Big Linked Data ETL Benchmark on Cloud Commodity Hardware

Dieter De Witte, Laurens De Vocht, Ruben Verborgh, Kenny Knecht, Filip Pattyn, Hans Constandt, Erik Mannens, and Rik Van de Walle

In: Proceedings of the International Workshop on Semantic Big Data, 12:1–12:6, 2016.

http://doi.acm.org/10.1145/2928294.2928304

Big Linked Data ETL Benchmark on Cloud Commodity Hardware

Dieter De Witte[1] Laurens De Vocht[1] Ruben Verborgh[1] Kenny Knecht[2]
Filip Pattyn[2] Hans Constandt[2] Erik Mannens[1] Rik Van de Walle[1]

[1]iMinds - Data Science Lab, Ghent University
{firstname.lastname}@ugent.be

[2]Ontoforce
{firstname}@ontoforce.com

ABSTRACT

Linked Data storage solutions often optimize for low latency querying and quick responsiveness. Meanwhile, in the back-end, offline ETL processes take care of integrating and preparing the data. In this paper we explain a workflow and the results of a benchmark that examines which Linked Data storage solution and setup should be chosen for different dataset sizes to optimize the cost-effectiveness of the entire ETL process. The benchmark executes diversified stress tests on the storage solutions. The results include an in-depth analysis of four mature Linked Data solutions with commercial support and full SPARQL 1.1 compliance. Whereas traditional benchmarks studies generally deploy the triple stores on premises using high-end hardware, this benchmark uses publicly available cloud machine images for reproducibility and runs on commodity hardware. All stores are tested using their default configuration. In this setting Virtuoso shows the best performance in general. The other tree stores show competitive results and have disjunct areas of excellence. Finally, it is shown that each store's performance heavily depends on the structural properties of the queries, giving an indication of where vendors can focus their optimization efforts.

CCS Concepts

•**Information systems** → *Database query processing; Database performance evaluation; Resource Description Framework (RDF);*

Keywords

Big Data; Linked Data; Benchmark; SPARQL; Cloud Computing

1. INTRODUCTION

Linked Data has the potential to alleviate the burden of data integration by explicitly incorporating the semantics in the data using standardized ontologies. Integrating multiple datasets is especially beneficial for multidisciplinary research domains such as Life Sciences. Integration of multiple datasets though, introduces a set of Big Data related problems, requiring data management solutions offering scalable performance.

Scaling up has been the preferred method in the past: adding more RAM memory and CPUs in a single high-end machine, and indeed it has been shown that a single high-memory machine provides a very attractive platform to perform graph analytics [10]. Scaling up is further explored in for example Blazegraph which can make use of GPU accelerations.[1].

The *scaling out* approach relies on low-end commodity hardware but uses many nodes in a distributed system. When looking at distributed systems, multiple approaches to query "Resource Description Framework" (RDF)[2] data sources co-exist.

- *Specialized scalable RDF stores, the focus of this paper;*
- Translating SPARQL and RDF to existing NoSQL stores [3];
- Translating SPARQL and RDF to existing Big Data approaches such as MapReduce [11], Impala [12], Apache Spark [4];
- Distributing the data in physically separated SPARQL endpoints over the Semantic Web, using federated querying techniques [14] to resolve complex questions.

Compression (in-memory) is an alternative for the distributed approaches. RDF datasets can be compressed using for example the "Header Dictionary Triples" HDT [6] format and queried using Apache Jena Fuseki[3]. HDT has demonstrated compression ratios up to 20, pre-empting the need for a distributed environment. HDT-MR [7] is a recent HDT extension which allows the compression of big datasets which overcomes the in-memory constraints of the original HDT implementation by relying on a MapReduce pipeline. HDT is also the preferred choice in a new Semantic Web querying approach, called Linked Data Fragments [15] which offers linked data interfaces over the web excelling in availability.

This research was conducted in collaboration with Ontoforce[4], a company specializing in semantic search solutions for Life Sciences data. They require RDF query solutions which are mature, have reliable support backing their product, are rich in features and embrace the full SPARQL 1.1 standard. The strength of their product lies in the ability to find (unexpected) links between different datasets. In order for link discovery to be combined with a responsive product much of the heavy lifting should be offloaded into a preparatory ETL process. The latter explains the need for a benchmark which focuses on ETL as opposed to query responsiveness and latency in most other benchmark studies.

