Beazley: a New Storage Systems Evaluation

Mikalai Yatskevich[1], Ian Horrocks[1], and Graham Klyne[2]

[1] Oxford University Computing Laboratory,
Wolfson Building, Parks Road,
Oxford OX1 3QD, UK
mikalai.yatskevich@comlab.ox.ac.uk
ian.horrocks@comlab.ox.ac.uk
[2] Zoology Department, Oxford University,
South Parks Road,
Oxford OX1 3PS, UK
graham.klyne@zoo.ox.ac.uk

Abstract. Evaluation is a major issue in the development of systems, sometimes as important as the implementation of a system itself. In the Semantic Web area, and especially in the area of the storage systems that provide a persistence layer for ontologies and instance data, evaluation efforts have been intermittent and area specific. In this paper we propose a new dataset for storage systems evaluation called Beazley dataset. The complete dataset version includes more than 16 millions of triples and 35 queries. We evaluate dataset exploiting several storage models of the state of the art storage systems.

1 Introduction

Evaluation is a systematic assessment of system properties against a set of predefined criteria. Evaluation is a major issue in the development of systems, sometimes even as important as the implementation of a system itself. It has been shown in the past that performance evaluation can help implementers to better understand the sources of intractability and/or inefficiency in their systems, and to propose novel optimization techniques in an effort to make their systems more scalable in specific application scenarios.

Storage systems often called RDF stores provide a persistence layer for ontologies and instance data. They provide basic reasoning services such as computing transitive closure of the subsumption hierarchies. Storage systems differ from description logics (DL) reasoners that provide more complex reasoning services but do not provide storage facilities. The main inference services in the DL reasoners can be performed as conceptual satisfiability. For RDF stores, the main inference service is query answering.

In the Semantic Web area, and especially in the area of storage systems, evaluation efforts have been intermittent and area specific. There is no agreed standard or methodology for systems evaluation. In the evaluation of the DL systems artificially generated datasets served an important role [13] until a large

Proceedings of the International Workshop on Evaluation of Semantic Technologies (IWEST 2010). Shanghai, China. November 8, 2010.

number of the real-world ontologies has been developed. This real-world ontologies have been used in the large scale evaluation efforts [9]. The evaluation of the storage systems are focused on the artificially generated datasets [12, 17, 10]. Thus, the evaluation of the storage systems will benefit from the real-world datasets that will overcome the limitations of the state of the art generation methods. The most common instance generator and evaluation suite that is used by the Semantic Web community for storage systems evaluation is the Lehigh University Benchmark (LUBM) [12]. Although, LUBM is used mainly for testing instance retrieval and query answering algorithms, it also has many shortcomings. First of all the $\mathcal{ALEHI}_{\mathcal{R}+}$ DL used is significantly less expressive than the DL underpinning OWL. Moreover, the data that are created for each university are completely independent. Consequently, if one applies a clustering method during loading it is possible to apply query answering over each university independently.

In this paper we propose a new dataset for storage systems evaluation called Beazley dataset. The dataset comprises real world archeological data gathered in CLAROS initiative [15] and a set of queries used in CLAROS web cite application. The dataset presents the information about archeological artifacts. It instantiates CIDOC-CRM OWL DL ontology [8]. The complete dataset version includes more than 16 millions of triples and 35 queries. We evaluate the dataset exploiting both memory and disk-based storage models of the two state of the art storage systems.

The paper is structured as follows. Section 2 provides a brief introduction to the storage systems along with the datasets used for their evaluation. Section 3 provides a detailed description of the Beazley dataset. Section 4 provides a detailed description of the dataset evaluation set up. Section 5 describes the evaluation results. Section 6 concludes the paper.

