Benchmarking Commercial RDF Stores with Publications Office Dataset

Ghislain Atemezing[1] and Florence Amardeilh[1]

Mondeca, 35 Boulevard de Strasbourg, 75010, Paris, France.

Abstract. This paper presents a benchmark of RDF stores with real-world datasets and queries from the EU Publications Office (PO). The study compares the performance of four commercial triple stores: Stardog 4.3 EE, GraphDB 8.0.3 EE, Oracle 12.2c and Virtuoso 7.2.4.2 with respect to the following requirements: bulk loading, scalability, stability and query execution. The datasets and the selected queries (44) are used in the Linked Data publication workflow at PO. The first results of this study provides some insights into the quantitative performance assessment of RDF stores used in production environment in general, especially when dealing with large amount of triples. Virtuoso is faster in querying and loading scenarios while GraphDB shows better results regarding stability.

Keywords: benchmarking, triple stores, RDF, semantic web, knowledge management, SPARQL, Stardog, GraphDB, Enterprise RDF store

1 Introduction

The adoption of semantic technologies for data integration has evolved in the recent years, thus the use of triple stores as back-end for data publishers has become popular. Triple stores stability and robustness are then critical in production environment where many thousands of users need to have access to fresh data, such as the case of the Publications Office (PO)[1]. In this article, we propose a quantitative analysis of four popular commercial triple stores (POSB); Stardog, GraphDB, Oracle 12c and Virtuoso. The initial list of triple stores also included Blazegraph and Neo4j. The latter were not included in this final article for two reasons: (i) It was not possible to load the dataset with Neo4j and (ii) Although we succeeded in loading the dataset with Blazegraph, we were not satisfied with the high number of time-outs (15) for the first set of queries, and the vendor was taking too long to answer technical inquiries. The benchmark is based on queries that are actually issued by PO employees to manage and create applications using their own dataset related to legal publications in all the languages of the European Union. This benchmark is motivated by PO requirements to evaluate and document the advances on triple stores performance compared

[1] `https://publications.europa.eu`

to their current solution adopted some years ago. The results show that a real-world SPARQL benchmark gives some indicators for comparing enterprise-based triple stores and provide insightful output for their quantitative-based analysis . The comparison of our results with other benchmark studies confirms that the performance of triple stores is not always homogeneous and depends on many parameters such as the types of queries, the characteristic of the datasets and more importantly the underlying hardware set up. Triple stores are used to store knowledge bases in RDF for data management and data web applications. The W3C SPARQL recommendation [10] is the vendor-independent query language to build more effective queries. It is clearly important for data integration applications to assess the performance of triple store implementations. Publications Office produces and publishes legal information and official journal of the European Union, in more than 20 languages in RDF. Currently, Virtuoso is part of their publishing architecture. After some years, PO wants to reevaluate the current state of the art of triple stores with respect to their use case. This paper contributes to give an overview of the status of Virtuoso compared to other commercial triple stores regarding bulk loading, benchmarking and stability. The results presented here are subsets of criteria needed by a data producer to assess the strengths and weaknesses of RDF triple stores. To mitigate the Virtuoso-Bias, we asked to other vendors if they could provide us with reformulated queries. Unfortunately, only Oracle gave us some recommendations that we implemented and reported as "oracle 12c optimized." This paper[2] presents the results of the evaluation of four popular commercial RDF stores conducted in 2017: Virtuoso, Stardog, Oracle 12c and GraphDB using the datasets from the PO and 44 queries used daily in production. The fraction of the commercial RDF stores covered in this paper is a subset of an initial list of seven triple stores including Marklogic, Blazegraph and Neo4J. PO suggested the list of the RDF stores for the benchmark. The results for Marklogic 8 are not presented because we did not have all the results at the time of writing the paper. It does not intend to be yet another benchmark, but rather a quantitative assessment by a data publisher for evaluating existing commercial triple stores. The paper describes a general purpose benchmarking with real datasets and queries by looking at bulk loading time, stability test and multi-client benchmark for 20 queries from the "instantaneous queries". The first results show that Virtuoso and Stardog are faster in bulk loading, while Virtuoso outperforms respectively to GraphDB, Stardog and Oracle in query-based performance. GraphDB shows to be the winner in the stability test performed in this benchmark.

The rest of the paper is structured as follows: Section 2 presents an overview of the ontology and the datasets. Section 3 describes the selected queries and analyses the features form and Basic Graph Pattern (BGP)[3] involve in their construction. Section 4 describes the set up of the benchmark, followed by Section

[2] We would like to highlight that the conclusions reached are exclusively those of the authors, not necessarily those of the Publications Office.

[3] `https://www.w3.org/TR/rdf-sparql-query/#BasicGraphPatterns`

5 and discussions. Section 7 presents some related works and Section 8 concludes the paper and highlights future work.

