Choosing the Best Knowledge Base System for Large Semantic Web Applications

Yuanbo Guo
Lehigh University
19 Memorial Drive West
Bethlehem, PA18015, USA
+1-610-758-4719
yug2@cse.lehigh.edu

Zhengxiang Pan
Lehigh University
19 Memorial Drive West
Bethlehem, PA18015, USA
+1-610-758-4719
zhp2@cse.lehigh.edu

Jeff Heflin
Lehigh University
19 Memorial Drive West
Bethlehem, PA18015, USA
+1-610-758-6533
heflin@cse.lehigh.edu

ABSTRACT
We present an evaluation of four knowledge base systems with respect to use in large Semantic Web applications. We discuss the performance of each system. In particular, we show that existing systems need to place a greater emphasis on scalability.

Categories and Subject Descriptors
I.2.4 [**Artificial Intelligence**]: Knowledge Representation Formalisms and Methods – *representation languages*

H.3.4 [**Information Storage and Retrieval**]: Systems and Software – *Performance Evaluation*

General Terms
Experimentation, Measurement, Performance.

Keywords
Semantic Web, Knowledge Base System, Evaluation, Benchmark, DAML+OIL.

1. INTRODUCTION
In this work, we evaluate two memory-based systems (DAMLJessKB and memory-based Sesame) and two systems with persistent storage (database-based Sesame and DLDB) with respect to use in large Semantic Web applications. Especially we evaluate the systems in terms of how well they support the conflicting requirements of scalability/efficiency and reasoning capabilities.

We have based the experiment on the Lehigh University Benchmark [2], which supports the evaluation of DAML+OIL repositories with respect to extensional queries over large data sets that commit to a single realistic ontology. The test bed is accessible at http://www.cse.lehigh.edu/~heflin/research.

In our test, the smallest data set used consists of 15 DAML+OIL files totaling 8MB, while the largest data set consists of 999 files totaling 547MB. To our knowledge, no Semantic Web KBS has been tested with the scale of data used here.

2. THE EXPERIMENT
Sesame [1] supports RDF/RDF Schema inference, but is an incomplete reasoner for DAML+OIL. We test two implementations of it, main memory-based and database-based (Sesame-Memory and Sesame-DB hereafter). For Sesame-DB, we use MySQL as the underlying DBMS since it is reported that Sesame performs best with it. We evaluate DAMLJessKB [6], a memory-based tool for description logic languages, as a system that supports most DAML+OIL entailments. As the fourth system, DLDB [7] is a repository featuring the extension of a relational database system (MS Access®) with description logic inference capabilities (provided by FaCT reasoner [3]). We have created a wrapper over each of the above systems as an interface to the benchmark's test module.[1]

We have created 5 sets of synthetic test data containing DAML+OIL files for 1, 5, 10, 20, and 50 universities. They have respectively 99,565, 623,537, 1,271,585, 2,687,066 and 6,653,612 class instances and property instances in total.[2] We measure the elapsed time for loading each data set, and also the consequent database sizes of Sesame-DB and DLDB. For Sesame-Memory and DAMLJessKB, we evaluate their memory efficiency by looking at the largest data set they can handle.

Thirteen benchmark queries (cf. [2]) are expressed in RQL [5], Jess [4] and a KIF-like language and issued to Sesame, DAMLJessKB, and DLDB respectively. We do not use a common language in the test to eliminate the affect of query translation to the query response time. Query response time is collected and answer completeness is measured in the way as it is defined in the benchmark.

The test has been carried out on a modern desktop computer with 1.8GHz P4 CPU, 256MB of RAM, and 40GB of hard disk.

3. RESULT DISCUSSION
As it turned out, DAMLJessKB could only load the 1-university data set (99,565 instances), and took over five 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 fourth of the time of the next fastest system. However, for 20 or more universities, Sesame-Memory also succumbed to memory limitations.

Figures 1 and 2 show the load time and repository sizes respectively. The results reveal a problem for Sesame: neither of its two implementations scales in data loading: their load time is

WWW 2004, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-912-8/04/0005.

[1] We use Sesame v0.96, MySQL v4.0.16, DAMLJessKB 03-6-9 release, MS Access 2002, and DLDB 03-4-16 release.

[2] To our knowledge, prior to this experiment, Sesame has been tested with at most 3,000,000 statements.

clearly non-linear. As an example, it cost Sesame-DB over 750 times longer to load the 50-university data set than the 1-university data set. In contrast, DLDB displays good scalability in this regard.

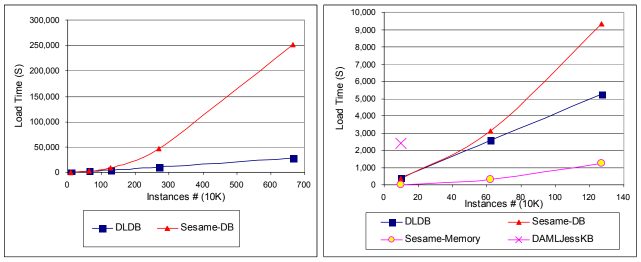

Figure 1. The left figure shows load time for DLDB and Sesame-DB over all the data sets. The right one shows load time for all the systems but only until the 10-university data set.

