An Evaluation of Knowledge Base Systems for Large OWL Datasets

Yuanbo Guo, Zhengxiang Pan, and Jeff Heflin

Computer Science and Engineering Department, Lehigh University, Bethlehem, PA18015, USA
{yug2, zhp2, Heflin}@cse.lehigh.edu

Abstract. In this paper, we present an evaluation of four knowledge base systems (KBS) with respect to use in large OWL applications. To our knowledge, no experiment has been done with the scale of data used here. The smallest dataset used consists of 15 OWL files totaling 8MB, while the largest dataset consists of 999 files totaling 583MB. We evaluated two memory-based systems (OWLJessKB and memory-based Sesame) and two systems with persistent storage (database-based Sesame and DLDB-OWL). We describe how we have performed the evaluation and what factors we have considered in it. We show the results of the experiment and discuss the performance of each system. In particular, we have concluded that existing systems need to place a greater emphasis on scalability.

1 Introduction

Various knowledge base systems (KBS) have been developed for processing Semantic Web information. They vary in a number of important ways. Many KBSs are main memory-based while others use secondary storage to provide persistence. Another key difference is the degree of reasoning provided by the KBS. Many systems are incomplete with respect to OWL [2], but still useful because they scale better or respond to queries quickly.

In this paper, we consider the issue of how to choose an appropriate KBS for a large OWL application. Here, we consider a large application to be one that requires the processing of megabytes of data. Generally, there are two basic requirements for such systems. First, the enormous amount of data means that scalability and efficiency become crucial issues. Second, the system must provide sufficient reasoning capabilities to support the semantic requirements of the application. However, increased reasoning capability usually means an increase in query response time as well. An important question is how well existing systems support these conflicting requirements. Furthermore, different applications may place emphasis on different requirements.

It is difficult to evaluate KBSs with respect to these requirements, particularly in terms of scalability. The main reason for this is that there are few Semantic Web data sets that are of large size and commit to semantically rich ontologies. The Lehigh

S.A. McIlraith et al. (Eds.): ISWC 2004, LNCS 3298, pp. 274–288, 2004.

University Benchmark [12] is our first step in order to fill this gap. In this work, we have made a further step: by making use of the benchmark, we have evaluated four KBSs for the Semantic Web from several different aspects. We have evaluated two memory-based systems (OWLJessKB and memory-based Sesame) and two systems with persistent storage (database-based Sesame and DLDB-OWL). We present our experiment, discuss the performance of each system, and show some interesting observations. Based on that, we highlight some issues with respect to the development and improvement of the same kind of systems, and suggest some potential ways in using and developing those systems. We also discuss some issues related to the evaluation of Semantic Web KBSs.

The outline of the paper is as follows: Section 2 briefly introduces the aforementioned Lehigh University Benchmark. Section 3 describes the target systems. Section 4 discusses the results. Section 5 talks about some related work. Section 6 concludes.

2 Lehigh University Benchmark for OWL

The Lehigh University Benchmark [12] was originally developed to evaluate the performance of Semantic Web repositories with respect to extensional queries over a large DAML+OIL [9] data set that commits to a single realistic ontology. For this paper, we extended the benchmark to provide support for OWL ontologies and datasets. The benchmark suite for OWL consists of the following:

- A plausible OWL ontology named univ-bench[1] for the university domain.
- Repeatable synthetic OWL data sets that can be scaled to an arbitrary size. Both the univ-bench ontology and the data are in the OWL Lite sublanguage.
- Fourteen test queries that cover a range of types in terms of properties including input size, selectivity, complexity, assumed hierarchy information, and assumed inference, etc. (Refer to the appendix for a list of them).
- A set of performance metrics including data loading time, repository size, query response time, and query answer completeness and soundness. With the exception of completeness, they are all standard database benchmarking metrics [3, 4, 8, 25]. Completeness is described in Section 3.2.
- The test module.

In addition to the language change, the major differences from the original benchmark include: one more query (Query 14); more individuals in the data sets to be classified; more RDFS vocabulary used in the ontology (e.g., *rdfs:domain*); and some domain constraint changes to allow emphasis on description logic subsumption.

Using this benchmark, we have conducted an experiment on the aforementioned systems. We describe it in next section. The benchmark suite is accessible at http://www.lehigh.edu/~yug2/Research/SemanticWeb/LUBM/LUBM.htm.

[1] http://www.lehigh.edu/~zhp2/2004/0401/univ-bench.owl

3 The Experiment

3.1 Target Systems

In this experiment, we wanted to evaluate the scalability and support for OWL Lite in various systems. We believe a practical KBS must be able to read OWL files, support incremental data loading, and provide programming APIs for loading data and issuing queries. As a result, we have settled on four different knowledge base systems, including two implementations of Sesame, OWLJessKB, and DLDB-OWL. We briefly describe each system below.

Sesame [6] is a repository and querying facility based on RDF and RDF Schema [27]. It features a generic architecture that separates the actual storage of RDF, functional modules offering operation on this RDF, and communication with these functional modules from outside the system. Sesame supports RDF/RDF Schema inference, but is an incomplete reasoner for OWL Lite. It can evaluate queries in SeRQL, RQL and RDQL. We evaluate two implementations of Sesame, main memory-based and database-based.

OWLJessKB [22] is a memory-based reasoning tool for description logic languages, particularly OWL. It uses the Java Expert System Shell (Jess) [19], a production system, as its underlying reasoner. Current functionality of OWLJessKB is close to OWL Lite plus some. We evaluate it as a system that supports most OWL entailments.