2 Related Work

The majority of the evaluation efforts in the storage systems area were focused on artificially generated datasets. They provide a mechanism to cover a class of inputs in a scalable manner. The most prominent example is the Lehigh University Benchmark (LUBM) [12]. LUBM consist of a small ontology, with 43 classes, 25 roles, 85 TBox axioms and 8 RBox axioms, and several Java classes that can be used to create instance assertions (ABox) for this specific TBox and RBox. The ontology describes universities, i.e. courses, students, departments, publications as well as their interrelations. For example, a student is enrolled in some courses that is taught by some academic staff, while academic staffs are associated with publications, are affiliated with other universities, lead research teams or are heads of departments. The DL of LUBM is $\mathcal{ALEHI}_{\mathcal{R}+}$, nevertheless it does not make heavy use of the constructors since there is just one transitive role, 5 sub-role axioms and 2 inverse role axioms. The ABox is created following the method described in [4]. Finally, the benchmarking suite also comes with 14 queries that are proposed for testing a system against the generated ABoxes.

Although, LUBM is used mainly for testing instance retrieval and query answering algorithms, it also has many shortcomings. First of all the DL used is significantly less expressive than the DL underpinning OWL. Moreover, the data that are created for each university are completely independent. Consequently, if one applies a clustering method during loading it is possible to apply query answering over each university independently. An extension of LUBM to remedy these problems was the University Ontology Benchmark (UOBM) [17]. UOBM extends LUBM by adding more concepts and roles that are intended to connect individuals from different universities. Although UOBM is still not large when compared to ontologies such as NCI [11] or GALEN [21], it uses a relatively expressive ontology language $\mathcal{SHIN}(\mathbf{D})$. Finally, a set of test queries is also offered. Unfortunately, although UOBM does indeed make use of more complex constructors and is more structurally complex it has not been widely accepted by the Semantic Web community.

The Berlin SPARQL benchmark [5] focus on integration and visualization data from various data sources. It is build around scenario that does not require heavyweight reasoning. The class hierarchy is generated in random way. The query mix includes 25 queries that represent navigation pattern in e-commerce use case. SP2Bench [22] uses DBLP [16] bibliographic scenario. The ontology used have 9 classes and 77 properties. The query mix includes 11 queries utilizing various SPARQL language constructs. The Billion Triple Challenge [3] aims at the evaluation of the Semantic Web applications to process a large quantities of the RDF data that is represented by various schemata.

3 The Beazley dataset

The Beazley dataset [15] presents the information about archeological artifacts. The RDF data instantiates CIDOC-CRM ontology [8]. The complete dataset version includes more than 16 millions of triples. The frequency of the triples in the dataset depending on predicate values $f_p = freq(D, p)$ is depicted on Figure 1a. The frequency of the triples with a given subject value frequency $f_{s_n} = freq(D, s_n)$, depicted on Figure 1b, varies depending on a predicate value p. The $f_{s_n} = freq(D, s_n, is\_represented)$ and $f_{s_n} = freq(D, s_n, is\_reffered)$ are depicted on Figures 1c, 2a. This makes Beazley archive dataset different from RDF datasets produced using automatic generation procedures [12, 17]. In these works the uniform distributions and, hence, frequencies are assumed. The frequency of the triples with a given object value frequency $f_{o_n} = freq(D, o_n)$, depicted on Figure 2b, varies depending on a predicate value p and a subject value frequency s_n. The $f_{o_n} = freq(D, o_n, s_n, has\_time\_span)$, $f_{o_n} \in [1, 30]$ is depicted on Figure 2c. The $f_{o_n} = freq(D, o_n, took\_place)$, $f_{o_n} = freq(D, o_n, not\_after)$ are depicted on Figures 3a, 3b. The query set used in CLAROS web cite application [15] composed from 35 queries of various size and complexity. The different queries are executed different number of times during the web application life cycle. They could be classified into two large groups. The first query group QG1 comprises Q1-Q18 presented at Table 1. The queries from QG1 are executed at