SBD'16, July 01 2016, San Francisco, CA, USA
© 2016 ACM. ISBN 978-1-4503-4299-5/16/07...$15.00
DOI: *http://dx.doi.org/10.1145/2928294.2928304*

[1] *https://www.blazegraph.com/product/gpu-accelerated/*

[2] RDF is the generally accepted method for conceptual description or modeling of information in which data is represented as triples.

[3] *http://jena.apache.org/documentation/serving_data/index.html*

[4] *http://www.ontoforce.com*

1.1 Problem Definition

Since the focus of this paper is on Extract Transform Load (ETL) pipelines, *execution cost* and *total runtime* are the two most important success measures as opposed to low latency and responsiveness which are less crucial in this context. Robustness under long periods of heavy stress loads, the absence of fails and timeouts are essential for ETL. The influence of the performance of an individual query on the total ETL workload is often negligible. However, an RDF solution's performance on certain query *types* might have a large impact on the total cost.

In this paper we address the following research questions:

- What is the most cost-effective storage solution to support Linked Data applications that need to be able to deal with heavy ETL query workloads? Which performance trade-offs do storage solutions offer in terms of vertical and horizontal scalability?
- What is the best configuration of a distributed solution to deal with different dataset sizes?
- What is the impact of different query types (templates)? Is there a difference in performance between the stores based on the structural properties of the queries?

We do not take into account implicitly derived facts or *reasoning* on the datasets. We turned off the reasoning feature in all stores.

1.2 Related Work

Artificial benchmarks are often used to assess the performance of triple stores. Some examples are the Lehigh University Benchmark [8] (LUBM), the Berlin SPARQL Benchmark [2] (BSBM), DBPBM[9], and the SP^{2}Bench [13].

Recently, an in-depth study of the university of Waterloo has shown that all existing artificial benchmark suites are lacking in diversity, both in the diversity of the structural properties of the queries they offer and in the data-driven properties [1]. The latter are related to the connectedness of the data and therefore have a high influence on the selectivity of query fragments and the performance of the query planner. As a result they offer a benchmark suite which generates more diverse queries and more diverse benchmark datasets: The Waterloo SPARQL Diversity Test Suite (Watdiv).

Apart from a lack of diversity another criticism is that artificial benchmarks do not generalize well to real-world datasets [5]. BioBenchmark Toyama 2012 [16] is the most recent large scale RDF benchmark on real Life Sciences data. They evaluated the runtime of 5-10 queries on 5 real datasets varying in size from 10 million to 8 billion triples (DDBJ) on high-end single node triple store installations.

Mauroux [3] evaluated the performance of NoSQL stores to handle RDF data workloads and uses the notion of *cost* to evaluate different data management solutions, a metric also used in this study.

1.3 Our Contributions

The main contribution of this paper is the re-usability, straightforward interpretation of the test results and very detailed insights it offers without reinventing well-proven SPARQL benchmarks. The results presented in this paper can be easily applied to other storage solutions.

Rather than focusing on high-end infrastructure or case-specific benchmarks, this benchmark supports enterprise scenarios where the data endpoint is not used for interactive querying but used as a system in an ETL pipeline used to interlink RDF data of multiple (open) data sources.

Section 2 explains the benchmark and how it can be reused: data and queries can be replaced to suit test case needs and we explain the choice of RDF stores to include in our benchmark. The main results of the benchmark are summarized in terms of execution time, and cost ($) in section 3.1 and an in-depth analysis of the results focusing on scalability and the influence of different query types is presented in section 3.2 and 3.3.

2. METHODOLOGY

The methodology for executing the benchmark consists of three major steps: (i) generating the RDF data and the ETL queries; (ii) set-up storage solutions; (iii) load data into the store and run the ETL queries while performing time measurements.

2.1 Data and Query Generation

WatDiv provides stress testing tools to address the observation that existing SPARQL benchmarks are not suitable for testing systems for diverse queries and varied workloads [1]. In this ETL benchmark we decided to choose Watdiv because it is a generic benchmark which is not application specific. It covers a broad spectrum in terms of result cardinality and triple-pattern selectivity. This is ensured through the way it generates data and queries. Furthermore, others can repeat the benchmark, even with different dataset set sizes or different numbers of queries available [5].