The fourth system, DLDB-OWL [23], is a repository for processing, storing, and querying large amounts of OWL data. Its major feature is the extension of a relational database system with description logic inference capabilities. Specifically, DLDB-OWL uses Microsoft Access® as the DBMS and FaCT [16] as the OWL reasoner. It uses the reasoner to precompute subsumption and employs relational views to answer extensional queries based on the implicit hierarchy that is inferred.

Originally, we had targeted four other systems. The first is Jena [18], a Java framework for building Semantic Web applications. Jena currently supports both RDF/RDFS and OWL. We have done some preliminary tests on Jena (v2.1) (both memory-based and database-based) with our smallest data set. Compared to Sesame, the most similar system to Jena here, Jena with RDFS reasoning was much slower in answering nearly all the queries. Some of the queries did not terminate even after being allowed to run for several hours. The situation was similar when Jena's OWL reasoning was turned on. For this reason, and also due to space constraints, we have decided not to include Jena in this paper. Those who are interested in more details are referred to [13] instead. The second is KAON [20], an ontology management infrastructure. KAON provides an API for manipulating RDF models, however, it does not directly support OWL or RDFS in its framework. We had also considered TRIPLE [24] and Racer. TRIPLE is an RDF query, inference, and transformation language and architecture. Instead of having a built-in semantics for RDF Schema, TRIPLE allows the semantics of languages on top of RDF to be defined with rules. For languages where this is not easily possible, TRIPLE also provides access to external programs like description logic classifiers. We were unable to test TRIPLE

because it does not support incremental file loading and it does not provide a programming API either. Racer [15] is a description logic inference engine currently supporting RDF, DAML+OIL and OWL. Running as a server, Racer provides inference services via HTTP or TCP protocol to client applications. Racer's query interface predefines some queries and query patterns, but these are insufficient for the test queries in the benchmark. Furthermore, there is no API in Racer for importing customized queries. Thus we were unable to test Racer either.

3.2 Experiment Methodology

System Setup

The systems we test are DLDB-OWL (04-03-29 release), Sesame v1.0, and OWLJessKB (04-02-23 release). As noted, we test both the main memory-based and database-based implementations of Sesame. For brevity, we hereafter refer to them as Sesame-Memory and Sesame-DB respectively. For both of them, we use the implementation with RDFS inference capabilities. For the later, we use MySQL (v4.0.16) as the underlying DBMS since it is reported that Sesame performs best with it. The DBMS used in DLDB-OWL is MS Access® 2002. We have created a wrapper over each target system as an interface to the benchmark's test module.

Data Sets and Loading

To identify the data set, we use LUBM(N, S) in the subsequent description to denote the data set that contains N universities beginning at University0 and is generated by the benchmark tool using a seed value of S. (Readers are referred to [12] for details about synthetic data generation in the benchmark.)

We have created 5 sets of test data: LUBM(1, 0), LUBM(5, 0), LUBM(10, 0), LUBM(20, 0), and LUBM(50, 0), which contain OWL files for 1, 5, 10, 20, and 50 universities respectively, the largest one having over 6,800,000 triples in total. To our knowledge, prior to this experiment, Sesame has been tested with at most 3,000,000 statements. We have easily exceeded that by virtue of the benchmark supporting tool.

In the test data, every university contains 15 to 25 departments, each described by a separate OWL file. These files are loaded to the target system in an incremental fashion. We measure the elapsed time for loading each data set, and also the consequent database sizes for Sesame-DB and DLDB-OWL. We do not measure the occupied memory sizes for Sesame-Memory and OWLJessKB because it is difficult to accurately calculate them. However, since we evaluate all systems on a platform with a fixed memory size, the largest data set that can be handled by a system measures its memory efficiency.

Query Test

For query testing, the 14 benchmark queries are expressed in RQL [21], Jess, and a KIF[11]-like language (see the appendix) and issued to Sesame, OWLJessKB, and DLDB-OWL respectively. We do not use a common language in the test to eliminate affect of query translation to the query response time.

Query response time is collected in the way defined by the benchmark, which is based on the process used in database benchmarks [3, 4, 8, 25]. To account for

caching, each of the fourtheen queries is executed for ten times consecutively and the average time is computed.

We also examine query answer completeness of each system. In logic, an inference procedure is complete if it can find a proof for any sentence that is entailed by the knowledge base. With respect to queries, we say a system is complete if it generates all answers that are entailed by the knowledge base, where each answer is a binding of the query variables that results in an entailed sentence. However, on the Semantic Web, partial answers will often be acceptable. So it is important not to measure completeness with such a coarse distinction. Instead, we measure the degree of completeness of each query answer as the percentage of the entailed answers that are returned by the system. Note that we request that the result set contains unique answers.

In addition, as we will show in next section, we have realized in this evaluation that query soundness is also worthy of examination. With similar argument to the above, we in this evaluation measure the degree of soundness of each query answer as the percentage of the answers returned by the system that are actually entailed.

Test Environment

We have done the test on a desktop computer. The environment is as follows:

- 1.80GHz Pentium 4 CPU; 256MB of RAM; 80GB of hard disk
- Windows XP Professional OS; Java SDK 1.4.1; 512MB of max heap size

In order to evaluate OWLJessKB, we needed to adjust this configuration slightly. With the standard setting for max heap size in Java, the system failed to load the one-university data set due to out of memory errors. As a workaround, we increased the maximum heap size to 1GB, which requests large amount of virtual memory from operating system. This change allowed OWLJessKB to properly load the dataset.

