RDF Stores Performance Test on Servers with Average Specification

Nikola Nikolić, Goran Savić, Milan Segedinac, Stevan Gostojić, Zora Konjović
University of Novi Sad, Faculty of Technical Sciences, Novi Sad, Serbia
{nikola.nikolic, savicg, milansegedinac, gostojic, ftn_zora}@uns.ac.rs

***Abstract* — The paper analysis the performances of different RDF stores on servers with average hardware specification. For this purpose, various tests have been performed on three RDF stores, namely Jena-Fuseki, Sesam-OWLIM and Virtuoso. Using different data sets and queries, the tests have measured CPU usage, heap memory consumption and execution time. Based on the results, for different application scenarios, an appropriate RDF store has been suggested.**

I. INTRODUCTION

Most applications that are based on semantic web store their data within RDF stores [1]. So far, various RDF stores have been developed providing different characteristics and performances. Depending on a particular application and its usage scenario, it must be decided which RDF store to use in order to satisfy both functional and non-functional requirements.

Many software applications do not contain complex functionalities implying that they do not store data sets larger than one million triplets. Executing queries over such smaller RDF stores doesn't require powerful and expensive servers. This paper is focused on such applications and is trying to propose an appropriate RDF store for non-expensive servers with average features.

The paper presents tests performed on three commonly used RDF stores. The tests have been ran within custom-made client application that executes various SPARQL queries to an RDF store. RDF stores have been set up on a server with average characteristics where RAM memory does not exceed 8 GB, CPU has up to 4 cores with disk drive space not larger than 500 GB.

Based on widely recognized RDF store ranking [2], we have chosen to test Virtuoso and Jena-Fuseki open source solutions and a free Lite version of commercial RDF store Sesame-OWLIM.

The following text has been organized as follows. The next section gives a short overview of similar performance tests. Section 3 describes our tests providing details about machine configuration used RDF stores, data sets and SPARQL queries. Measured parameters and testing procedure are also explained in this section. Results are presented and analysed in the section 4. Finally, the last section concludes the paper giving the future directions of this research.

The result of this testing is two-fold: on one hand, the proposal of the appropriate RDF store for a particular case of usage, and on the other hand comparison of performances of most commonly used RDF stores.

II. RELATED WORK

According to W3C list of references [3], several RDF benchmarks have been performed so far. These benchmarks mostly test large data sets by executing queries on powerful servers.

The most popular is Berlin SPARQL benchmark [4], which supports testing of several RDF stores, such as Sesame, Virtuoso, Jena-TDB, BigData and BigOwlim. It is based on a generic data set that is a part of an e-commerce use case. The data set contains a set of products, offered by different vendors and consumers, which post reviews about products. The performances have been measured on different size of data sets using various SPARQL queries. Data sets size varies from 10 million up to 150 billion triplets. Tests were performed on capable server machines worth up to ~70,000€ [5]. Given that such server highly exceeds hardware limitations set for our research, results of these tests cannot be used in the analysis conducted in this paper. Still, we have used the same testing procedure and SPARQL queries as Berlin SPARQL benchmark.

Another popular benchmark is SP^{2}Bench SPARQL benchmark performed on its own data sets, which are based on library scenarios. The benchmark uses smaller data sets consisting of up to a million triplets. In contrast to Berlin SPARQL benchmark, SP^{2}Bench evaluates performances of a single RDF store with variable RDF schemas [6].

The last benchmark we present in this paper is *Lehigh University Benchmark* (LUBM). The tests were performed on data sets that contain data on university publications. Successive batches of the same queries were used with some minor data variations. Such testing procedure does not represent real life scenarios where an application must response to a wide set of different queries.

The remaining SPARQL benchmarks listed in [3] do not cover all testing parameters that are relevant for our research (these parameters are described in Section 3, part E).

III. TEST

This section describes tests we have performed within this research.

A. Test machine

The tests have been run on a server with following features:

- Processor: Intel i5-3470 3.2GHz
- RAM: 8GB DDR3
- HDD: 500GB SATA3 7200rpm
- Operating system: Linux-Ubuntu 14.04.1 LTS 64-bit

In addition, one of the tested RDF stores works only with data loaded into working memory (*In-Memory Backend*). To make the results comparable, we have configured all data stores using *In-Memory* setup.