2 PO Semantic Datasets

This section presents an overview of the ontology used to model the datasets at PO, with an analysis of the nquads dataset used for the benchmarking.

2.1 Ontology

The Common Metadata Model (CDM) is the ontology used by the Publications Office to generate data in RDF . CDM[4] is an ontology based on the Functional Requirements for Bibliographic Records (FRBR) model described in RDF(S)/OWL to represent the relationships between the resource types managed by the PO and their views according to the `FRBR` model in terms of Work, Expression, Manifestation and Item.

2.2 Datasets

Two original datasets are used for the loading experiment, a normalized dataset with 2,195 nquads files representing a dump of the production environment [11], and a non-normalized dataset from 64 nquads files. PO uses "normalization" to replace all subjects by URIs to avoid the use of `owl:sameAs` pragma in Virtuoso. We loaded 727,442,978 triples from the normalized dataset, while 728,163,464 triples from the non-normalized dataset. For querying, we also add the CDM ontology and the Named Authority Lists (NAL). The Name Authority List (NAL) are SKOS [7] concepts representing categories such as events, countries, organizations, treaties, etc. In the production dataset (PROD data), 187 `CDM` classes are instantiated, which represents 60.71% of the ontology. Additionally, 198 distinct object properties are present in the dataset.

Furthermore, the dataset contains around 4,958,220 blank nodes. This number is quite high as it implies the presence of 7 blank nodes on every 1,000 triples.

2.3 Generated Datasets

We generated extra datasets based on original data to perform scalability test during the loading process. We implemented a script to generate the new datasets without modifying the structure of the initial data. Our algorithm postfixes all the resources of the type `<http://publications.europa.eu/resource/cellar>` by new ones of the form `<http://publications.europa.eu/resource/cellar/$i/gen/g>` where $i was incremented according to the desired size. We generated datasets for 2 billion (2Bio) and 5 billion (5Bio) triples respectively.

[4] `http://publications.europa.eu/mdr/cdm/`

3 Query description and analysis

The queries used in this benchmark were received from the employees of PO working directly in the publication workflow of RDF data. Additionally the current endpoint used to publish the dataset at PO is Virtuoso, which means the queries are optimized for Virtuoso. We identified two categories of queries based on the goal achieved and the expected response time:

- Instantaneous queries[5]: These queries are generally used to dynamically generate dynamic visualizations on the website. Thus, they should be faster. In this group, we got 20 queries, divided into SELECT(16), DESCRIBE(3) and CONSTRUCT(1). Figure 1 depicts the Basic Graph Pattern (BGP) count per query.
- Analytical queries: These queries are used for validation and mapping purposes, where the most important feature is the quality of the results, not only the time to answer the query. In a total of 24 validation and mappings queries, 100% are SELECT queries. Table 1 depicts the number of BGP detected in each query of this category.

Table 2 shows the use of the 4 SPARQL query forms, i.e., SELECT, DESCRIBE, ASK, and CONSTRUCT in the whole set of queries used in POSB.

Fig. 1. BGP repartition of SPARQL queries in Category 1.

Query	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
BGP	**2**	**2**	**2**	**5**	**7**	**6**	**7**	**7**	**7**	**7**	**11**	**17**	**12**	**3**	**1**	**4**	**17**	**3**	**11**	**3**

For each of the query in the two categories, we analyze the features in the SPARQL query syntax. Table 3 and 4 depicts the forms used in the queries where column GRBY is GROUP BY, GRPC is GROUP CONCAT and ORBY is ORDER BY. Queries in category 2 contain less combination of SPARQL features, with only two queries Q13 and Q14 with 8 different number of features.

We define the notion of "Feature Mix per Query (FMpQ)" which is the number of distinct form of a query. The maximum number is set to be 14. As described in [13], the most important constructs are the following: UNION, DISTINCT, ORDERBY, REGEX, LIMIT, OFFSET, OPTIONAL, FILTER and GROUPBY.

Based on the above definition, we obtained 7 queries (IQ5, IQ6, IQ7, IQ8, IQ9, IQ10 and IQ19) with more than 7 number of features. They might be "slow" according to the threshold in the average response time set for each category as presented in Section 5.2. Query IQ10 contains the highest number of features, without UNION, VALUES and ORDER BY.

[5] `https://github.com/gatemezing/posb/tree/master/bench/queries/category1`

Table 1. BGP repartition of SPARQL queries in Category 2.

Query	#BGP	Query	#BGP
AQ1	1	AQ13	6
AQ2	1	AQ14	7
AQ3	2	AQ15	7
AQ4	5	AQ16	1
AQ5	5	AQ17	1
AQ6	4	AQ18	7
AQ7	5	AQ19	1
AQ8	4	AQ20	1
AQ9	1	AQ21	6
AQ10	15	AQ22	1
AQ11	16	AQ23	1
AQ12	2	AQ24	1

Table 2. SPARQL query forms detected for the PO SPARQL benchmark (POSB).