4 Results and Discussions

4.1 Data Loading

Table 1 shows the data loading time for all systems and the on-disk repository sizes of DLDB-OWL and Sesame-DB. Fig. 1 depicts how the data loading time grows as the data set size increases and compares the repository sizes of the two database-based systems.

The test results have reinforced scalability as an important issue and challenge for Semantic Web knowledge base systems. One of the first issues is how large of a data set each system can handle. As expected, the memory-based systems did not perform as well as the persistent storage systems in this regard. OWLJessKB, could only load the 1-university data set, and took over 20 times longer than any other system to do so. On the other hand, we were surprised to see that Sesame-Memory could load up to 10 universities, and was able to do it in 5% of the time of the next fastest system. However, for 20 or more universities, Sesame-Memory also succumbed to memory limitations.

Table 1. Load Time and Repository Sizes

	Data Set	Instance Num	Load Time (hh:mm:ss)	Repository Size (KB)
DLDB-OWL	LUBM (1, 0)	103,074	00:05:43	16,318
Sesame-DB			00:09:02	48,333
Sesame-Memory			00:00:13	-
OWLJessKB			03:16:12	-
DLDB-OWL	LUBM (5, 0)	645,649	00:51:57	91,292
Sesame-DB			03:00:11	283,967
Sesame-Memory			00:01:53	-
OWLJessKB			-	-
DLDB-OWL	LUBM (10, 0)	1,316,322	01:54:41	184,680
Sesame-DB			12:27:50	574,554
Sesame-Memory			00:05:40	-
OWLJessKB			-	-
DLDB-OWL	LUBM (20, 0)	2,781,322	04:22:53	388,202
Sesame-DB			46:35:53	1,209,827
Sesame-Memory			-	-
OWLJessKB			-	-
DLDB-OWL	LUBM (50, 0)	6,888,642	12:37:57	958,956
Sesame-DB			-	-
Sesame-Memory			-	-
OWLJessKB			-	-

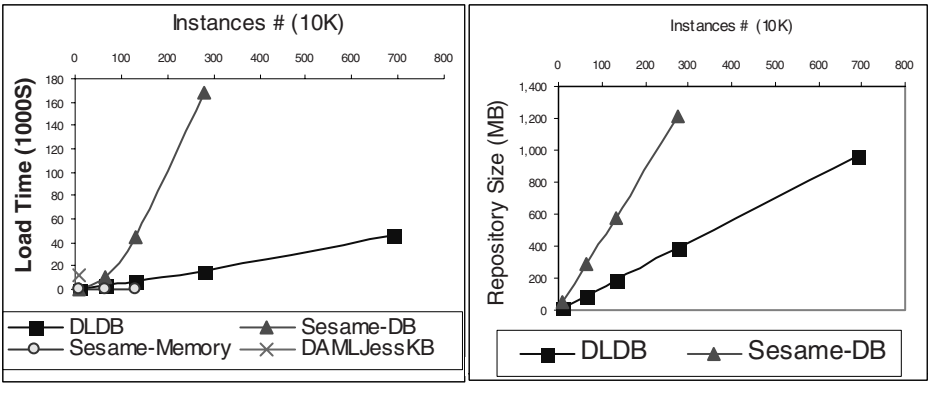

Fig. 1. Load Time and Repository Sizes. The left hand figure shows the load time. The right hand figure shows the repository sizes of the database-based systems.

Using the benchmark, we have been able to test both Sesame-Memory and Sesame-DB on larger scale data sets than what has been reported so far. The result reveals an apparent problem for Sesame-DB: it does not scale in data loading, as can be seen from Fig. 2. As an example, it took over 300 times longer to load the 20-university data set than the 1-university data set, although the former set contains only about 25 times more instances than the later. We extrapolate that it will take Sesame-DB over 3 weeks to finish up loading the 50-university data set. Therefore, we have decided not to do that unrealistic test.

In contrast, DLDB-OWL displays good scalability in data loading. We suspect the different performance of the two systems is caused by the following two reasons. First, to save space, both DLDB-OWL and Sesame map resources to unique IDs maintained

in a table. When a resource is encountered during the data loading, they will look up that table to determine if it has not been seen before and need to be assigned a new ID. As mentioned in [23], querying the ID table every time is very likely to slow down the data loading as the data size grows. In its implementation, Sesame also assigns every literal an ID, while DLDB-OWL stores literals directly in the destination tables, which means Sesame has to spend even more time on ID lookup. Moreover, in order to improve performance, DLDB-OWL caches resource-ID pairs during current loading.

A second reason for the performance difference is related to the way Sesame performs inference. Sesame is a forward-chaining reasoner, and in order to support statement deletions it uses a truth maintenance system to track all deductive dependencies between statements. As [5] shows, this appears to affect the performance significantly if there are many inferred statements or the data set is fairly large. We should note that this scalability problem was not as noticeable in our previous study involving a DAML+OIL benchmark [14]. We believe this is because the prior experiment used *daml:domain* (as opposed to *rdfs:domain*) in its ontology, which does not trigger inferences in Sesame.