B. RDF stores

Following RDF stores have been used:

1. Virtuoso – Version 6.1.8,
2. Sesame-OWLIM – Version OWLIM-Lite 5.4.6486, based on Sesame 2.7.13, deployed on Apache-Tomcat 6.0.41.
3. Jena-Fuseki – Version 1.1.1, open source RDF store

Virtuoso [7] is free and open source software product which is one of the most commonly used RDF system worldwide. It is a universal hybrid server for handling data such as RDF triplets and XML documents. Using Virtuoso it is possible to combine SPARQL and SQL queries for handling RDF data.

Sesame-OWLIM is a commercial RDF store that offers a free version with limited features. This free version has been used in our study. It supports wide set of tools developed in Python and Java programming language. These tools provide increased RDF(S) functionalities. Sesame itself lacks OWL reasoner. To address this limitation, Sesame was upgraded with OWLIM third-party store [8] that adds missing functionalities. OWLIM belongs to newer generations of RDF stores which are made for more frequent data updating and increased concurrent access. Since beginning of 2014. the popularity of this RDF store has increased which drew our attention towards testing it. Free Lite version of OWLIM [9] is available with restricted features. One important constraint requires that data must be loaded in working memory.

Jena [10] is an open source software framework written in Java providing both storage and access of RDF data. It comes with its own OWL RDF graph reasoning component. It uses Fuseki SPARQL interface for accessing the abstract RDF graph model through HTTP protocol. Fuseki can be run as a stand-alone SPARQL server too.

C. Data sets

The tests use the same data sets as Berlin SPARQL benchmark. As mentioned, these data sets are taken from e-commerce domain containing sets of products that are classified by vendors and rated by reviewers. Data sets were programmatically generated in different sizes and representations depending on product count using BIBM (*Business Intelligence Benchmark*) generator [11].

Each data set was built from different class instances of vendors, producers, product offers, product types, product features, reviews, reviewers and their web pages. An example of a product class instance is shown in Listing 1.

```
dataFromProducer021:Product015
   rdfs:label "Dell Inspirion 3521";
   rdfs:comment "New machine";
   rdf:type ftn:Product;
   rdf:type ftn-inst:ProductType123;
   ftn:producer ftn-inst: Producer021;
   ftn:productFeature ftn-inst:ProductFeature456;
   ftn:productPropertyTextual1 "The best";
   ftn:productPropertyNumeric1 "17"^^xsd:Integer;
   dc:publisher dataFromProducer021:Producer021;
   dc:date "2015-01-07"^^xsd:date .
```

Listing 1. Product class instance

Each product is described with label, comment and product type. Product type defines different product features which are also described with label and comment. The product is produced by one or more vendors. A vendor is described with label, comment, web page URL and country URI. Offer is described with price, expiration and delivery date. Reviewers are described with name, e-mail address and nationality.

Table 1. shows the characterictics of the generated data sets. The data sets contain up to a million triplets. Table rows display number of class instances for the given number of expected triplets starting at 1K triplets and all the way up to 1M. Given that BIBM generator cannot generate the exact number of required triplets, the penultimate row display how many triplets have been generated. The last row presents the size of the file containing the data.

TABLE I.
CHARACTERISTICS OF DATA SETS

RDF triplets	1K	10K	100K	1M
Products	1	25	260	2848
Producers	1	1	6	61
Product Features	289	289	1954	4745
Product Types	7	7	37	151
Vendors	1	1	3	30
Offers	20	500	5200	56960
Reviewers	1	13	129	1451
Reviews	10	250	2600	28480
Exact RDF triplets	1844	10250	101817	1022446
File size (unzipped)	210.6kB	968.9kB	9.3MB	93.6MB

D. SPARQL queries

Data sets of all sizes have been tested using a combination of two groups of queries. First group gathers queries aimed on searching and navigation through the required product fragments. It includes 12 patterns [12], whereby the most important are:

1. Generic search for a given set of generic product properties.
2. More specific search for products with a given set of product properties.

3. Finding similar products of a given product.
4. Retrieving detailed information on several products.
5. Retrieving reviews for given products.
6. Getting background information about reviewers.
7. Retrieving offers for given products.
8. Checking information about vendors and their delivery conditions.

