Benchmarking Virtuoso 8 at the Mighty Storage Challenge 2018: Training Results

Milos Jovanovik[1,2] and Mirko Spasić[1,3]

[1] OpenLink Software, United Kingdom
[2] Faculty of Computer Science and Engineering,
Ss. Cyril and Methodius University in Skopje, Macedonia
[3] Faculty of Mathematics, University of Belgrade, Serbia
{mjovanovik,mspasic}@openlinksw.com

Abstract. Following the success of Virtuoso at last year's Mighty Storage Challenge - MOCHA 2017, we decided to participate once again and test the latest Virtuoso version against the new tasks which comprise the MOCHA 2018 challenge. The aim of the challenge is to test the performance of solutions for SPARQL processing in aspects relevant for modern applications: ingesting data, answering queries on large datasets and serving as backend for applications driven by Linked Data. The challenge tests the systems against data derived from real applications and with realistic loads, with an emphasis on dealing with changing data in the form of streams or updates. Virtuoso, by OpenLink Software, is a modern enterprise-grade solution for data access, integration, and relational database management, which provides a scalable RDF Quad Store. In this paper, we present the training phase results for Virtuoso, for the MOCHA 2018 challenge. These results will serve as a guideline for improvements in Virtuoso which will then be tested as part of the main MOCHA 2018 challenge.

Keywords: Virtuoso, Mighty Storage Challenge, MOCHA, Benchmarks, Data Storage, Linked Data, RDF, SPARQL

1 Introduction

Last year's Mighty Storage Challenge, MOCHA 2017, was quite successful for our team and Virtuoso – we won the overall challenge [5, 9]. Building on that, we intend to participate in this year's challenge as well, in all four challenge tasks: (i) RDF data ingestion, (ii) data storage, (iii) versioning and (iv) browsing. The Mighty Storage Challenge 2018[4] aims to provide objective measures for how well current systems perform on real tasks of industrial relevance, and also help detect bottlenecks of existing systems to further their development towards practical usage. This arises from the need for devising systems that achieve acceptable performance on real datasets and real loads, as a subject of central importance for the practical applicability of Semantic Web technologies.

[4] https://project-hobbit.eu/challenges/mighty-storage-challenge2018/

2 Virtuoso Universal Server

Virtuoso Universal Server[5] is a modern enterprise-grade solution for data access, integration, and relational database management. It is a database engine hybrid that combines the functionality of a traditional relational database management system (RDBMS), object-relational database (ORDBMS), virtual database, RDF, XML, free-text, web application server and file server functionality in a single system. It operates with SQL tables and/or RDF based property/predicate graphs. Virtuoso was initially developed as a row-wise transaction oriented RDBMS with SQL federation, i.e. as a multi-protocol server providing ODBC and JDBC access to relational data stored either within Virtuoso itself or any combination of external relational databases. Besides catering to SQL clients, Virtuoso has a built-in HTTP server providing a DAV repository, SOAP and WS* protocol end-points and dynamic web pages in a variety of scripting languages. It was subsequently re-targeted as an RDF graph store with built-in SPARQL and inference [2, 3]. Recently, the product has been revised to take advantage of column-wise compressed storage and vectored execution [1].

The largest Virtuoso applications are in the RDF and Linked Data domains, where terabytes of RDF triples are in use – a size which does not fit into main memory. The space efficiency of column-wise compression was the biggest incentive for the column store transition of Virtuoso [1]. This transition also made Virtuoso a competitive option for relational analytics. Combining a schemaless data model with analytics performance is an attractive feature for data integration in scenarios with high schema volatility. Virtuoso has a shared cluster capability for scaling-out, an approach mostly used for large RDF deployments.

A more detailed description of Virtuoso's triple storage, the compression implementation and the translation of SPARQL queries into SQL queries, is available in our paper from MOCHA 2017 [9].

3 Evaluation

In this section, we present our preliminary results for all the tasks in the challenge, based on the training data available on the project website and the benchmark parameters for the training phase specified by the tasks' organizers. For this purpose, we used a local deployment of the HOBBIT platform[6].