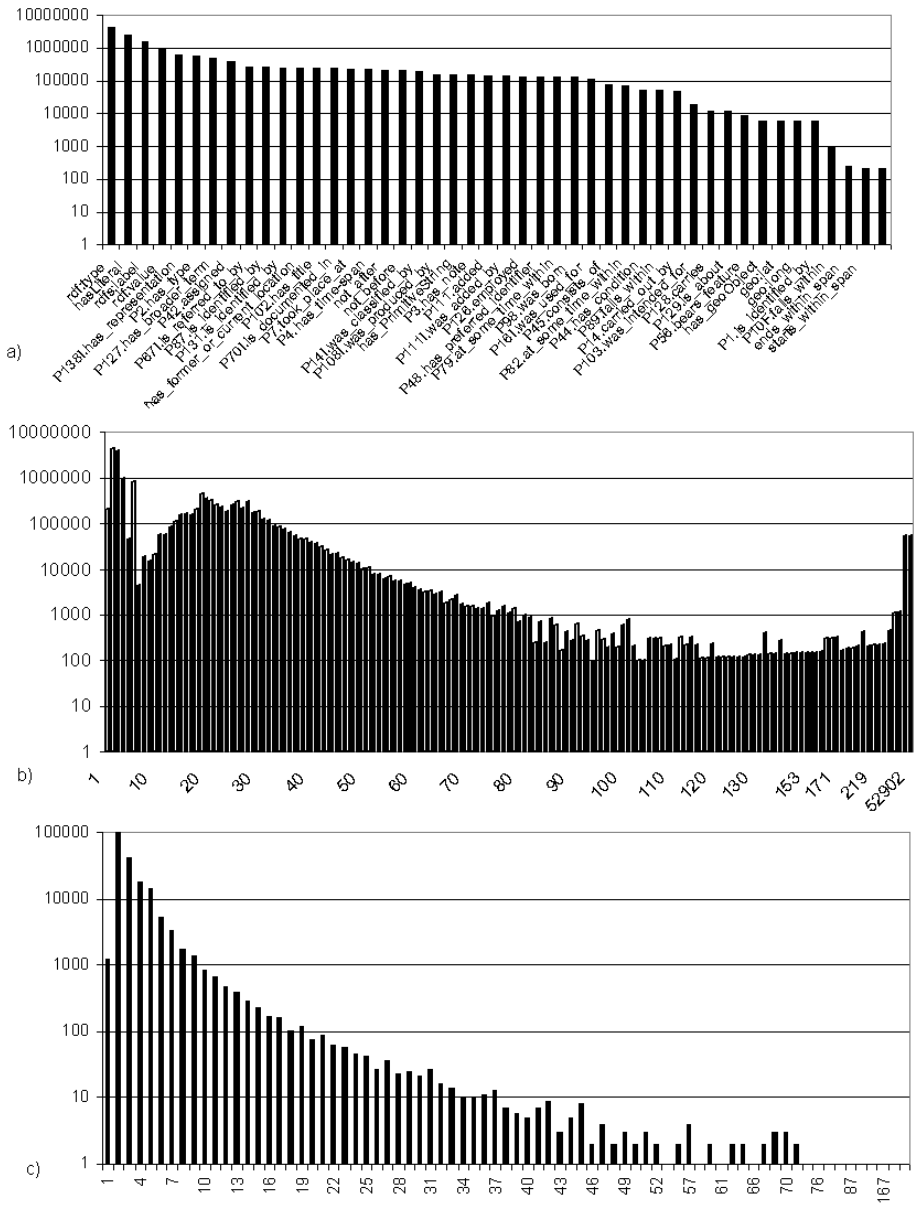

Fig. 1. a) The frequencies of predicates; b) The frequencies of the triples with a given subject value frequency; c) The frequencies of the triples with a given subject value frequency for $predicate = is\_represented$