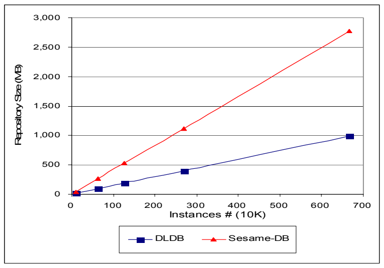

Figure 2. Repository Sizes of Sesame-DB and DLDB

In addition, the results lead to some efficiency concerns. In terms of query, although Sesame-DB scales well, it was extremely slow in answering Queries 2, 8 (cf. Figure 3), and 9. DAMLJessKB also spent a lot of time on these queries and almost two hours on Query 8 on the smallest data set. Even worse, Sesame-DB failed to answer Query 2 on the 50-univeristy data set after a JDBC connection timed out and was forced to close.

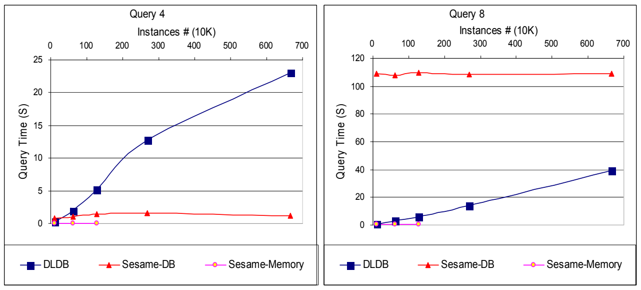

Figure 3. Query time of DLDB, Sesame-DB, and Sesame-Memory for Queries 4 and 8

Compared to the others, Sesame-Memory, although having the above mentioned scalability problem, is the fastest in data loading and in answering most of the queries. This suggests that it might be a better choice over Sesame-DB for data of small scale if persistent storage is not required. DAMLJessKB, though also memory based, underperforms the others both in load time and query time.

It is interesting to notice that Sesame and DLDB have distinct sets of queries which they are fast to answer. Particularly, we have observed that, for Queries 2, 8, and 9, which do not contain a specific URI as a subject or object in the statements, Sesame's performance goes down dramatically. On the other hand, Sesame shows a nice property in answering some other queries like Queries 1, 3, 4 (cf. Figure 3), 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. This suggests a subject for future work.

It turned out that all the four systems could answer Queries 1 through 4, which requires no extra DAML+OIL inference in order to get complete results. As we expected, DLDB was able to find all the answers for Queries 5 to 10, which requires subsumption inference, while Sesame could only find partial or no answers. However, it is surprising that DAMLJessKB could not infer the necessary subsumption for Queries 6 to 10 and hence only returned the same number of answers as Sesame. On the other hand, DAMLJessKB is the only system that was able to answer Query 11, which assumes daml:TransitiveProperty inference. Finally, none of the systems could answer Queries 12 and 13 which requires realization and daml:inverseOf inference respectively. Surprisingly DAMLJessKB did not find any answers for these two queries.

4. CONCLUSIONS

In this paper, we did not attempt to answer the question of which system is best. Instead, we revealed through the experiment some important issues and challenges for Semantic Web knowledge base systems. We found that Sesame did not scale in data loading and its database-based version was very slow in answering some queries, and DAMLJessKB could only handle a very small data set and was slow in both data loading and querying compared to the others. We believe the results show that existing systems need to place a greater emphasis on scalability. We also found that DLDB and Sesame performed well respectively on certain queries with interesting characteristics and proposed to investigate what design features contribute to these differences. Moreover, we showed how the systems varied in query answer completeness and pointed out DAMLJessKB was less complete with respect to DAML+OIL entailment than we originally expected.

5. REFERENCES

[1] Broekstra, J. and Kampman, A. Sesame: A Generic Architecture for Storing and Querying RDF and RDF Schema. In Proc. of ISWC2002.

[2] Guo, Y., Heflin, J., and Pan, Z. Benchmarking DAML+OIL Repositories. In Proc. of ISWC2003.

[3] Horrocks, I. The FaCT System. In Proc. of Tableaux'98.

[4] Jess: the Rule Engine for the Java Platform http://herzberg.ca.sandia.gov/jess

[5] Karvounarakis, G. et al. RQL: A Declarative Query Language for RDF. In Proc. of WWW2002.

[6] Kopena, J.B. and Regli, W.C. DAMLJessKB: A Tool for Reasoning with the Semantic Web. In Proc. of ISWC2003.

[7] Pan, Z. and Heflin, J. DLDB: Extending Relational Databases to Support Semantic Web Queries. In Workshop on Practical and Scalable Semantic Systems, ISWC2003.