Task 1 - RDF Data Ingestion: The aim of this task is to measure the performance of SPARQL query processing systems when faced with streams of data from industrial machinery in terms of efficiency and completeness. This benchmark, called ODIN (StOrage and Data Insertion beNchmark), increases the size and velocity of RDF data used, in order to evaluate how well can a system store streaming RDF data obtained from the industry. The data is generated from one or multiple resources in parallel and is inserted using SPARQL INSERT queries. At some points in time, SPARQL SELECT queries check the triples that

[5] `https://virtuoso.openlinksw.com/`
[6] `http://hobbit_demo.openlinksw.com/`

are actually inserted and test the systems ingestion performance and storage abilities [4].

Results: We tested our system, Virtuoso 8.0 Commercial Edition, against ODIN as part of the training phase. The Virtuoso configuration parameters are available at Github[7]. The task organizers specified the benchmark parameters for this phase and the values of these parameters are shown in Table 1, while the achieved KPIs for our system are presented in the Table 2.

Table 1: ODIN Configuration.

Parameter	**Value**
Duration	600000
Mimicking algorithm	TRANSPORT_DATA
Output folder	output_data/
Number of DG - agents	1
Insert queries per stream	5
Number of TG - agents	1
Population of gen. data	50
Seed	123

Table 2: ODIN KPIs for Virtuoso 8.0.

KPI	**Value**
Avg. Delay of Tasks (in s)	0.2246
Macro-Average-F-Measure	0.9494
Macro-Average-Precision	0.9219
Macro-Average-Recall	0.9786
Micro-Average-F-Measure	0.9546
Micro-Average-Precision	0.9292
Micro-Average-Recall	0.9813
Maximum Triples/s	6473.7

Task 2 - Data Storage: This task uses the Data Storage Benchmark (DSB) and its goal is to measure how data storage solutions perform with interactive, simple, read, SPARQL queries as well as complex ones, accompanied with a high insert data rate via SPARQL UPDATE queries, in order to mimic real use-cases where READ and WRITE operations are bundled together. It also tests systems for their bulk load capabilities [6].

Results: The benchmark parameters for the test phase are shown in Table 3, and the achieved KPIs for our system are presented in Table 4.

Task 3 - Versioning RDF Data: The aim of this task is to test the ability of versioning systems to efficiently manage evolving datasets, where triples are added or deleted, and queries evaluated across the multiple versions of said datasets. It uses the Versioning Benchmark (VB) [7].

Results: Table 5 shows the benchmark configuration and Table 6 shows the Virtuoso 8.0 results for the versioning task.

Task 4 - Browsing: The task on faceted browsing checks existing solutions for their capabilities of enabling faceted browsing through large-scale RDF datasets, that is, it analyses their efficiency in navigating through large datasets,

[7] `https://github.com/hobbit-project/DataStorageBenchmark/blob/master/system/virtuoso.ini.template`

Table 3: DSB Configuration.

Parameter	Value
Scale factor	1
Number of Operations	15000
Enable Sequential Tasks	true
Seed	100

Table 4: DSB KPIs for Virtuoso 8.0.

KPI	Value
Average Query Execution Time	22.2736
Loading Time (in ms)	372332
Query Failures	0
Throughput (queries/s)	40.0752

Table 5: VB Configuration.

Parameter	Value
Generated Data Form	IC
Initial Version Size	50000
Number of Versions	5
Version Deletion Ratio (%)	3
Version Insertion Ratio (%)	5

Table 6: VB KPIs for Virtuoso 8.0.

KPI	Value
Applied changes speed (changes/s)	9819.67
Initial Ingestion speed (triples/s)	12583.44
Queries Failed	2
Throughput (queries/s)	2.7946

where the navigation is driven by intelligent iterative restrictions. The goal of the task is to measure the performance relative to dataset characteristics, such as overall size and graph characteristics [8].