Query Form	Total	Percentage (%)
SELECT	40	90.9%
DESCRIBE	3	6.81%
CONSTRUCT	1	2.27%
ASK	-	-

Table 3. Queries form detected in instantaneous queries. The last column shows the total number of distinct element forms for each query.

ID	DISTINCT	FILTER	OPTIONAL	UNION	LANG	REGEX	STR	LIMIT	OFFSET	GRBY	VALUES	GRPC	ORBY	**FMpQ**
IQ1	-	-	X	-	-	-	-	-	-	-	X	-	-	2
IQ2	-	-	X	-	-	-	-	-	-	-	X	-	-	2
IQ3	-	-	X	-	-	-	-	-	-	-	X	-	-	2
IQ4	-	X	-	X	X	-	-	-	-	-	X	-	-	4
IQ5	X	X	X	-	-	-	-	X	X	X	-	X	X	8
IQ6	X	-	X	-	-	-	-	X	X	X	-	X	X	7
IQ7	X	-	X	-	-	-	-	X	X	X	-	X	X	7
IQ8	X	X	X	-	-	-	-	X	X	X	-	X	X	8
IQ9	X	-	X	-	-	-	-	X	X	X	-	X	X	7
IQ10	X	X	X	-	X	X	X	X	X	X	-	X	-	10
IQ11	X	-	-	-	-	-	-	-	-	-	-	-	X	2
IQ12	X	X	X	-	-	X	-	-	-	-	-	X	X	6
IQ13	X	X	X	-	-	X	-	-	-	-	-	-	X	5
IQ14	X	X	X	-	-	X	-	-	-	-	-	-	-	4
IQ15	X	-	X	-	-	-	-	-	-	-	-	-	-	2
IQ16	-	X	X	-	-	-	X	-	-	-	X	-	X	5
IQ17	X	X	X	-	-	X	-	-	-	-	-	X	X	6
Q18	-	X	X	-	-	X	-	-	-	-	-	-	X	4
IQ19	X	X	X	X	-	X	-	-	-	X	-	-	X	7
IQ20	-	X	X	-	-	X	-	-	-	-	-	-	-	3

4 Experimental Setup

This section presents the setup we used for benching four triple stores commonly used in production environment. We first describe the triple stores and their configuration, followed by our experimental strategy and finally the results. The experiments were conducted on a server with the following characteristics: (i) Model: DELL PowerEdge R730; processor: Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz , 6C/12T; (ii) RAM: 128 GB DDR4 ECC; (iii) Disk capacity: 4 TB SATA ; RAID: Dell PERC H730p, (Raid 0/1) and (iv) Operating System:

Table 4. Queries form detected in analytical queries with the FMpQ for each query.

ID	DISTINCT	FILTER	OPTIONAL	UNION	LANG	REGEX	STR	LIMIT	OFFSET	GRBY	VALUES	GRPC	ORBY	**FMpQ**
AQ1	X	X	-	-	-	X	X	-	-	-	-	-	-	4
AQ2	X	X	-	-	-	X	X	-	-	-	-	-	-	4
AQ3	X	X	X	-	-	X	X	-	-	X	-	-	-	6
AQ4	-	X	X	-	-	-	-	-	-	-	-	-	-	2
AQ5	-	X	X	-	-	-	-	-	-	-	-	-	-	2
AQ6	-	X	X	-	-	-	-	-	-	-	-	-	-	2
AQ7	-	X	X	-	-	X	X	-	-	-	-	-	-	4
AQ8	X	X	-	-	-	X	X	-	-	-	-	-	-	4
AQ9	X	X	-	-	-	X	X	-	-	-	-	-	-	4
AQ10	X	X	X	-	X	-	X	-	-	-	-	X	-	6
AQ11	X	X	X	-	X	-	X	-	-	-	-	-	-	5
AQ12	X	X	X	-	-	-	-	-	-	-	-	X	X	5
AQ13	X	-	X	X	-	-	-	X	X	X	-	X	X	8
AQ14	X	-	X	X	-	-	-	X	X	X	-	X	X	8
AQ15	X	X	X	-	-	X	X	-	-	-	-	-	X	6
AQ16	X	X	-	-	-	X	-	-	-	-	-	-	X	4
AQ17	X	X	-	-	-	X	-	-	-	-	-	-	X	4
AQ18	X	X	X	-	-	X	X	-	-	-	-	-	X	6
AQ19	X	X	-	-	-	X	-	-	-	-	-	-	X	4
AQ20	X	X	-	-	-	X	-	-	-	-	-	-	X	4
AQ21	-	X	X	-	-	X	X	-	-	-	-	-	-	4
AQ22	X	X	-	-	-	X	-	-	-	-	-	-	X	4
AQ23	X	X	-	-	-	X	-	-	-	-	-	-	X	4
AQ24	X	X	-	-	-	-	-	-	-	-	-	-	-	3

CentOS 7, 64 bits and Java 1.8.0 running. The system settings follow the best practices recommended by the vendors and double checked by the experts team. The material is available online at `https://github.com/gatemezing/posb/`.