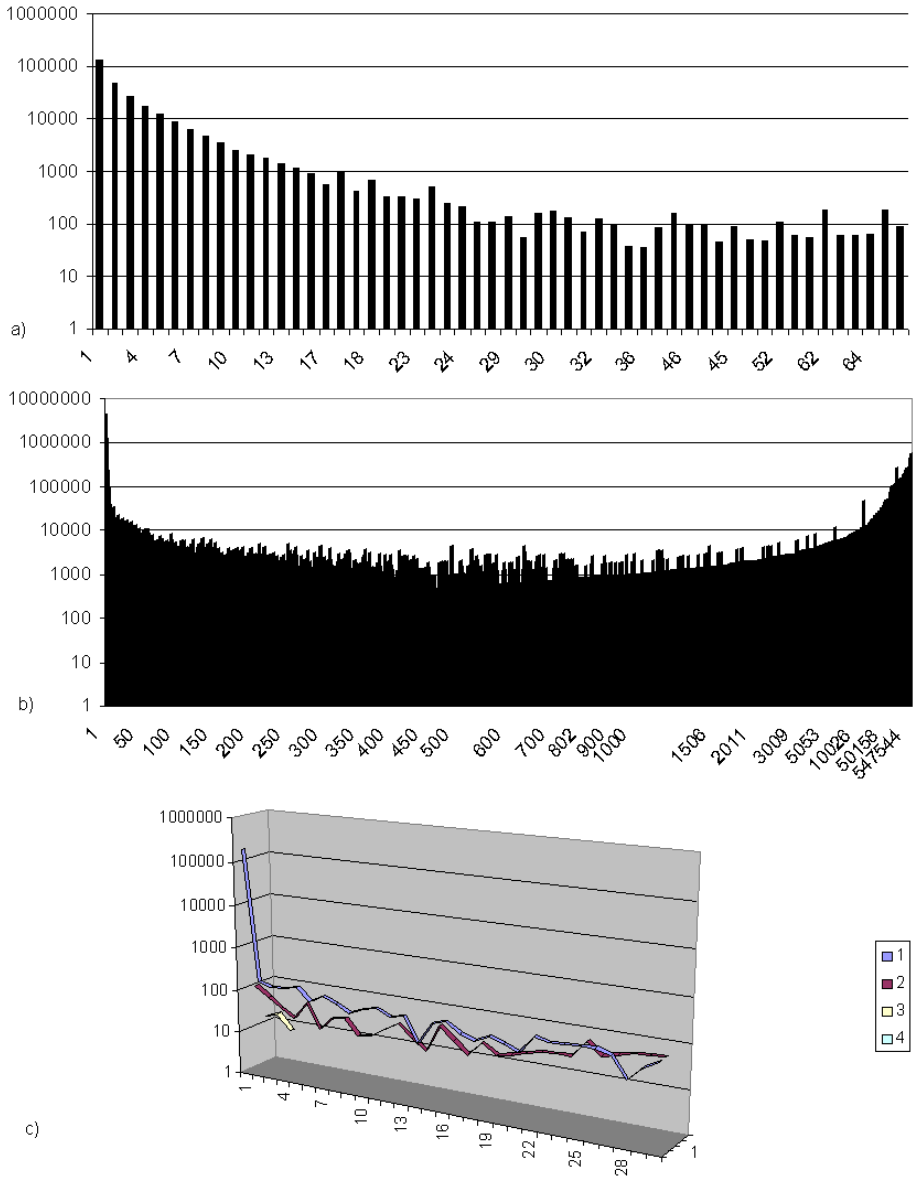

Fig. 2. a) The frequencies of the triples with a given subject value frequency for $predicate = is\_reffered$; b) The frequencies of the triples with a given object value frequency; c) The frequencies of the triples with a given object value frequencies in $[1, 30]$ for $predicate = has\_time\_span$