Results: Unlike the previous tasks where we executed our experiments in the training phase on the HOBBIT platform, the new version of this benchmark has not been ported to it yet. Therefore, we executed the Versioning task on a local Virtuoso instance, using the training data and queries made available by MOCHA 2018. The Virtuoso 8.0 query execution times for the benchmark queries are presented in the Table 7 and Table 8.

Table 7: Query Execution Times (in ms) for Scenario 1.

Q_Id	Time	Q_Id	Time	Q_Id	Time
1	119	7	40	13	16
2	27	8	94	14	26
3	21	9	25	15	16
4	11	10	81	16	26
5	95	11	24		
6	40	12	14		

Table 8: Query Execution Times (in ms) for Scenario 2.

Q_Id	Time	Q_Id	Time	Q_Id	Time
1	12	7	13	13	13
2	8	8	7	14	10
3	7	9	13	15	9
4	8	10	8	16	9
5	8	11	13	17	9
6	13	12	11		

4 Conclusion and Future Work

This paper is to be considered as a part of the registration process of MOCHA 2018, a challenge included in the Challenges Track of ESWC 2018. We express interest to participate in the following tasks: (i) RDF data ingestion, (ii) data storage, (iii) versioning and (iv) faceted browsing. A short overview of the Virtuoso Universal Server has been presented. The evaluation part of the paper contains the measurements from the training phase of all the tasks of MOCHA 2018, performed on the HOBBIT platform. The results represent an excellent guideline as to where the Virtuoso optimizer should be improved.

As future work, a further Virtuoso evaluation has been planned, using other dataset sizes and especially larger datasets, stressing its scalability. We can already foresee improvements of the query optimizer, driven by the current evaluation. A comparison of our performance with other systems registered for the challenge will be based on these four tasks, but with different benchmark parameters specified by the challenge organizers. We expect more demanding parameters, which will provide a fair comparison in the official results after the challenge.

Acknowledgments. This work has been supported by the H2020 project HOBBIT (GA no. 688227).

References

1. Orri Erling. Virtuoso, a Hybrid RDBMS/Graph Column Store. *IEEE Data Eng. Bull.*, 35(1):3–8, 2012.
2. Orri Erling and Ivan Mikhailov. RDF Support in the Virtuoso DBMS. In *Networked Knowledge-Networked Media*, pages 7–24. Springer, 2009.
3. Orri Erling and Ivan Mikhailov. Virtuoso: RDF support in a native RDBMS. In *Semantic Web Information Management*, pages 501–519. Springer, 2010.
4. Kleanthi Georgala. Data Extraction Benchmark for Sensor Data, 2017. `https://project-hobbit.eu/wp-content/uploads/2017/06/D3.1.1_First_Version_of_the_Data_Extraction_Benchmark_for_Sensor_Data.pdf`.
5. Kleanthi Georgala, Mirko Spasić, Milos Jovanovik, Henning Petzka, Michael Röder, and Axel-Cyrille Ngonga Ngomo. *MOCHA2017: The Mighty Storage Challenge at ESWC 2017*, pages 3–15. Springer International Publishing, Cham, 2017.
6. Milos Jovanovik and Mirko Spasić. First Version of the Data Storage Benchmark, 2017. `https://project-hobbit.eu/wp-content/uploads/2017/06/D5.1.1_First_version_of_the_Data_Storage_Benchmark.pdf`.
7. Vassilis Papakonstantinou, Irini Fundulaki, Giannis Roussakis, Giorgos Flouris, and Kostas Stefanidis. First Version of the Versioning Benchmark, 2017. `https://project-hobbit.eu/wp-content/uploads/2017/06/D5.2.1_First_Version_Versioning_Benchmark.pdf`.
8. Henning Petzka. First Version of the Faceted Browsing Benchmark, 2017. `https://project-hobbit.eu/wp-content/uploads/2017/06/D6.2.1_First_Version_FacetedBrowsing.pdf`.
9. Mirko Spasić and Milos Jovanovik. *MOCHA 2017 as a Challenge for Virtuoso*, pages 21–32. Springer International Publishing, Cham, 2017.