4.1 Triples Store setup

We carried out our experiments using Virtuoso [5], Stardog [15], Oracle [9] and GraphDB [2]. The configuration and the version of each triple store were the following:

- Virtuoso: Open-Source Edition version 7.2.4: We set the following memory-related parameters: NumberOfBuffers = 5450000, MaxDirtyBuffers = 4000000.
- Stardog: Stardog Enterprise Edition version 4.2.3. We set the Java heap size to 16GB and MaxDirectMemorySize to 8GB. We deactivate the strict parsing option and use the default SL reasoning inference during the loading process of dataset.
- GraphDB: GraphDB Enterprise Edition version 8.0.3. We use a configuration file with entity index size set to 500000000 with entity predicate list enabled, disabling the content index. We use two different rulesets: one empty and the other set to "rdfs-optimized".
- Oracle: Oracle 12.2 database. We set the following parameters in the pfile.ora file: pga_max_size set to 2G , pga_aggregate_limit set to 64G and pga_aggregate_target set to 32G. The configurations are available online for further exploitation.[6].

[6] `https://github.com/gatemezing/posb/tree/master/config`

4.2 Query validation

For avoiding parsing errors with other triple stores, the first step before launching the bench tool is the validation the queries with the Jena ARQ tool [7] . The command to parse is: *qparse –query query.rq*. This step aims at providing with standardized SPARQL queries to be used across different RDF stores.

4.3 Benchmark Execution

The benchmark starts once the datasets are loaded into the RDF stores. Each run of loading the dataset is performed in a unique process running on the server. The benchmark comprises the following steps:

1. **Configuration step**: We set in the corresponding configuration file the timeout value for the queries. This forces the store to abort or kill the process running the query.
2. **Warm-up step**: In order to measure the performance of a triple store under operational conditions, a warm-up phase is used. In the warm-up phase, query mixes are posed to the triple store. We used a warm-up set to 20, meaning that we run 20 times the set of queries in a given category before starting the run phase.
3. **Hot-run step**: During this phase, the benchmark query mixes were sent to the tested store. We keep track of each run and output the results in a CSV file containing the statistics. We perform 5 runs in this stage, adding also the timeout value similar to the corresponding configuration file setup of the RDF store. We also set the max delay between query is set to 1000s.

5 Results

In the section we present the results of the loading process for the datasets, according to the settings described in Section 4.

5.1 Bulk loading

We found the best configuration for the bulk loading with two specificities in Stardog and GraphDB. In the latter, we load in a repository with inference set to RDFS optimized ruleset while in the former, we remove the strict parser option. Figure 2 shows the overall time taken by each RDF store, with the fastest being the one with less time. In GraphDB, the `rdfs-optimized`[8] ruleset is an optimized version of RDFS with the support of subClassOf and related type inference subPropertyOf, symmetric, inverse and transitive properties.

Figure 2 depicts the performance time in hours taken by the four RDF stores. Regardless of data volume, the fastest RDF store to load the datasets is Virtuoso,

[7] `https://jena.apache.org/`
[8] `http://graphdb.ontotext.com/documentation/enterprise/reasoning.html`

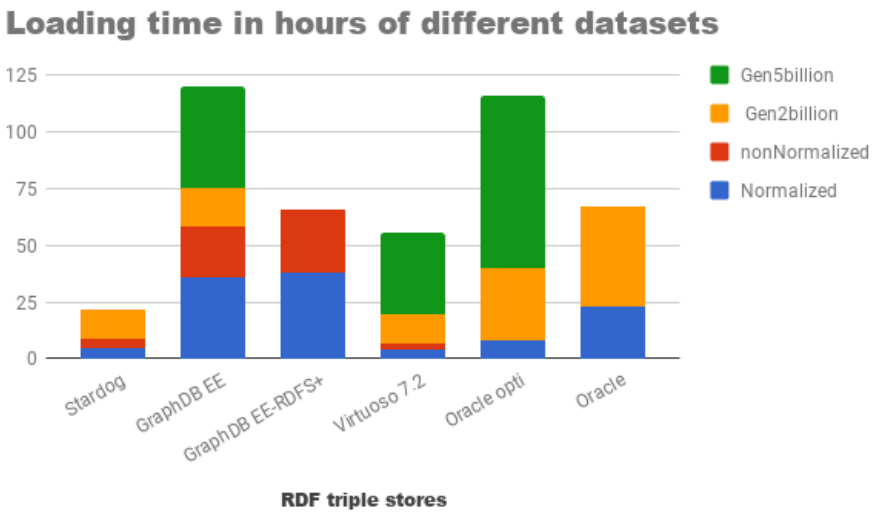

Fig. 2. Performance time during bulk loading of PO datasets and generated ones up to 5Billion triples

followed closely by Stardog. Oracle optimized is slower compared to GraphDB for loading 2Bio and 5Bio datasets. Stardog failed to load 5Bio dataset. The loading process with GraphDB of 2Bio and 5Bio uses a different loader (8.0.6-SNAPSHOT), optimized by the support team for this specific task. Regarding the bulk loading of PROD dataset, Virtuoso and Stardog are very closed and are faster than the two other stores, achieving all the loads in less than 5 hours. This result is useful in case of a database corruption with a need to reload the dataset from scratch within a reasonable exploitation time frame.