For this study three datasets have been generated via the Watdiv Suite of resp. 10, 100 and 1000 million RDF triples. For each dataset a set of 2000 queries is generated in a randomized fashion. The queries are generated from 20 query templates which can be classified in 4 more general template types, this will be discussed in section 3.3.

2.2 Selection of RDF Storage Solutions

The selection of triple stores to benchmark is guided by the requirement that the RDF store should be capable of serving in a production environment with Life Sciences RDF data. The initial selection was made by choosing stores with a) a high adoption/popularity as defined by DB-Engines ranking for RDF stores[6], b) enterprise support, c) support for distributed deployment and d) full SPARQL 1.1 compliance.

The four stores we selected all comply with the 4 constraints put forward in the previous section. Two stores opted for under nondisclosure and will be labeled as Entreprise Store I and II (ESI and ESII) All stores support running in a high availability mode, (automated replication), and all but ESI allow for automated sharding. The latter implies that the dataset must fit into a single machine for ESI.

2.3 Infrastructure Setup

We choose to execute all benchmarks on the Amazon Webservices Elastic Compute Cloud (EC2) and Simple Storage Solutions (S3). We use the default commercial deployments of the storage solutions under test because we want the results to be completely reproducible: both the hardware and the machine images can be easily acquired. More generally cloud deployments offer the advantage of not requiring dedicated on-premises hardware.

A PAGO license was used for Virtuoso[7], Blazegraph[8] is an open-source product which was installed manually with the default con-

[5] *http://dsg.uwaterloo.ca/watdiv/*

[6] *http://db-engines.com/en/ranking/rdf+store*

[7] *https://aws.amazon.com/marketplace/pp/B011VMCZ8K/ref=srh_res_product_title?ie=UTF8&sr=0-5&qid=1455494788712*

[8] *https://www.blazegraph.com/product/*

Table 1: Runtime and cost for an ETL run of 2000 unique queries evaluated on 4 RDF solutions and 2 distributed setups.

Store	Dataset size (million triples)	Load Time (s)	Median Runtime (s)	Median Respon-setime (s)	Instance cost ($/Hr)	Load cost ($)	Run cost ($)	Total cost ($)
Blazegraph 2.0.0	10	246	1,578	142	0.33	0.02	0.15	0.17
	100	5,784	13,343	754	0.33	0.54	1.23	1.77
	1000	**181,362**	141,897	83,442	0.33	**16.78**	13.13	29.90
Enterprise Store I	10	**641**	1,069	721	0.33	**0.06**	0.10	0.51
	100	**10,457**	29,776	22,832	0.33	**0.97**	2.75	12.10
	1000	168,780	285,350	262,672	0.33	15.61	26.39	136.62
Enterprise Store II	10	601	**3,242**	**3,047**	0.33	**0.06**	**0.30**	**1.41**
	100	5,498	**33,524**	**33,522**	0.33	0.51	**3.10**	**14.34**
	1000	58,912	**357,478**	**357,460**	0.33	5.45	**33.07**	153.02
Virtuoso 7.2	10	68	152	138	0.33	0.01	0.01	0.06
	100	1,290	797	778	0.33	0.12	0.07	0.57
	1000	13,940	9,802	9,629	0.33	1.29	0.91	6.48
Enterprise Store II (3 nodes)	1000	56,915	86,757	86,754	1.00	15.79	24.08	**158.40**
Virtuoso 7.2 (3 nodes)	1000	54,850	20,570	20,392	1.00	15.22	5.71	61.78

figuration. SPARQL Query Benchmarker [9] is a general purpose API that was originally designed primarily for testing remote SPARQL servers but can be extended to test much more than that. One of the main advantages is that the actual benchmark consists of some number of runs of the operation mix. By default operations are run in a random order for each run to try and avoid the system under test (SUT) learning the pattern of operations and aggressively caching and thus gaming the benchmark.

A single benchmark consists of N (we consider $N = 1$ and $N = 3$) triple store nodes of the type *r3.xlarge* (4 vCPU, 30.5 GB RAM) and a *c3.2xlarge* (8 vCPU, 15 GB RAM) node to run the SPARQL benchmarker software, more instance details can be found on the AWS site[10].

