Applying Inference Engine to Context-Aware Computing Services

Jaemoon Sim, Jihoon Kim, Ohbyung Kwon
College of Management and International Relations
KyungHee University
Yongin, Kyunggi-do, 449701, South Korea
+82-31-201-2306
{ deskmoon, hdlamb, obkwon}@khu.ac.kr

Sean S. Lee, Jungho Kim, HK Jang, Myungchul Lee
IBM Ubiquitous Computing Lab
Dogok-dong, Kangnam-gu, Seoul, Korea Rep 135700
+82-2-3781-8598
{ lsean, kjungho, hkjang, mclee}@kr.ibm.com

ABSTRACT

Ubiquitous computing services started taking advantage of the reasoning capabilities of inference engines to acquire hidden and potentially useful contextual information. However, performance evaluations of the inference engines have been limited to the domain of static information reasoning; evaluations of requirements pertaining to ubiquitous computing environment have been largely neglected. This paper aims to examine how different types of inference engines perform by applying them to realistic ubiquitous computing scenarios. Based on the scenarios, three measurement criteria are proposed and measured including scalability as data set gets large, responsiveness for user's requests, and adaptability to frequent inference requests.

Keywords

Inference Engine, OWL-DL, scalability, MINERVA, DLDB-OWL

1. INTRODUCTION

Ubiquitous computing services aim to provide information and services in more intelligent ways with more seamless interfaces that aid users to be conveniently served anytime, anywhere with any devices without awkward user intervention. These must rely on not only multiple sensors capturing user's context but also reasoning capabilities for processing raw sensed context data into more useful and meaningful information within the user's environment. Recently, inference engines such as Jena, RacerPro, FacT++, and Minerva have been proposed as a core component of intelligent ubiquitous computing systems. There had been many extensive evaluations of their reasoning capabilities in correctness, completeness, and response time [10, 11]. It is less widely known without fully analyzing how well the inference engines can fare full-fledged ubiquitous computing services covering vast zones with various sets of requirements such as lots of fast moving users and other transient computing entities. The main purpose of this paper is to examine how well inference engines satisfy responsiveness requirement, scalability requirement and requirement to accommodate frequent inference requests in response to dynamic data insertion and deletion that realistic ubiquitous computing environments exhibit. To do so, we have modeled scenarios based in a major Korean university such as MyEntrance service. MyEntrance service scenario is a part of a larger Celadon project [8] in aim to study and adopt the most suitable reasoner for ubiquitous environment in its system. Specifically, five most prominent engines are considered based on their reasoning mechanisms: three memory-based and two DBMS-based engines.

UbiPCMM06 September 18, 2006, California, USA

2. INFERENCE ENGINES

MINERVA, JENA, Pellet, RacerPro, and DLDB-OWL (HAWK) [1, 2, 3, 4, 5, 6, 7] are discussed as representative instances of each class of reasoners and they are summarized in Table 1.

2.1 MINERVA

Minerva is high performance OWL storage, inference, and query system built on RDBMS. The advantages of the system could be categorized in two important aspects of a reasoner: response time and scalability. It does all required inferences in the time of data load instead of data query, making it more responsive at the time of user query. In addition, it calculates all inferences in relational database management system, making it more scalable than memory based counterpart. It is provided as a component of IBM's Integrated Ontology Development Toolkit (IODT) [4] and it supports DLP (Description Logic Program), a subset of OWL DL and conjunctive query, a subset of the SPARQL language. Minerva uses Description Logic reasoner for TBox and a set of logic rules translated from Description Logic Programming (DLP) for ABox inference.

2.2 DLDB-OWL/HAWK

Table 1. The Comparative Table for Inference Engine

	Import	Inference			Repository		Query	Version	Source
	Ontology Language	Capability	Internal Engine	External Engine	Persistent Storage	Memo -ry	Language		
Minerva	OWL	DL	O	Racer, Pellet	DB2, Derby, HSQLDB	X	SPARQL	Semantics Toolkit-1.1.1	IBM
DLDB-OWL	OWL	DL	X	Racer	MS-Access, PostgreSQL	O	KIF-Like language	HAWK 1.3 beta	Lehigh University SWAT Lab
Pellet	DIG tell document, OWL	DL	O	X	X	O	DIG ask document, RDQL	pellet-1.3	mindswap
Racer	DIG tell document, OWL	DL	O	O	X	O	DIG ask document, Racer Query Language	RacerPro 1.9.0	Racer Systems GmbH & Co. KG
JENA	OWL	DL	O	Racer Pro, Pellet, Fact++ (JenaOntModel)	MySQL, HSQLDB, PostgreSQL, Oracle, MS-SQL Server	O	SPARQL	Jena 2.4	HP Labs

DLDB-OWL is a repository framework and toolkit that supports OWL It provides APIs as well as implementations for parsing, editing, manipulating and preservation of OWL ontologies. The architecture of DLDB-OWL consists of three packages: core, owl and storage. The core package defines the generic interfaces of the data structures of ontology and ontology objects, e.g. class and property. The core package, that is independent of underlying model the application will use, provides an API for constructing and manipulating ontology models. The owl package provides the utilities for parsing and serializing ontologies in OWL language.

2.3 Pellet

Pellet is an open-source Java based OWL DL reasoner. It can be used in conjunction with both Jena and OWL API libraries and also provides a DIG interface. Pellet API provides functionalities to see the species validation, check consistency of ontologies, classify the taxonomy, check entailments and answer a subset of RDQL queries. It supports the full expressivity OWL DL including reasoning about nominals (enumerated classes).

