly be caused by a flaw in the `FILTER` implementation. For example, FedX and its HiBISCuS extension give zero results for queries L2, L3, and L5 and send a single endpoint request (ref., **??**) for each of these queries. All of these queries contain a `FILTER` clause. However, we found that FedX and its HiBISCuS extension are able to retrieve results by removing the `FILTER` clause and setting the `LIMIT=1` in these queries. We also noticed that for queries with incomplete results (e.g, L1, L4, L8 etc.), FedX and its HiBISCuS extension send a large number of endpoint requests and quickly get some initial results. After that the engines stop sending endpoint requests until the timeout limit is reached. This may be due to some memory leak or possible deadlock in the query execution portion of FedX. CostFed able to give complete results for 5/8 large data queries, the highest in comparison to other systems. The query L4 is executed by ANAPSID, SPLENDID, SPLENDID+HiBISCuS within the timeout limit with zero results.

While running large data queries, we found that Virtuoso imposes a limit[5] of maximally 2^{20} = 1,048,576 on the maximum number of rows returned as HTTP response. This means that a federation engine based on Virtuoso may end up returning incomplete results if it results for a sub-query with a result set size larger than 1,048,576 rows. To ensure that our results were not tainted by this technical limitation of Virtuoso, we have analyzed all the endpoint requests (given in Table **??**) sent by each of the federation engines for each of the LargeRDFBench queries. Our study showed that SPLENDID sends at least one endpoint request with result size greater than 1,048,576 rows. The rest of the federation engines do not send endpoint requests with result size greater than this limit. Given that the endpoints requests with answer set sizes beyond 2^{20} rows were sent exclusively to the 3 Linked TCGA (i.e. Linked TCGA-A, Linked TCGA-E, Linked TCGA-E) datasets, we reran our benchmarking experiments with all federation engines on large data queries (i.e., L1-L8) after replacing the Virtuoso servers for these 3 datasets with FUSEKI[6] servers. Note that the FUSEKI triple store does not have such limit on the maximum number of results returned in response to a query. In this series of experiments, SPLENDID times out with a recall of 0 (i.e., no results generated) for the queries L1, L2 and L4. The other federation engines return results comparable to those presented in Table 6.

4 Conclusion

We presented an evaluation of the state-of-the-art federation engines with CostFed based on Linked TCGA use-case data and queries. The evaluation shows the superiority of the proposed engine. However, there is still further improvements possible, since our engine was able to retrieve complete results for 5 out of 8 selected queries.

In the future, we want to devise a query planner based on Deep Reinforcement Learning (DRL) with feedback system.

[5]This Virtuoso problem is now solved. See `https://github.com/openlink/virtuoso-opensource/issues/700`
[6]FUSEKI: `https://jena.apache.org/documentation/serving_data/`

References

[1] Maribel Acosta, Maria-Esther Vidal, Tomas Lampo, Julio Castillo, and Edna Ruckhaus. ANAPSID: an adaptive query processing engine for SPARQL endpoints. In *ISWC*, 2011.

[2] Olaf Görlitz and Steffen Staab. Splendid: Sparql endpoint federation exploiting void descriptions. In *COLD at ISWC*, 2011.

[3] Gabriela Montoya, Maria-Esther Vidal, and Maribel Acosta. A heuristic-based approach for planning federated sparql queries. In *COLD*, 2012.

[4] Gabriela Montoya, Maria-Esther Vidal, Oscar Corcho, Edna Ruckhaus, and Carlos Buil-Aranda. Benchmarking federated sparql query engines: are existing testbeds enough? In *ISWC*, pages 313–324. 2012.

[5] Muhammad Saleem and Axel-Cyrille Ngonga Ngomo. HiBISCuS: Hypergraph-based source selection for sparql endpoint federation. In *ESWC*, 2014.

[6] Muhammad Saleem, Shanmukha S Padmanabhuni, Axel-Cyrille Ngonga Ngomo, Aftab Iqbal, Jonas S Almeida, Stefan Decker, and Helena F Deus. TopFed: TCGA tailored federated query processing and linking to LOD. *JBMS*, 2014.

[7] Andreas Schwarte, Peter Haase, Katja Hose, Ralf Schenkel, and Michael Schmidt. Fedx: Optimization techniques for federated query processing on linked data. In *ISWC*, 2011.

[8] P. Griffiths Selinger, M. M. Astrahan, D. D. Chamberlin, R. A. Lorie, and T. G. Price. Access path selection in a relational database management system. In *SIGMOD*, pages 23–34, 1979.