Second group of queries is designed to test independent analytical queries over the dataset. It includes 8 patterns [13]. These are:

1. The first 10 of most discussed product categories of products from a specific country which are based on number of reviews by reviewers from a certain country.
2. The first 10 products that are most similar to a specific product, rated by the count of features they have in common.
3. Products with the largest increase of interest (ratio of review counts) from one month to the next.
4. Feature with the highest ratio between price with that feature and price without that feature.
5. The most popular products of a specific product type for each country - by review count.
6. Reviewers who rated products by a specific Producer much higher than the average.
7. Products which are in first 1000 of most offered products of a certain product type that are not sold by vendors of a specific country
8. The top 10 cheapest vendors for a specific product type by the ratio of products below and above the average.

Both set of queries have been executed in random order using SPARQL protocol on chosen RDF stores.

E. Measuring parameters

RDF benchmarks explained in the Section 2 primarily measure these two variables:

- Time needed for loading and indexing of triplets
- Time needed for executing SPARQL queries

In our research we want to examine system performances in more details by measuring more parameters. Benchmark in this study is written in Java programming language. JvmTop [14] open source console application has been used for measuring the performances of RDF stores. For all running JVM (Java Virtual Machines) on a given system, JvmTop provides monitoring of following resources:

- Process ID
- Name of measured class
- Current heap memory usage depending on maximum allocated value
- Current non-heap memory usage depending on maximum allocated value
- CPU usage
- Percentage of garbage collector usage
- Number of infinite loops
- Number of created threads
- Thread state
- CPU usage by threads
- Number of blocking threads

All these variables were used to determine the system performance.

F. Method

Testing was done by implementing the following procedure for all data sets and RDF stores:

1. Load one data set in an RDF store
2. Execute first group of SPARQL queries
3. Execute second group of SPARQL queries
4. Save measured parameters results

Table 2. shows total number of executed queries per single data set. As mentioned, queries have been divided into two groups. Total of 15000 queries have been executed on each RDF store, where 10 000 queries belong to first group, while the remaining 5 000 queries belong to a second group of SPARQL queries.

TABLE II.
TOTAL EXECUTED SPARQL QUERIES

TOTAL EXECUTED QUERIES		
Data sets	Group 1	Group 2
1K, 10K, 100K	2500	1500
1M	2500	500

IV. RESULTS

This section presents the results of measuring four key parameters – CPU usage, heap memory usage, individual and total query execution time.

Table 3. shows CPU usage for all the tested RDF stores for a separate execution of queries on all data sets.

TABLE III.
CPU USAGE PER RDF STORE

RDF DB systems	Data sets	CPU USAGE [%]		
		MIN	MEAN	MAX
Jena-Fuseki	1K	0	31.32	103.47
	10K	0	27.19	105.85
	100K	0	28.66	400.00
	1M	0	21.54	100.00
Sesame-OWLIM Lite	1K	0	19.42	117.86
	10K	0	30.11	154.52
	100K	0	35.74	363.89
	1M	0	22.74	84.09
Virtuoso	1K	0	20.05	83.34
	10K	0	21.15	129.17
	100K	0	21.11	218.75
	1M	0	18.75	62.50

Results were shown in 3 columns representing minimum, maximum and mean value, expressed in percentages. We can notice that for Jena-Fuseki RDF store the maximum value reached as far as 400%. It corresponds to sum of usages of all loaded CPU cores. It can be noted that for data sets beginning at million triplets Virtuoso RDF store puts the lowest load on CPU.

Figures 1, 2, 3 and 4 show the charts of CPU usage in time during queries execution. The chart series are derived using 6th degree polynomial regression which by definition introduces certain error. It can be noticed that Sesame-OWLIM Lite store reached a slightly higher CPU load than the other two RDF stores.