5.2 Benchmarking

We use the Sparql Query Benchmarker tool[9], an open-source tool based on Jena, with 20 runs per categories to warm up the server in a mix queries order, with an actual benchmark with 5 runs. The timeout settings are 60s for instantaneous queries and 600s for analytical queries. Furthermore we use three datasets, each of them loaded in a dedicated named graph: (i) the normalized dataset, (ii) the NAL dataset and (iii) the CDM ontology. We use the methodology presented in [12] to perform the benchmark and gather the results output in CSV files for further analysis.

[9] `https://github.com/rvesse/sparql-query-bm`

Benchmarking instantaneous queries We proceed to compare the results of querying the RDF stores using the benchmarker tool. We gathered the output of each run in CSV files.

Table 5. Runtime, standard deviation and QMpH when benchmarking instantaneous queries for PROD dataset

RDF Store	Average Runtime (s)	Standard Deviation	QMpH
Virtuoso 7.x	1.10	0.01	3,246.82
GraphDB EE	240.29	0.01	14.98
GraphDB RDFS+	240.30	0.02	14.98
Stardog 4.x	253.00	5.99	14.23
Oracle 12c	269.73	2.15	13.34
Oracle 12c optimized	112.75	.59	31.92

Single Thread Virtuoso is faster compared to GraphDB and Stardog. The last two RDF stores have similar number of QMpH values as shown in Table 5. There were no time outs when querying Virtuoso. However, two queries timed out with Oracle (IQ3 and IQ9), three queries timed out (IQ3, IQ10 and IQ19) with Stardog, while four queries timed out with GraphDB (the same as Stardog plus IQ4). The optimized queries for Oracle permits to double the speed of the queries. Table 6 presents the average runtime execution for queries in category 1.

Query	Virtuoso	Stardog	GraphDB	GraphDBRDFS+	Oracle
IQ1	**.02**	.03	**.02**	.04	.10
IQ2	.02	**.01**	.03	.04	.07
IQ3	**.02**	60	60	60	60
IQ4	**.009**	.35	60	60	.04
IQ5	.09	.82	**.01**	**.01**	31.35
IQ6	.28	.10	**.01**	**.01**	39.35
IQ7	.06	**.01**	**.01**	**.01**	34.82
IQ8	.10	1.35	**.01**	**.01**	31.88
IQ9	.05	.20	**.01**	**.01**	60
IQ10	**.12**	60	60	60	3.64
IQ11	**.004**	.01	.02	.006	.11
IQ12	.06	49.17	.05	**.04**	2.93
IQ13	.03	**.01**	**.01**	**.01**	.14
IQ14	**.01**	5.52	**.01**	**.01**	19.53
IQ15	**.006**	5.28	.009	.009	.08
IQ16	.01	1.50	.01	**.008**	.02
IQ17	.03	.06	**.01**	**.01**	.25
IQ18	**.06**	1.64	.07	.07	.09
IQ19	.11	60	60	60	**.04**
IQ20	**.006**	.02	.009	.008	.15

Table 6. Average response time in second per queries and RDF stores in the case of instantaneous queries

All the queries in Virtuoso finished in less than 1s. IQ3 timed out for GraphDB and Stardog, while it finishes in 0.02s with Virtuoso. GraphDB shows also slower query time except when the queries timed out. Query IQ4 timed out with GraphDB, but faster with Virtuoso. Stardog presents 3 timed out and slower than GraphDB and Virtuoso. There were 6 queries in category 1 that finished in more than 1s (IQ8,IQ12,IQ14,IQ15,IQ16,IQ18). IQ19 timed out for GraphDB and Stardog, while it finishes in 0.11s with Virtuoso. Oracle takes particularly long time in average execution for queries IQ5-IQ9; IQ11 and IQ13 compared to the rest of the RDF stores.