2.4 Benchmark process

The benchmark process consists of a data loading phase, followed by running the SPARQL benchmarker. The data is loaded in compressed format (gzip). The benchmarker runs in multi-threaded mode (8 threads), runs a set of 2000 queries and does so multiple times. These runs consists of at least one warm-up run which is not used in the results and multiple regular runs. In order to obtain robust results the tail results (most extreme) are discarded before calculating average query runtimes. The benchmarker generates a CSV file containing the run times and response times etc. of all queries. We made all detailed results of the benchmark process online[11].

3. RESULTS

The results after executing the benchmark allow analysis of the tested storage solutions and focus on three main aspects: (i) cost-effectiveness; (ii) scalability; and (iii) influence of different query types (templates). This section goes over each of these aspects in detail applied to the four RDF storage solutions we tested: Blazegraph, Virtuoso, ESI and ESII.

3.1 ETL Runtimes and Cost

Apart from individual query runtimes the SPARQL Benchmarker also records the overall ETL process runtimes and response times. The response time is considered to be the time from when the query is fired by the client to when the store starts responding with its first results. The load times are reported by the RDF databases or extracted from their logs.

Table 1 shows the runtimes and associated costs for full ETL runs on 3 different dataset sizes of respectively 10, 100 and 1000 million (M) triples. For every dataset size the best value is highlighted, the worst is bold-faced.

Each of the data stores has a bulk loading functionality to insert RDF triples without transactional overhead. Virtuoso outperforms other solutions by an order of magnitude in terms of loading time. ESI spends the most time in the loading phase except for the 1000M dataset where suddenly the loading time of Blazegraph takes over. ESII, although on average 5 times slower than Virtuoso holds the second place for all datasets.

A longer load time might prove beneficial in the subsequent query runtimes. This is immediately obvious for the case of ESII which is worse for all dataset sizes. For small dataset sizes ESI ranks second, while for the other datasets Blazegraph is at least twice faster than the other two. Virtuoso, also for the ETL runtimes proves superior to the other 3 stores with runtimes over 10 times lower than the second in rank. In order to compare the performance on load and a single run combined, the column showing the total cost (without license) can be used. In general Blazegraph holds the second rank while ESI performs best for small datasets and ESII the other way around.

Response times are less important in an ETL context, but it is worth mentioning that Blazegraph is the only store who has a large difference in response time compared to the runtime. For small datasets the stores starts responding in 5 to 10% of the total runtime, while for the largest 1000 M dataset response time is close to 60% of the total runtime.

The right hand side of **Table 1** focuses on the cost. This can lead to a different ranking due to the difference in license costs of the PAGO instances. Blazegraph can be used without any license fees widening the cost gap with ESII and ESI, to a factor of 4 for the 1000M dataset. For the single instance solutions ESII is the most expense on all three datasets.

The speed-up for ESII in distributed mode in terms of runtime is remarkable, almost 5 times as fast as ESII on a single instance, even though the number of nodes increased from 1 to 3. This speedup factor can at least partially be attributed to query timeouts,