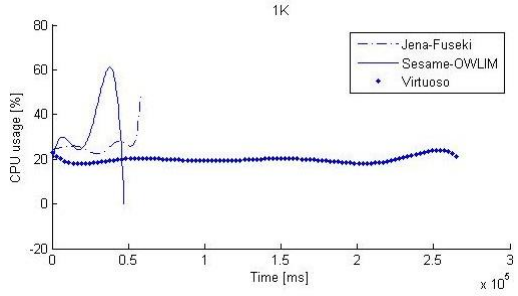

Figure 1. CPU usage for data set 1K

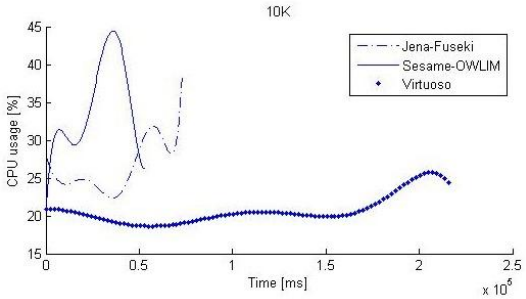

Figure 2. CPU usage for data set 10K

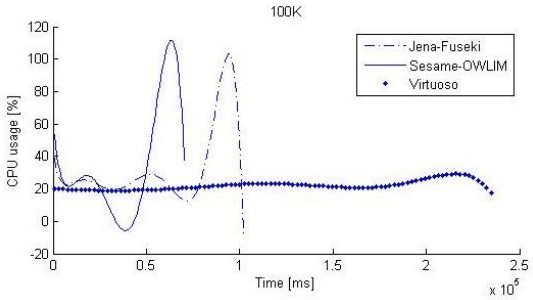

Figure 3. CPU usage for data set 100K

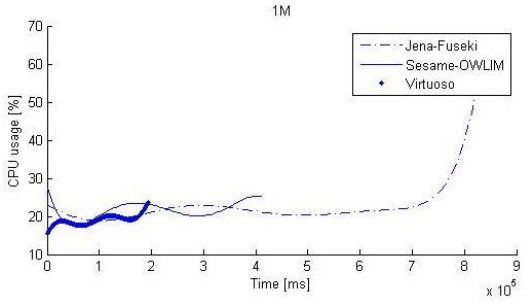

Figure 4. CPU usage for data set 1M

Given that all RDF stores use RAM memory to store the data, the measurement of heap memory usage has been necessary. Table 4. shows heap memory usage during queries execution over all data sets. If we analyse mean values, it can be noticed that Virtuoso occupies the least amount of heap memory on average.

TABLE IV.
HEAP USAGE PER RDF STORE

RDF DB systems	Data sets	HEAP USAGE [MB]		
		MIN	MEAN	MAX
Jena-Fuseki	1K	32.5	113.88	187
	10K	21.75	96.49	182
	100K	15.5	105.24	316.5
	1M	24	86.09	397
Sesame-OWLIM Lite	1K	9	69.93	175
	10K	17	104.61	178
	100K	16	108.12	292
	1M	25	86.46	396.5
Virtuoso	1K	10.25	51.69	171.5
	10K	17.25	64.82	175.5
	100K	13.5	70.31	176
	1M	20	67.67	173

Figures 5, 6, 7 and 8 show the charts of heap usage in time during queries execution.