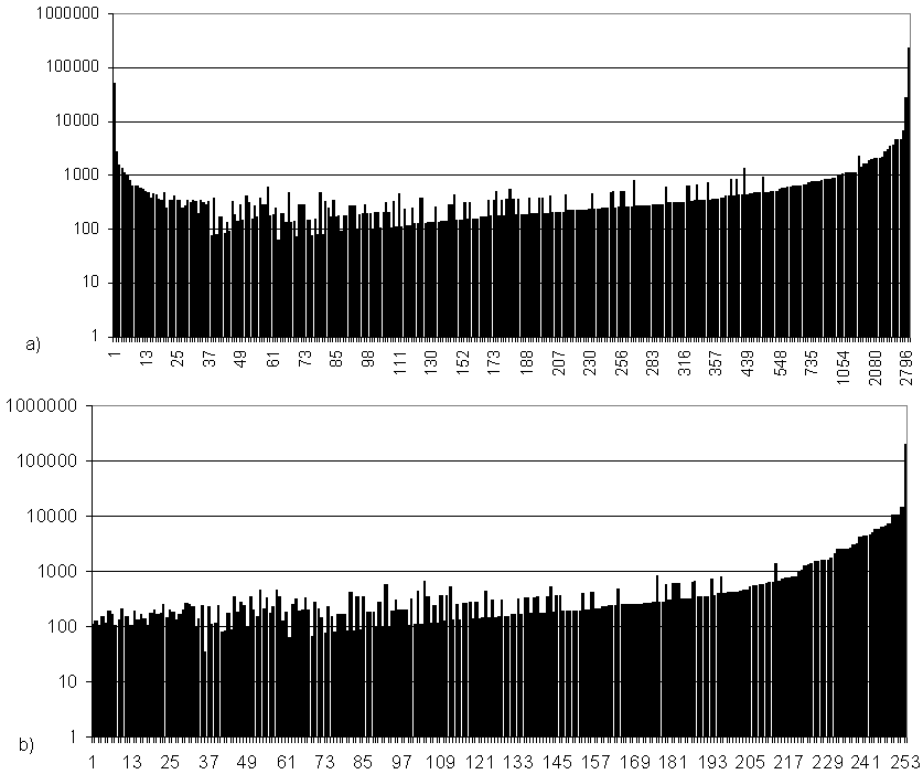

Fig. 3. a) The frequencies of the triples with a given object value frequency for $predicate$ = *took_place*; b) The frequencies of the triples with a given object value frequency for $predicate = not\_after$

most once. The second queries group QG10 comprises Q19-Q35. They are used for filling the time lines in the web application. Therefore, they are executed up to 10 times. The queries contain up to 21 variables, have up to 32 joins, use text search, comparisons, boolean expressions, OPTIONAL and BOUND constructs.

4 The evaluation set up

We evaluated dataset using both disk and memory-based storage models of the two state of the art storage systems: Jena TDB, Jena ARQ, Sesame-memory, Sesame-native. Jena [18] is a Java framework for building semantic web applications. It includes OWL [19] and RDF [14] API, in memory and persistent storage models, SPARQL [20] query engine. ARQ [1] is a general purpose query engine, supporting SPARQL and other query languages, that can utilize several Jena storage models. In our experiments we used ARQ in memory storage model. TDB [2] is a high-performance native storage engine that exploits custom indexing strategy. Sesame [7] is an open source Java framework for storage and querying RDF data. Sesame supports SPARQL and SERQL [6] query languages, memory-based and disk-based storage. We evaluated the systems exploiting their user interfaces.

The evaluation has been performed on AMD Phenom II 2600 Mhz Processor with 8Gb main memory installed.

The data used in the evaluation included Beazley dataset with 16 millions of triples and its reduced version with 10 millions of triples. The dataset loading time, query execution time and total query set times were measured. The query per hour and second per query measures were calculated given that each query in QG10 is executed 10 times while each query in QG1 is executed once. This setting allowed to represent the CLAROS application query mix.

5 The system performance

The data loading times are presented in Table 2. The memory based models ARQ and Sesame-memory were not able to load the complete Beazley dataset. There was insufficient memory for ARQ. The loading into Sesame-memory were terminated after 5 days. The reduced Beazley dataset version was loaded less than in 1 hour all the systems. The Jena memory and storage based models were more efficient in the data loading then the Sesame models.