[9] *http://sourceforge.net/p/sparql-query-bm/wiki/Introduction/*

[10] *https://aws.amazon.com/ec2/instance-types/*

[11] *http://lddemo.datasciencelab.be/sequel*

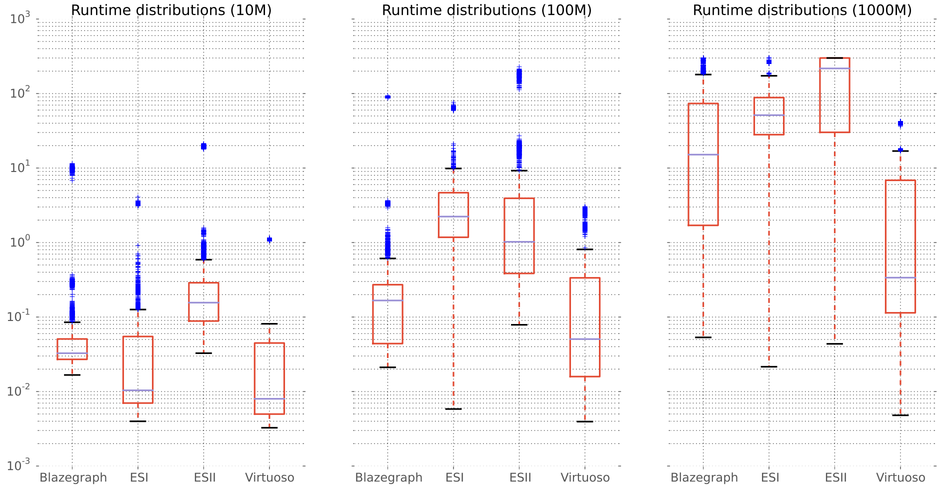

Figure 1: Boxplot (log scale) of the runtime distribution of a query mix per storage solution for three dataset sizes. The query mix consists of 2000 queries. Note that data points on the 300 seconds line correspond to timeouts.

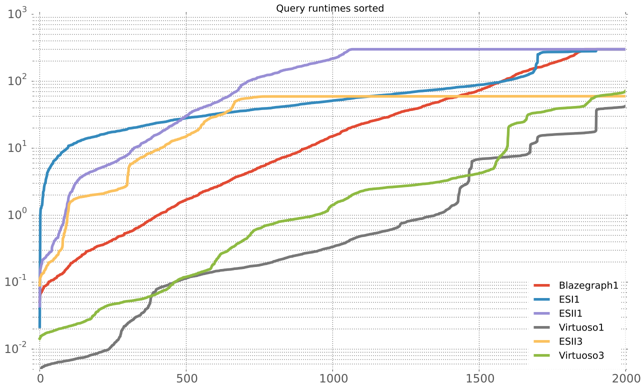

Figure 2: For every store the query runtimes are sorted in increasing size for a dataset of 1 billion triples. Note that ESII on three instances has a standard timeout of 60 seconds, explaining the early asymptotic behavior.

Figure 2 shows that the speedup for the other queries is significant. Nonetheless is this option still the most expensive one because of the higher infrastructure costs. If we focus on run cost alone the cost is actually over 25% less for the distributed mode. The load time for both modi is approximately equal. This can be attributed to the recommended configuration on AWS using only a single datanode. It can be assumed that in a different configuration scaling out will be beneficial for ESII.

3.2 Scaling from 10 million to 1 billion triples

Figure 1 shows the distribution of all average query runtimes of the datastores on the three different datasets. The warm-up and outlier values are excluded from the calculation of the average. Points of interests for the ETL case are the median runtime, the presence of a long tail and the 75% quartile (i.e. the upper side of the box).

Along all summary statistics Virtuoso is showing the best results. For the 10M dataset all stores perform within a similar range, except for ESII which is on average slower by at least an order of magnitude. For 10M the main differences for the ETL lies in the position of the datapoints in the outliers. Blazegraph is not capable to generate very low latency responses to the ETL queries, but manages to have the same runtime cap for 75% of the queries as Virtuoso for 10M and 100M dataset sizes, for 1000M the superiority of Virtuoso becomes more outspoken with a difference of an order of magnitude. While ESI's performance is close to that of Virtuoso on 10M, this observation cannot be made for the larger datasets where ESI in general ranks third.

It might seem counter-intuitive at first, but the tails of the distributions contribute the most to the ETL cost in **Table 1**. This can be seen in the 100M panel where ESII has a superior median and 75% runtime value, the heavy tail though is responsible for making ESII 50% slower as compared to ESI. In section 3.3 we further examine if and how this can be attributed to certain queries types.

The SPARQL benchmarker uses a timeout parameter to cap the maximum runtime of a query, this parameter is set to 300 seconds. For the 10M and 100M datasets timeouts do not come into play, except for ESII where in 3.85% of the queries a timeout occurs at least once, but there are no queries which always timeout. With the 1000M dataset only Virtuoso is completely free from timeouts. The three other stores, Blazegraph, ESI and ESII, have a number of queries which timeout with every time: this is the case in respecitvely 7%, 5% and 47% of the cases. The timeouts are an important means to control the ETL runtime since they limit the effect of the extremely long running queries (even if it is only a single one).