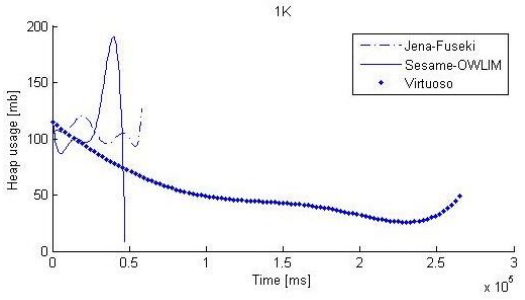

Figure 5. Heap usage for data set 1K

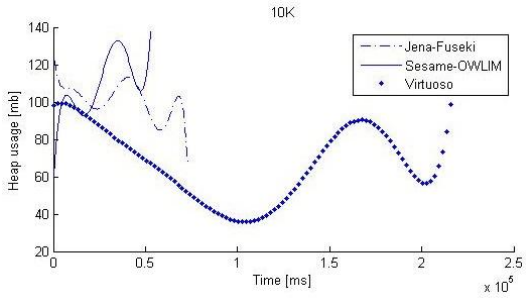

Figure 6. Heap usage for data set 10K

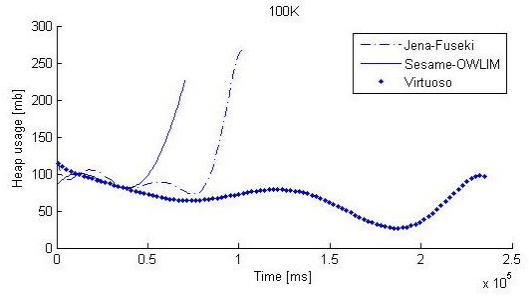

Figure 7. Heap usage for data set 100K

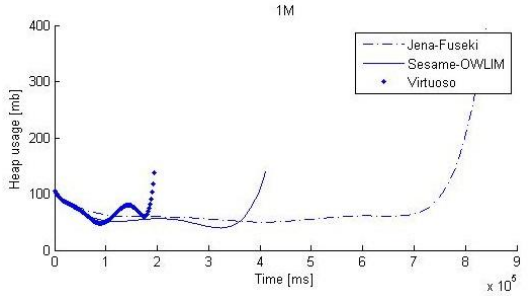

Figure 8. Heap usage for data set 1M

The third measured parameter is execution time of individual queries over all data sets. It is shown in Table 5.

TABLE V.
EXECUTE QUERY TIME PER RDF STORE

RDF DB systems	Data sets	EXECUTION TIME [ms]		
		MIN	MEAN	MAX
Jena-Fuseki	1K	3.08	12.08	1157.53
	10K	3.25	12.32	1114.64
	100K	3.01	20.41	1451.98
	1M	3.08	275.25	135709.74
Sesame-OWLIM Lite	1K	2.39	10.16	1081.49
	10K	2.31	9.10	1039.85
	100K	1.90	13.76	1057.26
	1M	2.30	136.28	65660.16
Virtuoso	1K	2.67	66.86	852.82
	10K	2.88	52.45	875.35
	100K	2.90	55.11	2300.94
	1M	3.04	58.59	3248.01

By observing mean values of execution queries, we can notice that Sesame-OWLIM Lite store takes considerably less time to execute SPARQL queries.

Figure 9, 10, 11 and 12 show the charts of individual query execution in time. Although Virtuoso has best performances in CPU and heap usage benchmarks, it doesn't show good results in execution time benchmark. However, for queries executed on data set of million triplets Virtuoso retains the lead.

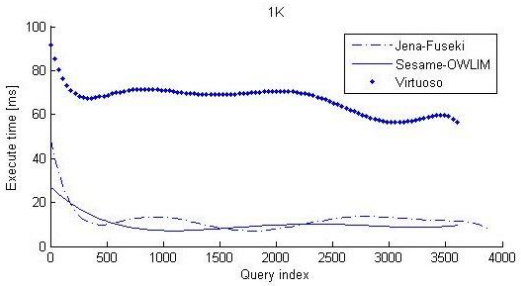

Figure 9. Execute time for data set of 1K triplets

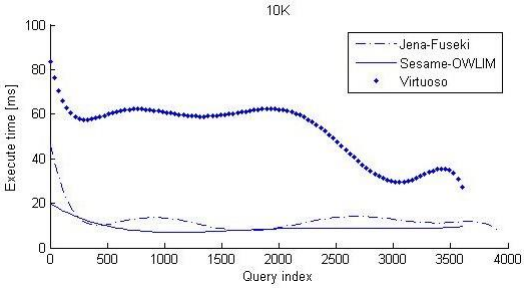

Figure 10. Execute time for data set of 10K triplets

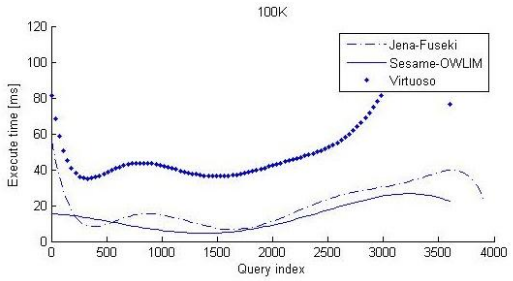

Figure 11. Execute time for data set of 100K triplets

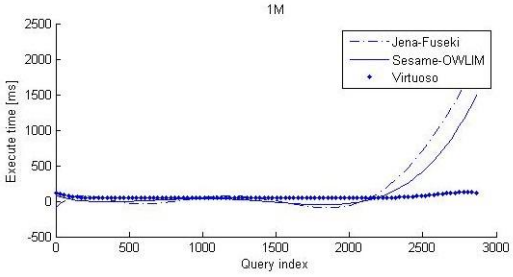

Figure 12. Execute time for data set of 1M triplets

Table 6 shows total execution time of both query groups on different data sets. Rows represent data sets, while columns represent RDF stores.