4.2 Query Response Time

Table 2 is a complete list of the query test results, including the query response time, number of answers, and their completeness. Note that what are displayed in each answer row are only the numbers of correct answers (Refer to Section 4.3). Fig. 2 compares by graphs the query response time of the systems except OWLJessKB.

In terms of query, the results also lead to some scalability and efficiency concerns. Although compared to the performance of its predecessor DAMLJessKB [22] in [14], OWLJessKB improves its query time greatly at the sacrifice of much longer loading time, it is still the slowest in answering thirteen of the queries. Sesame-DB was also very slow in answering some queries (even for one university), including Queries 2, 8, and 9. As for DLDB-OWL, it is the only system that has been tested with the largest data set. One concern is that when it comes to the larger data sets especially the 50-university set, DLDB-OWL's query time no longer grows linearly for some queries, i.e., Queries 2, 5, 6, 7, 9, and 14. Moreover, it failed to answer Query 2 on the 50-univeristy data set after MS Access ran out of temporary space. Compared to other systems, Sesame-Memory, is the fastest in answering all of the queries. It is also the fastest in data loading. This suggests that it might be the best choice for data of small scale if persistent storage and OWL inference is not required.

We have observed that those queries for which Sesame-DB's performance goes down dramatically are common in that they do not contain a specific URI as a subject or object in the statements. On the other hand, Sesame-DB shows a nice property in answering some other queries like Queries 3, 4, 5, 7, and 8: there was no proportional increase in the response time as the data size grows. We have also noticed a common feature of these queries, i.e., they have constant number of results over the test data sets. Whether these are the causes or coincidences is a subject for future work.

It is beyond the scope of this paper to analyze in depth the query evaluation and optimization mechanism in each system. Instead, we propose some topics for future investigation. One is to explore the potential relationship between query types and the

Table 2. Query Test Results. (* correct answers only, refer to Table 3)

Query	Repository & Data Set / Metrics	LUBM(1,0)				LUBM(5,0)			LUBM(10,0)			LUBM(20,0)		LUBM (50,0)
		DLDB-OWL	Sesame-DB	Sesame-Mem-	OWL JessKB	DLDB-OWL	Sesame-DB	Sesame-Mem-	DLDB-OWL	Sesame-DB	Sesame-Mem-	DLDB-OWL	Sesame-DB	DLDB-OWL
1	Time(ms)	59	46	15	9203	226	43	37	412	40	106	887	96	2211
	Answers	4	4	4	4	4	4	4	4	4	4	4	4	4
	Completeness	100	100	100	100	100	100	100	100	100	100	100	100	100
2	Time(ms)	181	51878	87	116297	2320	368423	495	14556	711678	1068	392392	1474664	failed
	Answers	0	0	0	0	9	9	9	28	28	28	59	59	-
	Completeness	100	100	100	100	100	100	100	100	100	100	100	100	-
3	Time(ms)	218	40	0	13990	2545	53	1	5540	59	0	11956	56	36160
	Answers	6	6	6	6	6	6	6	6	6	6	6	6	6
	Completeness	100	100	100	100	100	100	100	100	100	100	100	100	100
4	Time(ms)	506	768	6	211514	2498	823	4	5615	762	4	14856	881	10115
	Answers	34	34	34	34*	34	34	34	34	34	34	34	34	34
	Completeness	100	100	100	100	100	100	100	100	100	100	100	100	100
5	Time(ms)	617	2945	17	5929	4642	3039	17	11511	3214	17	27756	3150	135055
	Answers	719	719	719	719	719	719	719	719	719	719	719	719	719
	Completeness	100	100	100	100	100	100	100	100	100	100	100	100	100
6	Time(ms)	481	253	48	1271	4365	1517	251	11158	3539	543	28448	12717	151904
	Answers	7790	5916	5916	7790*	48582	36682	36682	99566	75547	75547	210603	160120	519842
	Completeness	100	76	76	100	100	76	76	100	76	76	100	76	100
7	Time(ms)	478	603	3	128115	2639	606	4	7028	634	4	18073	657	121673
	Answers	67	59	59	67	67	59	59	67	59	59	67	59	67
	Completeness	100	88	88	100	100	88	88	100	88	88	100	88	100
8	Time(ms)	765	105026	273	164106	3004	108384	262	5937	108851	264	13582	103779	39845
	Answers	7790	5916	5916	7790*	7790	5916	5916	7790	5916	5916	7790	5916	7790
	Completeness	100	76	76	100	100	76	76	100	76	76	100	76	100
9	Time(ms)	634	34034	89	87475	7751	256770	534	19971	460267	1123	57046	1013951	323579
	Answers	208	103	103	208	1245	600	600	2540	1233	1233	5479	2637	13639
	Completeness	100	50	50	100	100	48	48	100	49	49	100	48	100
10	Time(ms)	98	20	1	141	1051	36	0	2339	40	0	5539	50	15831
	Answers	4	0	0	4	4	0	0	4	0	0	4	0	4
	Completeness	100	0	0	100	100	0	0	100	0	0	100	0	100
11	Time(ms)	48	65	1	1592	51	73	1	61	84	3	78	82	143
	Answers	0	0	0	224	0	0	0	0	0	0	0	0	0
	Completeness	0	0	0	100	0	0	0	0	0	0	0	0	0
12	Time(ms)	62	4484	12	11266	78	4659	14	123	4703	12	310	4886	745
	Answers	0	0	0	15*	0	0	0	0	0	0	0	0	0
	Completeness	0	0	0	100	0	0	0	0	0	0	0	0	0
13	Time(ms)	200	4	1	90	2389	9	1	5173	12	1	11906	21	34854
	Answers	0	0	0	1	0	0	0	0	0	0	0	0	0
	Completeness	0	0	0	100	0	0	0	0	0	0	0	0	0
14	Time(ms)	187	218	42	811	2937	1398	257	7870	14021	515	19424	11175	106764
	Answers	5916	5916	5916	5916	36682	36682	36682	75547	75547	75547	160120	160120	393730
	Completeness	100	100	100	100	100	100	100	100	100	100	100	100	100