In **Figure 2** the individual runtimes are sorted for each store, which implies that the X-axis corresponds to the rank of an individual query per store. The chart shows clearly the amount of timeouts in the 1000M dataset at the location where the asymptotic behaviour starts separates the successful from the unsuccessful queries which timed out. The results for ESII in distributed mode with three instances in the same chart expose that: (i) an internal timeout parameter is set, explaining why no queries can last longer than 60 seconds. 61% of the queries therefore are showing timeouts in every turn, for five minute timeouts this percentage is expected to be well below the one instance percentage; and (ii) the difference between single and three instance mode for ESII corresponds to a speedup of approximately 3.3. In Virtuoso the speed-up is less than 1. This is because the 1000M dataset's size does not exceed the capabilities of a single-server instance set-up of Virtuoso. As soon

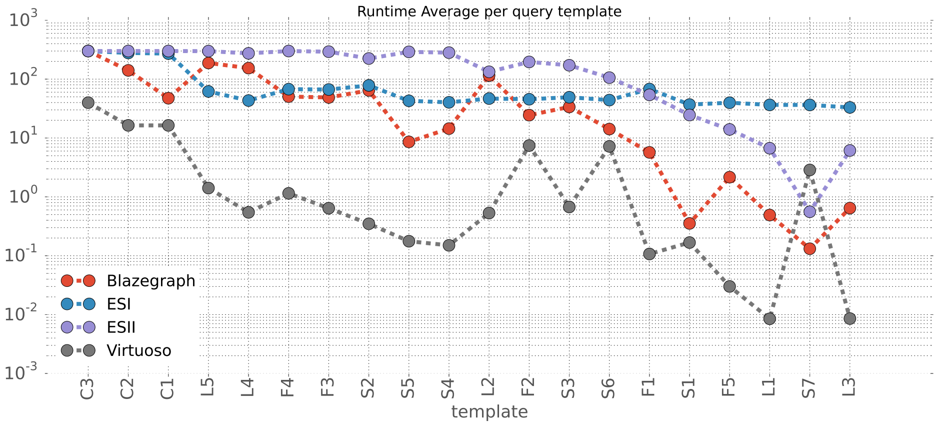

Figure 3: Behavior of storage solutions given 20 different query templates. Results are averaged of 100 queries per template. The query templates on the X axis are sorted by descreasing average runtime.

as a dataset size reaches a level that surpasses the memory available to a single-server instance, the scalability benefits of a multi-node configuration become clear. This occurs as a result of creating a greater memory pool, in that case, across multiple instances.

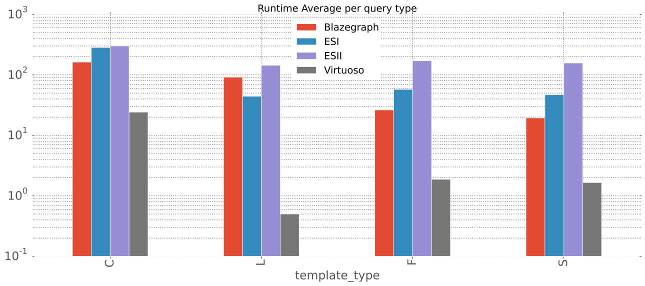

Figure 4: Behavior of storage solutions given different query template types: C (=complex), L (=linear), F (=snowflake), S (=star). Virtuoso is more than an order of magnitude faster than the other stores except of the query template type C.

3.3 Query template type performance

Each of the query types (templates) that Watdiv used to generate a query consists of basic graph patterns (BGPs). Every BGP combines triple patterns into query structures. This query structure is either a linear chain (**L**); a star (**S**), with a single node in the middle and one or more neighboring nodes; a snowflake (**F**), a combination of stars, and a combination (**C**) of all three. Given a set of query templates, the query generator instantiates these templates with actual RDF terms from the dataset. We instantiated 20 of these templates each with 100 queries, so in total we got the 2000 unique queries, more details of the templates can be found on the Watdiv website[12].