We can notice that Virtuoso store has notably better total execution time for data set of million triplets. Still, for data sets of less than one million triplets, Virtuoso store has by far worse total execution time. Sesame-OWLIM store has the best total execution time over data sets of less then million triplets.

TABLE VI.
TOTAL EXECUTION TIME

TOTAL EXECUTION TIME [s]			
Data sets	RDF DB systems		
	Jena-Fuseki	Sesame-OWLIM Lite	Virtuoso
1K	57.58	46.65	264.71
10K	72.36	52.07	215.27
100K	101.27	69.88	234.13
1M	838.89	410.01	192.91

In accordance with the results, we can determine which RDF store is the most appropriate for the particular scenario.

For data sets containing less than one million triplets we have shown that Virtuoso has better results in heap and CPU usage, but individual and total query execution time over such data sets is significantly greater than for Sesame-OWLIM Lite store.

For scenarios whose priority is the speed of query execution, we would recommend Sesame-OWLIM Lite store for data sets of less than one million triplets. However, for data sets greater than one million triplets, Virtuoso store is the most efficient regarding all parameters measured in this study.

V. CONCLUSION

In this paper we presented performance tests of commonly used RDF stores deployed on servers with average characteristics. The test results should facilitate the selection of RDF store for applications that do not work with data set larger than one million triplets. We tested Virtuoso and Jena-Fuseki as open source RDF stores, as well as Sesame-OWLIM as a free version of a commercial RDF store. The tests measured CPU and heap usage as well as time needed for query execution. The paper proposed recommendations for different scenarios, depending of the importance of the specific performance indicator.

Future research will be aimed on testing mentioned RDF data stores in native-storage mode where data is stored on a disk in contrast to the research presented in this paper where data is stored in RAM memory. In that case a commercial version of Sesame-OWLIM would be needed.

ACKNOWLEDGMENT

Results presented in this paper are part of the research conducted within the Grant No. III-47003, Ministry of Education, Science and Technological Development of the Republic of Serbia.

REFERENCES

[1] W3C RDF, *http://www.w3.org/RDF/*

[2] DB-Engines Ranking, *http://db-engines.com/en/ranking_definition*

[3] W3C RDF store benchmarking, *http://www.w3.org/wiki/RdfStoreBenchmarking*.

[4] C. Bizer and A. Schultz. "The Berlin SPARQL Benchmark.", Int. J. Semantic Web Inf. Syst., 5(2):1–24, 2009.

[5] Berlin BSBM benchmark machine, *http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/results/V7/index.html#machine*.

[6] M. Schmidt, T. Hornung, G. Lausen, C. Pinkel. "SP^{2}Bench: A SPARQL performance benchmark.", ICDE, pages 222–233. IEEE, 2009.

[7] Virtuoso – OpenLink Software, *http://virtuoso.openlinksw.com*.

[8] A. Kiryakov, D. Ognyanov; D. Manov, OWLIM – a Pragmatic Semantic Repository for OWL, WISE 2005, 20 Nov, New York City, USA.

[9] Sesame-OWLIM Lite RDF store, *http://owlim.ontotext.com/display/OWLIMv54/OWLIM-Lite+Fact+Sheet*

[10] Apache Jena-Fuseki RDF store, *http://jena.apache.org/documentation/*

[11] BSBM generator, *http://sourceforge.net/projects/bibm/*

[12] SPARQL queries pattern – set 1., *http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/spec/ExploreUseCase/index.html#queriesTriple*

[13] SPARQL queries pattern – set 2., *http://wifo5-03.informatik.uni-mannheim.de/bizer/berlinsparqlbenchmark/spec/BusinessIntelligenceUseCase/index.html#queriesTriple*

[14] JvmTop – Google code, *https://code.google.com/p/jvmtop/wiki/Documentation*