The query answering times and the other query performance measures are presented in Table 3.

The systems showed performance ranged from 1 millisecond to 74.8 hours per query. The native Sesame storage model was more than 5 times more efficient than its memory storage model. The ARQ was 2 orders of magnitude less efficient than TDB. It was more then order of magnitude more efficient than TDB given that query Q33 was excluded from the query set. This query took ARQ 74.8 hours to execute. Thus, it influenced on the total result. The other 34 queries were completed in 270 seconds.

Table 1. The Beazley query set.

	Variables	Joins	Text search	Ordering	Comparisons	OPTIONAL	BOUND
Q1	2	0					
Q2	11	15	X	X			
Q3	11	15	X	X			
Q4	9	14	X	X			
Q5	6	7	X	X			
Q6	17	10	X	X			
Q7	1	2					
Q8	1	2					
Q9	5	2					
Q10	9	14	2X				
Q11	20	31	2X	X		X	
Q12	11	19	2X	X			
Q13	7	12	X	X			
Q14	6	11	X				
Q15	18	32	X	X		X	
Q16	11	20	X	X			
Q17	11	20	2X				
Q18	19	32	2X	X		X	
Q19	10	14	X				
Q20	14	22	X		2X		
Q21	20	32	X	X	2X		
Q22	12	19	X		2X		
Q23	13	22	X		2X		
Q24	21	32	2X	X	2X		
Q25	17	28	2X		2X		
Q26	12	20	X		2X		
Q27	8	12			2X		
Q28	14	24		X	2X		
Q29	11	20			2X		
Q30	11	19	X		2X		
Q31	18	32	X	X	2X		
Q32	16	28	X	X	2X		
Q33	11	20	X		2X		2X
Q34	15	25	X	X	2X	X	2X
Q35	13	23	X		2X		

Table 2. Loading times, seconds.

	ARQ	TDB	Sesame-native	Sesame-memory
Beazley-16Mt	N/A	1445.68	1878.43	N/A
Beazley-10Mt	360.19	434.07	1087.14	2991.44

Table 3. Query answering times and aggregate performance measures on Beazley dataset

	Query answering times-16Mt,s		Query answering times-10Mt,s			
	Sesame-native	TDB	Sesame-memory	ARQ	Sesame-native	TDB
Q1	0.03	0.05	0.02	0.006	0.03	0.12
Q2	39.82	119.14	210.34	5.96	38.29	171.7
Q3	38.39	125.16	217.98	4.93	37.33	115.91
Q4	26.01	94.86	137.1	26.89	22.2	89.25
Q5	39.9	97.32	128.05	5.11	38.62	90.35
Q6	57.04	139.98	343.98	25.95	56.24	135.24
Q7	0.001	0.09	0.001	0.01	0.001	0.08
Q8	0.002	0.09	0.001	0.01	0.001	0.08
Q9	19.66	70.91	39.02	37.22	11.2	38.52
Q10	16.34	91.5	58.87	25.2	13.08	88.78
Q11	21.57	186.85	93.36	4.22	18.3	197.23
Q12	19.52	105.94	80.26	1.24	16.01	102.27
Q13	18.15	6.99	35.63	1.23	8.39	3.36
Q14	49.66	111.32	209.43	7.17	48.31	110.1
Q15	33.7	105.46	222.81	6.32	29.69	99.14
Q16	31.48	122.49	194.79	5.73	27.13	119.2
Q17	19.69	106.55	81.16	5.84	15.47	99.42
Q18	22.63	173.64	106.07	5.53	18.52	168.48
Q19	32	114	140.64	5.36	31.06	112.16
Q20	25.99	122.97	125.51	5.72	21.55	116.97
Q21	31.61	147.94	182.62	25.64	26.75	149.24
Q22	35.99	133.69	206.71	5.53	38.02	126.08
Q23	0.43	109.18	0.12	5.37	0.4	103.41
Q24	25.71	154.82	116.25	5.09	20.7	142.33
Q25	24.47	139.02	114.79	5.13	18.31	132.38
Q26	18.21	5.75	57.63	9.56	12.31	5.91
Q27	13.77	5.56	46.6	7.48	13.41	5.04
Q28	20.51	12.46	90.98	5.39	14.83	11.92
Q29	19.1	6.38	69.27	5.01	12.66	5.73
Q30	36.75	131.25	207.13	4.78	38.63	123.26
Q31	32.77	148.84	193.96	5.65	27.18	142.03
Q32	30.22	135.05	162.23	4.76	24.15	137.57
Q33	2.39	8.52	0.005	74.8h	0.007	2.41
Q34	3.33	12.91	0.005	4.95	0.008	4.86
Q35	0.48	111.27	0.12	5.01	0.4	100.23
Total	13.45m	52.63m	64.55m	74.87h	11.65m	50.84m
QpH	169.54	40.63	35.05	0.25	198.86	42.71
SpQ	21.23	88.59	102.68	14330.4	18.1	84.28