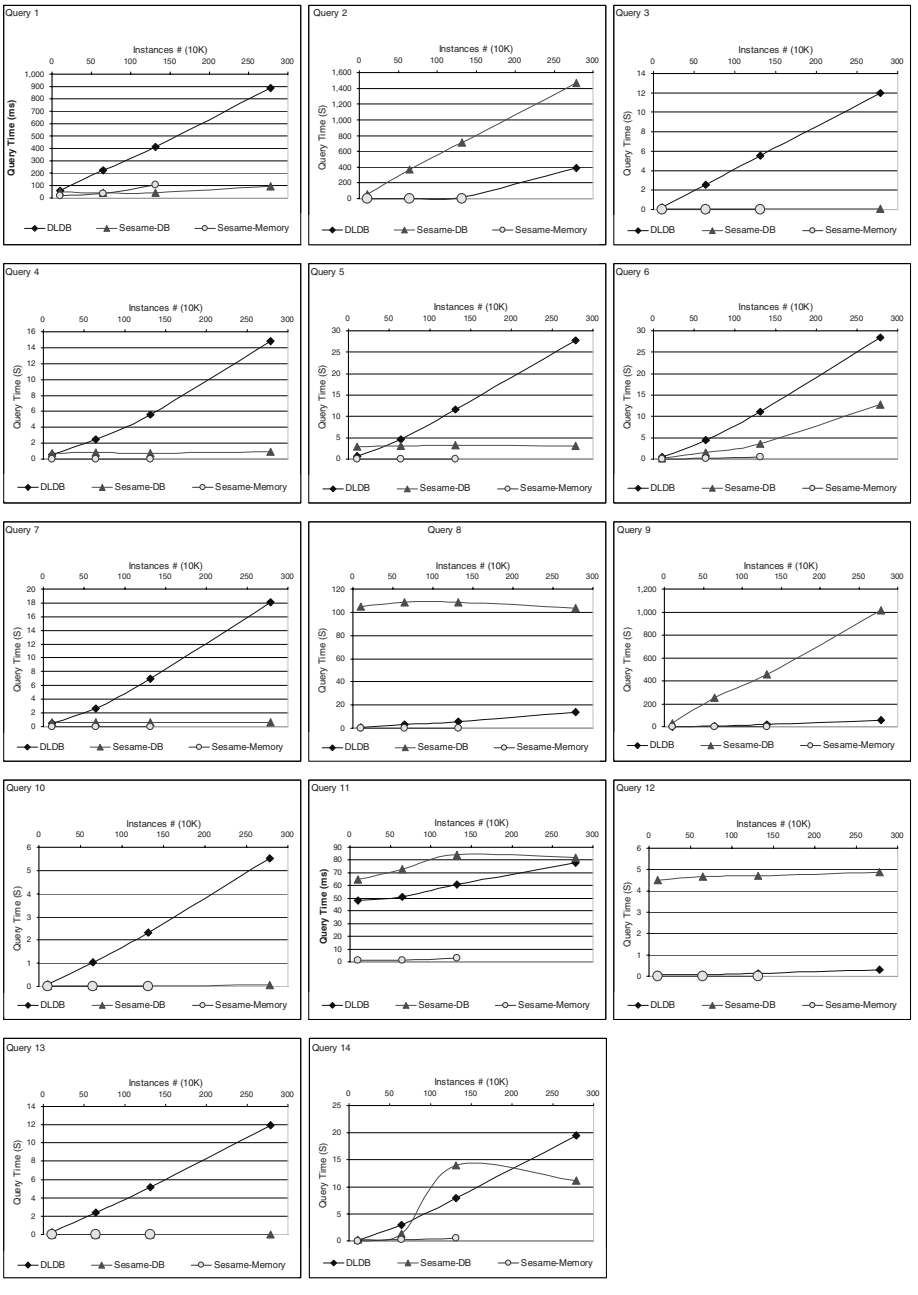

Fig. 2. Query time of DLDB-OWL, Sesame-DB, and Sesame-Memory (up to 20 universities)

performance of a certain system and its characteristics. Of course how to categorize queries is yet another issue. As another, Sesame-DB implements the main bulk of the evaluation in its RQL query engine while its query engine for another query language

SeRQL pushes a lot of the work down to the underlying DBMS. As for DLDB-OWL, it directly translates as much of the query for the database. Further work should be done to investigate how these design differences as well as the underlying DBMS used impact performance.

4.3 Query Completeness and Soundness

As described in [12], we have chosen the benchmark test queries according to several criteria. In fact, another effort we have made in defining these queries is to make them as realistic as possible. In other words, we want these queries to represent, to some extent, those in the real world. We are very interested in seeing what queries can be answered by each system.