RDF Store	5clients	20clients	50clients	70clients	100clients
Virtuoso 7.2.4.2	367.22	358.27	371.76	354.60	341.02
GraphDB EE	2.13	2.12	2.13	2.12	2.13
GraphDB EE RDFS+	2.13	2.13	2.13	2.36	2.13
Stardog 4.3	1.973	1.94	1.97	1.96	1.95
Oracle 12c	2.10	1.99	2.01	2.01	2.02

Table 7. QMpH values in multi-threading bench for instantaneous queries

Multi-Threading Table 7 presents the results of QMpH values in case of multithread benchmark for instantaneous queries. Virtuoso performs again by far better, follows by GraphDB and Oracle. In general Virtuoso performs 10x slower than in the single thread, compared to GraphDB and Oracle which is only 7x slower. The results also shed lights on the likely constant behavior of GraphDB and Oracle when it comes to concurrent access, one of a key criteria for SPARQL endpoints usage.

Bench for analytical queries We set 600s for timeout because the queries in this category are more analytical-based queries. Virtuoso is faster than all the other triple stores (cf. Table 8). Stardog reveals 4 timed out queries (AQ15, AQ16, AQ19 and AQ22), 1 timed out query (AQ10) for Oracle. No timed out in GraphDB even when reasoning in RDFS+ activated in the repository. This shows that inferences can be activated in this category of queries in GraphDB without altering the performance of the queries.

Virtuoso is faster than all the rest of the triple stores as shown in Table 9 with the values of QMpH. The slowest query AQ10 involves `DISTINCT`, `FILTER` and `OPTIONAL`. However, this same query is 4x slower on GraphDB, 20x slower on Stardog and timed out in Oracle. Also, the well performing query AQ20 involves the DISTINCT feature.

GraphDB is the second fastest triple store for this set of queries. The slowest query AQ10 involves `DISTINCT`, `FILTER` and `OPTIONAL` features. Also, the fastest query AQ20 involves `DISTINCT`.

Four analytical queries timed out with Stardog (AQ15, AQ16, AQ19 and AQ22) whereas no query timed out with GraphDB and Virtuoso. All the queries

Table 8. Average response time in second per queries and RDF stores in analytical queries

Query	Virtuoso	Stardog	GraphDB	GraphDBRDFS+	Oracle 12c
AQ1	10.72	**.16**	5.96	7.31	4.24
AQ2	9.87	**9.66**	9.94	12.30	29.13
AQ3	33.52	**15.84**	26.38	31.74	590.37
AQ4	1.65	15.42	.01	**.009**	.09
AQ5	**.677**	12.16	20.89	15.09	39.78
AQ6	.66	22.29	**.15**	.16	.27
AQ7	**.01**	26.27	**.01**	**.01**	.52
AQ8	**.04**	27.11	.48	.51	14.22
AQ9	**.02**	.11	1.7	1.96	.46
AQ10	**16.97**	314.39	60.05	65.56	600
AQ11	**.602**	57.51	2.92	2.43	2.42
AQ12	**7.15**	14.52	60.02	65.64	82.95
AQ13	.06	26.28	**.01**	**.01**	85.19
AQ14	.06	.10	**.01**	**.01**	206.308
AQ15	.72	600	.06	**.05**	3.74
AQ16	.01	600	.01	**.009**	.09
AQ17	.01	14.87	.01	**.008**	.06
AQ18	.06	292.24	.01	**.009**	2.93
AQ19	.01	600	.008	**.007**	.62
AQ20	.01	16.34	.008	**.007**	.60
AQ21	.02	23.00	.02	**.01**	.98
AQ22	.01	600	.01	**.008**	.70
AQ23	.01	15.70	.01	**.007**	.68
AQ24	.03	45.73	**.01**	.012	1.33

Table 9. Runtime, standard deviation and QMpH when benchmarking analytical queries

RDF Store	Avg. Runtime (s)	Std. Deviation	QMpH
Virtuoso 7.2.4.2	44.86	0.14	80.23
GraphDB EE RDFS+	134.95	3.98	36.67
GraphDB EE	180.50	1.02	19.94
Stardog 4.3	4000.26	1226.11	0.89
Oracle 12c	1490.53	67.88	2.41

that timed out in Stardog involved `FILTER`. Additionally, AQ15 also involves `OPTIONAL`. This indicates that using complex `FILTER` in conjunction with additional `OPTIONAL` might affects the overall runtime of the query.

5.3 Stability Test

We perform a stress test on the triple stores to have a quantitative indication related to stability. For this purpose, the queries of category 1 are run continuously under a progressively increasing load to see how the system reacts to high load. The test starts by specifying the number of parallel clients in the script. Each client completes the run of the mix queries in parallel. The number of parallel clients is then multiplied by the ramp up factor which defaults to 2 and the process is repeated. This repeats until either the maximum runtime or the maximum number of threads are reached. We set the maximum runtime to 180min (3h) and set the maximum parallel threads to 128.

Table 10. Results of the stress test on triple stores using instantaneous queries.