None of the systems was able to execute the complete query mix on the complete dataset in less that 10 minutes what makes the CLAROS application and, therefore, Beazley dataset challenging for state of the art storage systems.

The quality of the query answering results is affected by quality of the original data input. Making improvements to the incoming data (which are obtained by extraction from existing databases) is an ongoing activity, which the Beazley Archive team are addressing by (a) improving the data extraction processes, (b) by applying heuristics to clean up some of the data values (e.g. dates), (c) highlighting inconsistencies that are detected by the extraction processes and passing these back to the data originators for correction, and (d) use of thesauri and authority lists to map terminology variations to common terms.

6 Conclusion

The query set tested in the paper was used in an initial development of the CLAROS application. Naively constructed, it was designed mainly to provide functionality rather than performance. The new version of the CLAROS application will include an updated query set designed with partners from the Jena team to identify bottlenecks and improve the queries. The goal of this efforts is to redesign queries to achieve sub-second response times. The strategies for the dataset improvement are (1) pre-calculation of certain path queries to reduce run-time joins (roughly equivalent to "materialized views" in relational data), and (2) use of additional indexes associated with "virtual properties" that can reduce the need for in-memory sorting of results when processing SPARQL queries (analogous to schema-defined indexes in relational databases). Essentially, 4 techniques have been used:

1. reordering of queries so that more selective selective elements are evaluated earlier (this can also be performed automatically by the ARQ query processor in Jena);
2. "materialization" of property paths and UNIONS in queries - adding "short cut" properties to the triple store, and use these properties in queries;
3. customized indexes for finding earliest- and latest- occurrences of a given object type, and also for providing consistent ordering in other keyword-based object access queries. These new indexes are not Lucene-based, as originally intended, as Lucene handing of result sorting is less scalable than had been anticipated. Instead, a simple arrangement of flat files named by keywords, with contents sorted by the ordering key is used;
4. pre-calculation of object counts by various categories, so that counting queries can run without having to access every matching object.

Our hope is that this kind of ad-hoc optimization work can suggest ways forward for more principled ontology-based optimization of triple store access. We intend that this revised system will be the basis of a public version of the CLAROS application developed by academic groups who are focused on application of the technologies rather than technology research.

We described a new dataset for storage systems evaluation called Beazley dataset. The dataset proved to be challenging for state of the art storage systems. In fact, none from the systems evaluated was able to demonstrate the level of performance needed for the real world application utilizing the dataset data. The work suggests that Semantic Web technologies applied indiscriminately (or naively) may not always yield acceptable performance, but significant performance improvements are possible through judicious optimizations to the stored data and queries used, without distorting the semantic coherence of the original data. Performance improvement work to date has been ad hoc, but suggests some strategies that might be considered for automated query optimization.