Figure 4 shows the overall performance per query template type, while **Figure 3** goes into more detail by showing the performance on the 20 query templates. In **Figure 4**, the C-type is a summary result, containing structural properties of both L, F and S-types. An interesting observation is that the overall ranking of the RDF solution per template is in general consistent except for the linear queries. This is a clear indication that Blazegraph should direct their optimization efforts towards providing better performance on linear chain queries. Another curious result is that for the C-type queries the results of ESI and ESII are almost the same, while they differ a lot on the other query template types, clearly ESI has been optimized for low complexity queries. The results of Virtuoso are in general better by an order of magnitude but interestingly for the linear chain queries they are better by even two orders of magnitude. The underlying data layout of Virtuoso 7.2 is a columnar while ESI and Blazegraph store the data as graphs. Therefore one would not expect the difference to benefit Virtuoso for this type of queries.

In **Figure 3** we get a more detailed overview of the performance on individual query templates. The templates are sorted on the runtimes averaged over all stores in ascending order. ESI and ESII are the only two stores which have maximum query runtimes for a fraction of the queries. For ESI the worst runtime behavior is seen for 6 templates, for ESII this for 14 templates. This difference is also dependent on the average template runtime: for the longer running queries ESII has the longest runtime while for the faster queries ESI has the worst runtime.

Virtuoso has the best performance for all but 2 templates: C1 and S7. Together with F2, S1 and S6 it might be worthwile to perform a more thorough analysis to understand the cause of this behaviour. For Blazegraph a number of templates have worse runtimes than ESI: L5, L4 and L2. This is consistent with the conclusion from the previous paragraph that linear chain queries prove to be somewhat problematic for Blazegraph.

4. CONCLUSIONS AND FUTURE WORK

The results of different benchmark studies might depend on many (hidden) factors leading to different or even contradicting results. The goals of this work is to identify the most suitable RDF database for ETL workloads in terms of total runtime and execution cost. An additional goal is to make the results as neutral and reproducible as possible. We tested four interesting solutions in this work and found that Virtuoso is still 2x - 20x cheaper and up to an order of magnitude faster than the other three stores. In general Blazegraph ranks second in the majority of the results which is a rewarding result for the open source community backing its development. ESI is most suitable for small datasets and ESII is interesting for its ability to scale out, the performance speed-up is a factor 3. It seems that Virtuoso lacks this ability to scale out: the cost of Virtuoso with 3 nodes is much more expensive while the cost for ESII remains the same. The difference in performance between the stores might be attributed to the use of commodity hardware. The choice for this

[12] *http://dsg.uwaterloo.ca/watdiv/basic-testing.shtml*

hardware was made with the affordability of scaling out to even larger datasets in mind. Very important to take into consideration is that the benchmark was run with the recommended configuration parameters offered by the PAGO machine images. The difference between the results might at least be partially attributed to the quality of the recommended configuration parameters for this benchmark. Note that one should be careful with generalizing the results to other use cases than the ETL case considered here. One minor result backing this warning is the fact that the response times for queries on Blazegraph are much lower than their full runtimes. The results of the queries may also be influenced in a nonlinear manner in workloads with less simultaneous query threads and different query types. One of the limitations of the Watdiv benchmark is that only BGPs and Filters occur in the queries. There are many SPARQL 1.1 features which are not covered by these two types, and although most queries can be rewritten in terms of BGPs and Filters, that does not imply that every storage solution will create an equally efficient query plan for these.

In terms of ETL runtime and cost Virtuoso dominates but there is no clear second place. Whereas ESI performs well on small datasets, Blazegraph shows better results for larger datasets. ESII's performance on a single instance benchmark is worse than the others, but its claim of being highly scalable is confirmed in a configuration with three instances where it performs significantly better. All data stores have acceptible results for 10 and 100 million triples and the choice to go by one or the other could depend on additional features each of the stores has to offer such as support for full-text indexing, support for linked data fragment interfaces or superior automatic inferencing. For the larger datasets Virtuoso should be the first choice as a single instance solution. The initial results in a distributed setup with ESII shows promising results in terms of scaling out, proving that this store's power might only be revealed in large multi-instance benchmarks. Also Blazegraph should definitely be analyzed in a distributed setting. As a final conclusion it is shown that the structural properties of the queries can play a major role in the ETL performance, with the most concrete example being the lesser performance of Blazegraph on linear chain queries, which can be a worthwile pointer for the team backing its development.