As mentioned before, Sesame is able to address RDF/RDFS semantics while DLDB-OWL and OWLJessKB integrate extra OWL inference capability. As the results turned out, all systems could answer Queries 1 through 5 and Query 14 completely. As we expected, DLDB-OWL was able to find all the answers for Queries 6 to 10, which requires subsumption inference in order to get complete results, while Sesame could only find partial or no answers for them. It is interesting to notice that DLDB-OWL and Sesame found complete answers for Query 5 in different ways: DLDB-OWL made use of subsumption, while Sesame, although not able to figure out the subsumption, used an *rdfs:domain* restriction to determine the types of the individuals in the data set and thus achieved the same result. OWLJessKB could find all the answers for every query, and was the only system to answer Queries 11 and 13 completely, which assume *owl:TransitiveProperty* and *owl:inverseOf* inference respectively. Nevertheless, we have discovered that OWLJessKB made unsound inferences with respect to some queries. Specifically, it returned incorrect answers to Queries 4, 6, 8, and 12 because it incorrectly inferred that Lecturer is a Professor, Employee a Student, and Student a Chair. We list in Table 3 the completeness and soundness of OWLJessKB for each query.

Table 3. Query Soundness of OWLJessKB.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Total/ Correct answers	4/4	0/0	6/6	41/34	719/719	8330/ 7790	67/67	8330/ 7790	208/208	4/4	224/224	540/15	1/1	5916/5916
Soundness	100	100	100	83	100	94	100	94	100	100	100	3	100	100

4.4 A Combined Metric

As the previous section shows, the target systems in this evaluation differ a lot in their inference capability. We feel it is insufficient to evaluate the query response time and answer completeness and soundness in isolation. We need a metric to measure them in combination so as to better appreciate the overall performance of a system and the potential tradeoff between the query response time and inference capability. At the same time, we have realized that this is a challenging issue. We introduce here our initial attempt to address this issue.

First, we use the F-Measure metric to compute the tradeoff between query completeness and soundness, since essentially they are analogous to recall and precision in Information Retrieval. In the formula below, C_q and S_q ($\in$ [0, 1]) are the answer completeness and soundness for query q. b determines the weighting of C_q and S_q. We set it to 1 here, which means we equally weight completeness and soundness.

$$F_q = \frac{2b * C_q * S_q}{b^2 * C_q + S_q}$$

Then, we define a composite metric CM of query response time and answer completeness and soundness as the following, which is also inspired by F-Measure:

$$CM = \frac{1}{M} \sum_{q=1}^{M} \frac{2\alpha * P_q * F_q}{\alpha^2 * P_q + F_q}$$

In the above, M is the total number of test queries; $P_q \in [0, 1]$ is defined as

$$P_q = \max\left(1 - \frac{T_q}{N}, \varepsilon\right)$$

T_q is the response time (ms) for query q and N is the total number of instances in the data set concerned. We have used a timeout value to eliminate undue affect of those query response time that is extremely far away from others in the test results: if to a certain query q, a system's response time per instance is greater than 1- ε, where ε is a very small positive value, we will use ε for P_q instead. We use ε of 0.0001 in this evaluation. α has the same role as b in F_q and is also set to 1.

Generally speaking, the CM metric will reward those systems that can answer queries faster, more completely and more soundly. We calculate the metric value of each target system with respect to each data set. Fig. 3 shows the results. We find that these numerical results are very helpful for us to appreciate the overall performance of each system. DLDB-OWL achieves higher scores across all the data sets than the others. This helps us believe that its extra inference capability is not counterproductive. On the contrary, OWLJessKB receives the lowest value, emphasizing the need of performance improvement in it. And the higher CM values Sesame-Memory gets than Sesame-DB again suggest that it is a reasonable choice for small scale application if persistent storage is not required, particularly if completeness is not significant.

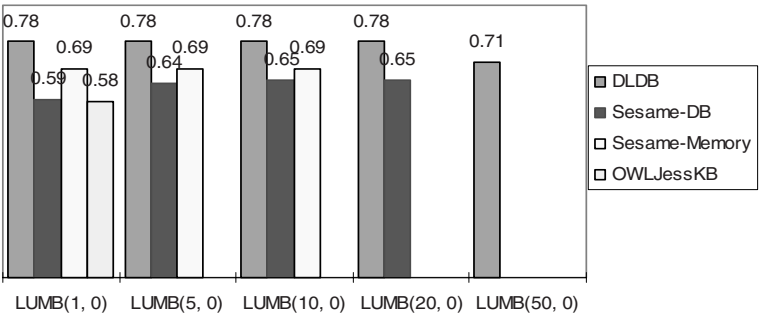

Fig. 3. CM values with weights b=1 and α=1

5 Related Work

To the best of our knowledge, the Lehigh University Benchmark we used in the evaluation is the first one for Semantic Web systems. [1] has developed some benchmark queries for RDF, however, these are mostly intensional queries, while we are concerned with extensional queries for OWL. Some attempts have been done to benchmark description logic systems [10, 17]. The emphasis of this work is to evaluate the reasoning algorithms in terms of the tradeoff between expressiveness and tractability in description logic. Our benchmark is not a description logic benchmark. We are more concerned about the issue of storing and querying large amount of data that are created for realistic Semantic Web systems. Detailed discussion on related work to the benchmark can be found in [12].

The Web Ontology Working Group provides a set of OWL test cases [7]. They are intended to provide examples for, and clarification of, the normative definition of OWL and focus on the completeness and soundness with respect to individual features. Different from our benchmark suite, they are not suitable for the evaluation of scalability.