References

1. ARQ - A SPARQL processor for Jena. http://jena.sourceforge.net/ARQ/.
2. TDB - A SPARQL database for Jena. http://jena.sourceforge.net/TDB/.
3. The Billion Triple Challenge. http://challenge.semanticweb.org/.
4. D. Bitton, D. DeWitt, and C. Turbyfill. Benchmarking database systems a systematic approach. In *VLDB '83: Proceedings of the 9th International Conference on Very Large Data Bases*, pages 8–19, San Francisco, CA, USA, 1983. Morgan Kaufmann Publishers Inc.
5. C. Bizer and A. Schultz. The berlin SPARQL benchmark. *Int. J. Semantic Web Inf. Syst.*, 5(2):1–24, 2009.
6. J. Broekstra and A. Kampman. SeRQL: A Second Generation RDF Query Language. In *SWAD-Europe Workshop on Semantic Web Storage and Retrieval*, 2003.
7. J. Broekstra, A. Kampman, and F. van Harmelen. Sesame: A generic architecture for storing and querying RDF and RDF Schema. In Ian Horrocks and James Hendler, editors, *Proceedings of the first Int'l Semantic Web Conference (ISWC 2002)*, volume 2342 of *Lecture Notes in Computer Science*, pages 54–68, Sardinia, Italy, May 2002. Springer Verlag.
8. M. Doerr and D. Iorizzo. The dream of a global knowledge network—a new approach. *J. Comput. Cult. Herit.*, 1(1):1–23, June 2008.
9. T. Gardiner, I. Horrocks, and D. Tsarkov. Automated benchmarking of description logic reasoners. In *Proc. of the 2006 Description Logic Workshop (DL 2006)*, volume 189, 2006.
10. Ian Horrocks Giorgos Stoilos, Bernardo Cuenca Grau. How incomplete is your semantic web reasoner? In *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence (AAAI 2010)*, 2010. To Appear.
11. J. Golbeck, G. Fragoso, F. Hartel, J. Hendler, J. Oberthaler, and B. Parsia. The national cancer institute's thésaurus and ontology. *J. Web Sem.*, 1(1):75–80, 2003.
12. 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(2-3):158 – 182, 2005.
13. I. Horrocks, P. F. Patel-Schneider, and R. Sebastiani. An analysis of empirical testing for modal decision procedures. *Logic Journal of the IGPL*, 8(3):293–323, 2000.
14. G. Klyne and J. Carroll. Resource Description Framework (RDF): Concepts and Abstract Syntax. W3C Recommendation, 10 February 2004.

15. D. Kurtz, G. Parker, D. Shotton, G. Klyne, F. Schroff, A. Zisserman, and Y. Wilks. CLAROS - Bringing Classical Art to a Global Public. *International Conference on e-Science and Grid Computing*, pages 20–27, 2009.
16. M. Ley. Dblp database. http://www.informatik.uni-trier.de/ ley/db/.
17. L. Ma, Y. Yang, Z. Qiu, G. T. Xie, Y. Pan, and S. Liu. Towards a complete OWL ontology benchmark. In *ESWC*, pages 125–139, 2006.
18. B. McBride. Jena: Implementing the RDF Model and Syntax Specification. In *SemWeb*, 2001.
19. P. Patel-Schneider, P. Hayes, and I. Horrocks. OWL Web Ontology Language semantics and abstract syntax. W3C Recommendation, 10 February 2004.
20. E. Prud'hommeaux and A. Seaborne. SPARQL Query Language for RDF. W3C Recommendation, 15 January 2008.
21. A. Rector, W. Nowlan, and S. Kay. Foundations for an electronic medical record. *Methods Inf Med*, 30(3):179–86, 1991.
22. M. Schmidt, T. Hornung, N. Küchlin, G. Lausen, and C. Pinkel. An experimental comparison of RDF data management approaches in a SPARQL benchmark scenario. In *ISWC '08: Proceedings of the 7th International Conference on The Semantic Web*, pages 82–97, Berlin, Heidelberg, 2008. Springer-Verlag.