RDF Store	#mix runs	#op. run	Max.//.threads	# Errors
Stardog	92	1,840	128	576
Virtuoso	255	5,100	256	4,732
GraphDB	255	5,100	256	139
Oracle	63	1,260	128	1,009

The results in Table 10 show that Stardog and Oracle finished with the limit of the parallel threads. On the other hand, Virtuoso and GraphDB have completed the test after 180 min, reaching 256 parallel threads. In this scenario, GraphDB shows fewer errors compared to Virtuoso. Based on the total execution errors encountered for this stress experiment, we can conclude that GraphDB is likely to be more stable respectively in this order to Stardog, Oracle and Virtuoso.

6 Discussion

The results indicate that no single RDF store is the absolute winner across different queries. For instantaneous queries, Stardog and GraphDB timed out for queries with REGEX, DISTINCT, FILTER, OPTIONAL and GROUP BY, whereas Oracle was able to finish without time out. However, Oracle timed out in IQ9 containing OPTIONAL and DISTINCT. GraphDB performs better in analytical queries, while having timed out in instantaneous queries. Stardog and Oracle are the only RDF stores not able to have "zero time-outs" according to our settings, and sometimes extremely slow for some queries involving complex `FILTER` and `OPTIONAL`.

Regarding the bulk loading, Virtuoso and Stardog are faster than the other systems. This has an advantage in case of crashes with a need to reload the whole dataset from scratch within a normal working day time frame. Also, we are aware that the figures obtained in this benchmark might have different values if SSD disks were used instead in our server. However, we argue that this benchmark gives an overview of the current state of the triple stores based on the PO datasets.

In Table 12, we present the time for querying those queries with a combination of at least three of the SPARQL forms REGEX, DIST, FLT, OPT and GRBY. Stardog timed out in AQ15, and took almost 5min for AQ18.

In summary, this benchmark shows the following insights:

- Overall Virtuoso is the fastest system followed by Stardog in the bulk loading scenario.
- No single system is unique winner across all queries. For instantaneous queries, Virtuoso is faster in 9 queries, followed by GraphDB in 8 queries. Stardog is faster in 3 queries. Oracle is faster in 1 query for which Stardog and GraphDB timed out.

Table 11. Comparison results time execution (in second) of the seven instantaneous queries with at least seven SPARQL features and BGP.

Query	Virtuoso	GraphDB	Stardog	Oracle	#FMpQ	#BGP
IQ5	.09	.01	.82	31.35	8	7
IQ6	.28	.01	.10	39.35	7	6
IQ7	.06	.01	.01	34.82	7	7
IQ8	.10	.01	1.35	31.88	8	7
IQ9	.05	.01	.20	>60s	7	7
IQ10	.12	>60s	>60s	3.64	10	7
IQ19	.11	>60s	>60s	.04	7	11

Table 12. Comparison results of analytical queries with a combination of at least three of the SPARQL forms REGEX, DIST, FLT, OPT and GRBY.

Query	Virtuoso	GraphDB	Stardog	Oracle	#FMpQ	BGP
AQ13	0.06	0.01	26.28	85.19	8	6
AQ14	0.06	0.01	0.10	206.308	8	7
AQ15	0.72	0.06	>600	3.74	6	7
AQ18	0.06	0.01	292.24	2.93	6	7

– In analytical queries, Virtuoso is the fastest system in not more than 25% of the queries, while GraphDBRDFS+ is the fastest in almost 42% of the queries.

7 Related Work

In the literature, several general purpose RDF benchmarks were developed on both artificial data and real datasets. The Lehigh University Benchmark (LUBM) [6] uses small number of classes, with plain SPARQL queries. The dataset generated are for the university domain. In LUBM, each query is executed 10 times and the average response time of the query is reported. In the publication domain, the SP2Bench [14] benchmark uses a synthetic test data and artificial queries. However it uses a very limited number of triples (25M) compared to the dataset used in this work.

The Berlin SPARQL Benchmark (BSBM) [3] applies a use case on e-commerce in various triple stores. It tests the SPARQL features on the triple stores, with the particularity of simulating operations performed by a human user. However, BSBM data and queries are artificial, with very limited classes and properties.

The DBpedia SPARQL Benchmark (DBPSB) [8] is another more recent benchmark for RDF stores. Its RDF data is DBPedia with up to 239M triples, starting with 14M to compare the scalability. It uses 25 heterogeneous queries. The procedure for benchmark includes the query log mining, clustering and