Tempich and Volz [26] have done some preliminary work towards a benchmark for Semantic Web reasoners. They also point out that the response time as well as the correctness and completeness should be taken into account when formulating benchmark metrics. Though their benchmark is still under construction, they analyze the publicly available ontologies and report them to be clustered into three categories. According to the characteristics of each category, our univ-bench ontology happens to be a synthetic "description logic-style" ontology, which has a moderate number of classes but several restrictions and properties per class. Therefore we argue that our evaluation represents at least a considerable portion of the real word situations. The other two categories are terminological ontologies and database schema-like ontologies. We plan to extend our benchmark suite to those two categories in the future.

6 Conclusions

We presented an evaluation of four knowledge base systems (KBS) with respect to use in large OWL applications, including two memory–based systems (OWLJessKB and memory-based Sesame) and two systems with persistent storage (database-based Sesame and DLDB-OWL). The experiment was conducted in a systematic and standard way by using the Lehigh University Benchmark. We tested those systems with 5 sets of benchmark data and 14 extensional queries. To our knowledge, no experiment has been done with the scale of data used here. The smallest data size used consists of 15 OWL files totaling 8MB, while the largest data size consists of 999 files totaling 583MB.

It is clear that a number of factors must be considered when evaluating a KBS. From our analysis, of the systems tested: DLDB is the best for large data sets where an equal emphasis is placed on query response time and completeness. Sesame-Memory

is the best when the size is relatively small (e.g., 1 million triples) and only RDFS inference is required; while for a larger data set (e.g., between 1 and 3 million triples), Sesame-DB may be a good alternative. OWLJessKB is the best for small datasets when OWL Lite reasoning is essential, but only after its unsoundness has been corrected.

It should be pointed out that we believe that the performance of any given system will vary depending on the structure of the ontology and data used to evaluate it. Thus the Lehigh University Benchmark does not provide the final say on what KBS to use for an application. However, we believe that is appropriate for a large class of applications. Furthermore, the basic methodology can be used to generate ontologies and datasets for other classes of applications.

Acknowledgements. Some of the material in this paper is based upon work supported by the Air Force Research Laboratory, Contract Number F30602-00-C-0188 and by the National Science Foundation (NSF) under Grant No. IIS-0346963. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force or NSF.

References

1. Alexaki, S. et al. On Storing Voluminous RDF Description: The case of Web Portal Catalogs. In Proc. of the 4th International Workshop on the Web and Databases. 2001.
2. M. Dean and G. Schreiber ed. OWL Web Ontology Language Reference. http://www.w3.org/TR/owl-ref/
3. Bitton, D., DeWitt, D., and Turbyfill, C. Benchmarking Database Systems, a Systematic Approach. In Proc. of the 9th International Conference on Very Large Data Bases. 1983
4. Bitton, D. and Turbyfill, C. A Retrospective on the Wisconsin Benchmark. In Readings in Database Systems, Second Edition. 1994.
5. Broekstra, J. and Kampman, A. Inferencing and Truth Maintenance in RDF Schema: exploring a naive practical approach. In Workshop on Practical and Scalable Semantic Systems (PSSS). 2003.
6. Broekstra, J. and Kampman, A. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In Proc. of ISWC2002.
7. Carroll, J.J. and Roo, J.D. ed. OWL Web Ontology Test Cases. http://www.w3.org/TR/2004/REC-owl-test-20040210/
8. Cattell, R.G.G. An Engineering Database Benchmark. In Readings in Database Systems, Second Edition. 1994.
9. Connolly, D. et al. DAML+OIL (March 2001) Reference Description. http://www.w3.org/TR/daml+oil-reference
10. Elhaik, Q, Rousset, M-C, and Ycart, B. Generating Random Benchmarks for Description Logics. In Proc. of DL'98.
11. Gensereth, M. and Fikes, R. Knowledge Interchange Format. Stanford Logic Report Logic-92-1, Stanford Univ. http://logic.standford.edu/kif/kif.html
12. Guo, Y., Heflin, J., and Pan, Z. Benchmarking DAML+OIL Repositories. In Proc. of ISWC2003.

13. Guo, Y., Heflin, J., and Pan, Z. An Evaluation of Knowledge Base Systems for Large OWL Datasets. Technical report, CSE department, Lehigh University. 2004. To appear.
14. Guo, Y., Pan, Z. and Heflin, J. Choosing the Best Knowledge Base System for Large Semantic Web Applications. Poster paper at WWW2004.
15. Haarslev, V. and Moller, R. Racer: A Core Inference Engine for the Semantic Web. In Workshop on Evaluation on Ontology-based Tools, ISWC2003.
16. Horrocks, I. The FaCT System. In Automated Reasoning with Analytic Tableaux and Related Methods International Conference (Tableaux'98).
17. Horrocks, I. and Patel-Schneider, P. DL Systems Comparison. In Proc. of DL' 98.
18. Jena – A Semantic Web Framework for Java. http://jena.sourceforge.net/
19. Jess: the Rule Engine for the Java Platform. http://herzberg.ca.sandia.gov/jess
20. KAON: The KArlsruhe ONtology and Semantic Web tool suite. http://kaon.semanticweb.org/
21. Karvounarakis, G. et al. Querying Community Web Portals. http://www.ics.forth.gr/proj/isst/RDF/RQL/rql.pdf
22. Kopena, J.B. and Regli, W.C. DAMLJessKB: A Tool for Reasoning with the Semantic Web. In Proc. of ISWC2003.
23. Pan, Z. and Heflin, J. DLDB: Extending Relational Databases to Support Semantic Web Queries. In Workshop on Practical and Scalable Semantic Systems, ISWC2003.
24. Sintek, M. and Decker, S. TRIPLE – A Query, Inference, and Transformation Language for the Semantic Web. In Proc. of ISWC2002.
25. Stonebraker, M. et al. The SEQUIOA 2000 Storage Benchmark. In Readings in Database Systems, Second Edition. 1994.
26. Tempich, C. and Volz, R. Towards a benchmark for Semantic Web reasoners–an analysis of the DAML ontology library. In Workshop on Evaluation on Ontology-based Tools, ISWC2003.
27. W3C. Resource Description Framework (RDF). http://www.w3.org/RDF/