The benchmark results in this paper compare RDF stores with default configuration and without the intervention of Enterprise support. We plan to publish results in the future with the stores running in their optimal configuration. Multi-instance benchmarks will be the subject of follow-up research with even larger datasets. Future work should result in a more complete overview of the most suitable RDF solution for different use cases, different configuration possibilities and for real-world datasets. Part of the future work will focus on investigating whether the Watdiv results generalize to other query types and real life sciences datasets. Eventually we plan to release a set of publicly available tools for easily repeating the benchmark with new datastores of interest to the end-user.

5. REFERENCES

[1] G. Aluç, O. Hartig, M. T. Özsu, and K. Daudjee. Diversified stress testing of RDF data management systems. In *The Semantic Web–ISWC 2014*, pages 197–212. Springer, 2014.

[2] C. Bizer and A. Schultz. The Berlin SPARQL Benchmark. *International Journal on Semantic Web and Information Systems (IJSWIS)*, 5(2):1–24, 2009.

[3] P. Cudré-Mauroux, I. Enchev, S. Fundatureanu, P. Groth, A. Haque, A. Harth, F. L. Keppmann, D. Miranker, J. F. Sequeda, and M. Wylot. NoSQL databases for RDF: an empirical evaluation. In *The Semantic Web–ISWC 2013*, pages 310–325. Springer, 2013.

[4] O. Curé, H. Naacke, M.-A. Baazizi, and B. Amann. On the Evaluation of RDF Distribution Algorithms Implemented over Apache Spark. *arXiv preprint arXiv:1507.02321*, 2015.

[5] S. Duan, A. Kementsietsidis, K. Srinivas, and O. Udrea. Apples and oranges: a comparison of RDF benchmarks and real RDF datasets. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 145–156, 2011.

[6] J. D. Fernández, M. A. Martínez-Prieto, C. Gutiérrez, A. Polleres, and M. Arias. Binary RDF representation for publication and exchange (HDT). *Journal of Web Semantics*, 19:22–41, 2013.

[7] J. M. Giménez-García, J. D. Fernández, and M. A. Martínez-Prieto. HDT-MR: A Scalable Solution for RDF Compression with HDT and MapReduce. In *The Semantic Web. Latest Advances and New Domains - 12th European Semantic Web Conference ESWC*, pages 253–268, 2015.

[8] 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.

[9] M. Morsey, J. Lehmann, S. Auer, and A.-C. Ngonga Ngomo. DBpedia SPARQL benchmark–performance assessment with real queries on real data. *The Semantic Web–ISWC 2011*, pages 454–469, 2011.

[10] Y. Perez, R. Sosič, A. Banerjee, R. Puttagunta, M. Raison, P. Shah, and J. Leskovec. Ringo: Interactive graph analytics on big-memory machines. In *Proceedings of the 2015 ACM SIGMOD International Conference on Management of Data*, pages 1105–1110. ACM, 2015.

[11] A. Schätzle, M. Przyjaciel-Zablocki, T. Hornung, and G. Lausen. PigSPARQL: A SPARQL Query Processing Baseline for Big Data. In *Proceedings of the ISWC 2013 Posters & Demonstrations Track, Sydney, Australia, October 23, 2013*, pages 241–244, 2013.

[12] A. Schätzle, M. Przyjaciel-Zablocki, A. Neu, and G. Lausen. Sempala: Interactive SPARQL Query Processing on Hadoop. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Proceedings, Part I*, pages 164–179, 2014.

[13] 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.

[14] A. Schwarte, P. Haase, K. Hose, R. Schenkel, and M. Schmidt. FedX: Optimization Techniques for Federated Query Processing on Linked Data. In *The Semantic Web - ISWC 2011 - 10th International Semantic Web Conference, Proceedings, Part I*, pages 601–616, 2011.

[15] R. Verborgh, O. Hartig, B. De Meester, G. Haesendonck, L. De Vocht, M. Vander Sande, R. Cyganiak, P. Colpaert, E. Mannens, and R. Van de Walle. Querying Datasets on the Web with High Availability. In *The Semantic Web - ISWC 2014 - 13th International Semantic Web Conference, Proceedings, Part I*, pages 180–196, 2014.

[16] H. Wu, T. Fujiwara, Y. Yamamoto, J. Bolleman, and A. Yamaguchi. BioBenchmark Toyama 2012: an evaluation of the performance of triple stores on biological data. *Journal of biomedical semantics*, 5(1):1, 2014.