SPARQL feature analysis for the popular triple stores Virtuoso, Sesame, Jena-TDB and BigOWLIM. POSB shares in common with DBPSB the relative high number of classes, real-world dataset and queries. However, POSB differs in having more heterogeneity in the dataset, the selected queries are SPARQL queries used in production using very large mixture of SPARQL features. Also the benchmark does not assess the comparison in multi-client scenario. The overall results with Virtuoso the fastest store with DBpedia dataset converge with our findings, albeit the diverse set of queries. Our work differs from the above in that the goal is to evaluate four commercial triple stores with dataset in production at PO, with given set of real-world queries not generated from logs. The Waterloo SPARQL Diversity Test Suite (WatDiv) [1] addresses the stress testing of five RDF stores for diverse queries and varied workloads. WatDiv introduces two classes of query features (structural and data-driven) to evaluate previous four benchmarks: LUBM, BSBM, SP2Bench and DBPSB. The systems evaluated two prototype stores (RDF-3X and gStore) and three industrial systems (Virtuoso 6.1.8, Virtuoso 7.1.0 and 4Store). Their results and findings highlight good results of Virtuoso compared to other RDF system, although remarking that one system may win in one query and timeout in another. This latter remark also applies to our findings for Stardog, GraphDB and Oracle. Recently, Iguana framework [4] provides with a configurable and integrated environment for executing SPARQL benchmark. It also allows a uniform comparison of results across different benchmarks. However, we use for this benchmark a different tool and plan to use Iguana to better mitigate the virtuoso-bias in the input queries and have comparable results with previous benchmarks only with commercial triple stores.

8 Conclusions and Future Work

We have presented in this paper a comparison of four RDF stores Virtuoso, GraphDB Stardog and Oracle, with real datasets and SPARQL queries from Publications Office. The results highlighted that according to our settings, Virtuoso performs better in loading and benching queries in both categories, with a total of 44 queries tested. GraphDB handles more queries mix per hour than Stardog, but shows more timed out queries (instantaneous queries). Inversely, Stardog loads datasets faster than GraphDB, but was unable to succeed "zero time-outs" during the benchmark, showing some weakness in handling queries with complex `FILTER` and `OPTIONAL`. This study helps assessing RDF triple stores with datasets with similar characteristics than the one discussed here. However, any comparison depends on the type of queries, the server settings to decide on which RDF store to use in production.

It should be noted that this work has helped to improve the bulk loading mechanism of GraphDB and Oracle and the query engine of Stardog that will be available in their respective future major releases. We plan to add to this benchmark other RDF stores such as Marklogic 8 and Neptune[10], as well as

[10] `http://blog.mondeca.com/2018/02/09/requetes-sparql-avec-neptune/`

to include SPARQL update queries and qualitative features. We are convinced that such studies shed lights to publishers willing to create enterprise knowledge graph to easily assess which RDF store can fit their needs. Also, we plan to find the impacts of using different storage in the back-end such as binary RDF (e.g., HDT) and client side approach with Linked Data fragments.

Acknowledgments. We would like to thank all the interviewees at the Publications Office for their valuable input. We also thank Oracle, Ontotext and Stardog Union for all their support.

References

1. G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee. Diversified stress testing of rdf data management systems. In *International Semantic Web Conference*, pages 197–212. Springer, 2014.
2. B. Bishop, A. Kiryakov, D. Ognyanoff, I. Peikov, Z. Tashev, and R. Velkov. Owlim: A family of scalable semantic repositories. *Semantic Web*, 2(1):33–42, 2011.
3. C. Bizer and A. Schultz. Benchmarking the performance of storage systems that expose sparql endpoints. *World Wide Web Internet And Web Information Systems*, 2008.
4. F. Conrads, J. Lehmann, M. Saleem, M. Morsey, , and A.-C. Ngonga Ngomo. IGUANA: A generic framework for benchmarking the read-write performance of triple stores. In *International Semantic Web Conference (ISWC)*, 2017.
5. O. Erling and I. Mikhailov. Rdf support in the virtuoso dbms. In *Networked Knowledge-Networked Media*, pages 7–24. Springer, 2009.
6. 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.
7. A. Isaac and E. Summers. Skos simple knowledge organization system. *Primer, World Wide Web Consortium (W3C)*, 2009.
8. M. Morsey, J. Lehmann, S. Auer, and A.-C. Ngonga Ngomo. DBpedia SPARQL Benchmark—Performance Assessment with Real Queries on Real Data. In *ISWC 2011*, 2011.
9. Oracle. Oracle 12 manual, 2017. http://docs.oracle.com/database/122/index.htm.
10. E. Prud'hommeaux and A. Seaborne. Sparql query language for rdf. w3c recommendation, 2008.
11. O. Publications and Mondeca. Dump of rdf dataset used for rdf benchmark, 2017. http://doi.org/10.5281/zenodo.1036739.
12. Revelytix. Triple store evaluation - performance testing methodology. Technical report, Revelytix, Inc, 2010.
13. M. Saleem, Q. Mehmood, and A.-C. Ngonga Ngomo. Feasible: A feature-based sparql benchmark generation framework. In *Proceedings of the 14th International Conference on The Semantic Web - ISWC 2015 - Volume 9366*, pages 52–69, New York, NY, USA, 2015. Springer-Verlag New York, Inc.
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. Stardog. Stardog 4.2.x: The manual, 2017. https://docs.stardog.com/.