Appendix: Test Queries

We herein describe each query in a KIF like language, in which a query is written as a conjunction of atoms. Following that we describe the characteristics of the query.

Query1

(type GraduateStudent ?X)
(takesCourse ?X http://www.Department0.University0.edu/GraduateCourse0)

This query bears large input and high selectivity. It queries about just one class and one property and does not assume any hierarchy information or inference.

Query2

(type GraduateStudent ?X) (type University ?Y) (type Department ?Z)
(memberOf ?X ?Z) (subOrganizationOf ?Z ?Y) (undergraduateDegreeFrom ?X ?Y)

This query increases in complexity: 3 classes and 3 properties are involved. Additionally, there is a triangular pattern of relationships between the objects involved.

Query3

(type Publication ?X)
(publicationAuthor ?X http://www.Department0.University0.edu/AssistantProfessor0)

This query is similar to Query 1 but class Publication has a wide hierarchy.

Query4

(type Professor ?X) (worksFor ?X http://www.Department0.University0.edu)
(name ?X ?Y1) (emailAddress ?X ?Y2) (telephone ?X ?Y3)

This query has small input and high selectivity. It assumes *subClassOf* relationship between Professor and its subclasses. Class Professor has a wide hierarchy. Another feature is that it queries about multiple properties of a single class.

Query5

(type Person ?X) (memberOf ?X http://www.Department0.University0.edu)

This query assumes *subClassOf* relationship between Person and its subclasses and *subPropertyOf* relationship between memberOf and its subproperties. Moreover, class Person features a deep and wide hierarchy.

Query6

(type Student ?X)

This query queries about only one class. But it assumes both the explicit *subClassOf* relationship between UndergraduateStudent and Student and the implicit one between GraduateStudent and Student. In addition, it has large input and low selectivity.

Query7

(type Student ?X) (type Course ?Y)
(teacherOf http://www.Department0.University0.edu/AssociateProfessor0 ?Y)(takesCourse ?X ?Y)

This query is similar to Query 6 in terms of class Student but it increases in the number of classes and properties and its selectivity is high.

Query8

(type Student ?X) (type Department ?Y) (memberOf ?X ?Y)
(subOrganizationOf ?Y http://www.University0.edu) (emailAddress ?X ?Z)

This query is further more complex than Query 7 by including one more property.

Query9

(type Student ?X) (type Faculty ?Y) (type Course ?Z)
(advisor ?X ?Y) (takesCourse ?X ?Z) (teacherOf ?Y ?Z)

Besides the aforementioned features of class Student and the wide hierarchy of class Faculty, this query is characterized by the most classes and properties in the query set. Like Query 2, there is a triangular pattern of relationships.

Query10

(type Student ?X) (takesCourse ?X http://www.Department0.University0.edu/GraduateCourse0)

This query differs from Query 6, 7, 8 and 9 in that it only requires the (implicit) *subClassOf* relationship between GraduateStudent and Student, i.e., *subClassOf* relationship between UndergraduateStudent and Student does not add to the results.

Query11

(type ResearchGroup ?X) (subOrganizationOf ?X http://www.University0.edu)

Query 11, 12 and 13 are intended to verify the presence of certain OWL reasoning capabilities in the system. In this query, property subOrganizationOf is defined as transitive. Since in the benchmark data, instances of ResearchGroup are stated as a sub-organization of a Department individual and the later suborganization of a University individual, inference about the subOrgnizationOf relationship between instances of ResearchGroup and University is required to answer this query. Additionally, its input is small.

Query12

(type Chair ?X) (type Department ?Y)
(worksFor ?X ?Y) (subOrganizationOf ?Y http://www.University0.edu)

The benchmark data do not produce any instances of class Chair. Instead, each Department individual is linked to the chair professor of that department by property headOf. Hence this query requires realization, i.e., inference that that professor is an instance of class Chair because he or she is the head of a department. Input of this query is small as well.

Query13

(type Person ?X) (hasAlumnus http://www.University0.edu ?X)

Property hasAlumnus is defined in the benchmark ontology as the inverse of property degreeFrom, which has three subproperties: undergraduateDegreeFrom, mastersDegreeFrom, and doctoralDegreeFrom. The benchmark data state a person as an alumnus of a university using one of these three subproperties instead of hasAlumnus. Therefore, this query assumes *subPropertyOf* relationships between degreeFrom and its subproperties, and also requires inference about *inverseOf*.

Query14

(@everyInstance UndergraduateStudent ?X)

This query is the simplest in the test set. This query represents those with large input and low selectivity and does not assume any hierarchy information or inference.