2.4 RacerPro

RacerPro is an OWL reasoner and inference server for the semantic web. The origins of RacerPro are within the area of description logics. It can be used as a system for managing semantic web ontologies based on OWL. However, RacerPro can also be seen as a semantic web information repository with optimized retrieval engine because it can handle large sets of data descriptions.

2.5 JENA

Jena is a Java framework for building Semantic Web applications. It provides a programmatic environment for RDF, RDFS and OWL, SPARQL and includes a rule-based inference engine. The Jena Framework includes a RDF API, reading and writing RDF in RDF/XML, N3 and N-Triples and an OWL API and SPARQL query engine.

3. PERFORMANCE EVALUATION

3.1 MyEntrance Service Scenario

To analyze how the legacy inference engines perform in realistic ubiquitous computing environments, we used a MyEntrance service scenario modeled based on Kyunghee Univerity (KHU). KHU has two great campuses, Seoul campus with 50 departments and YongIn campus with 51 departments.

“When a member of KHU enters the Student Union Building, a service agent recognizes the member’s preference and searches for an Event on the Application Server to provide it for him/her. Additionally, a sports equipment shop located in Student Union Building wants to advertise its sales promotion on new baseball products for the KHU students and faculty members. The Service Agent of the sports equipment shop in KHU requests Application Server to retrieve the information on the hobbies of members located in the Student Union Building. The Application Server hands over the request of Service Agent to Context Server about the hobby information of members who are located in the Student Union Building. Context Server returns the result of preference and product matching inference. Consequently, the Service Agent checks for the members who like to play baseball and it sends the Discount Event Information to appropriate members. The members who receive the event information make a purchase decision with the received offer.

To accommodate this scenario, we have developed KHU campus ontology based on the ontology generation program supported by IBM China Lab [9]. On top of the generated ontology, we have installed area-based location context. An experimental data set which represents an actual college (9 departments, file size= 11.8MB), International College in KHU at Suwon, was made and used in the performance evaluation.

3.2 Results: Performance Evaluation on Static and Dynamic Context Information

For the performance evaluation on static information, we placed our focus on scalability and subsequent performance issue. Specifically, in evaluating the performance of query processing, we considered 16 sets of University Ontology Benchmark [9] queries that were generated by IBM China Research Lab. by extending widely used Lehigh University Benchmark [10].
In order to evaluate handling of context information, SPARQL is used as follows:

```
SELECT
DISTINCT ?person ?zone ?Hobby1 ?Hobby2 ?Hobby3
WHERE
(?person benchmark:locatedIn
<http://rcubs.kyunghee.ac.kr/owl/rcubs-univ-bench-
dl.owl#Zone1>)
(?person benchmark:locatedIn ?zone)
(?person benchmark:like ?Hobby1)
```

For DLDB-OWL/HAWK evaluation, the above query is translated into KIF-like query as follows:

```
[http://rcubs.kyunghee.ac.kr/owl/univ-bench-
dl.owl]
(type Person ?x)
(locatedIn ?x
http://rcubs.kyunghee.ac.kr/owl/rcubs-univ-bench-
dl.owl#Zone1)
(locatedIn ?x ?z)
(like ?x ?y1)
```

We set up our scenario environment of campus like the topology shown as Figure 1. For context generation, user's current location is gathered by the context event handler on the client's portable device or tags at any time the user is passing by sensors located in the entrance gate of the campus buildings. At a fixed cycle, the context event handler sends the user's current context information as OWL-DL format to the context server. The context server returns the result if the ontology files comes through successfully to the context event handler. The user may invoke the application server to get location-based service at any time the user wants. Then the application server asks the context server to get service-related context data as facts. According to the facts, the inference engine determines the location-based services most appropriate for the user.

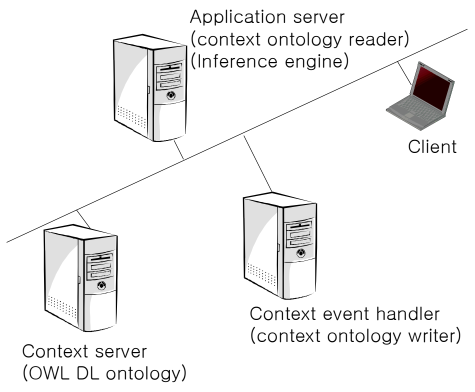

Figure 1. Simulation Environment

We select query response time as the performance measure. The summary of response time is listed in Table 2.

Table 2. Summary of query response time

Query #	Response Time (ms)		
	MINERVA	DLDB-OWL /HAWK	Pellet
1	424.90	10.00	5858.00
2	312.80	12.85	268.70
3	284.30	241.85	115.50
4	382.80	221.57	89.00
5	358.00	2.85	31.20
6	343.70	5.71	42.20
7	483.00	5.71	40.50
8	743.60	5.71	1813723.00
9	843.90	160.14	295.40
10	857.90	25.71	7.80
11	-	81.71	17.40
12	1176.60	43.00	9.20
13	906.40	2.85	1199564.80
14	1076.60	173.00	467724.80
15	978.10	158.71	208571.80
16	1186.00	291.71	51.50

Performance evaluations in case of dynamic context were also performed. Dynamic context evaluations were differentiated from the static context evaluation in that, engines were constantly requested to inference based on new data whereas the static evaluations were performed after the inference processing is finished. For this test, since the memory-based reasoning systems cannot be directly compared to the DBMS-based reasoning systems in their absolute loading time, we tested two database-based reasoning systems: MINERVA and DLDB-OWL/HAWK. Only the test with data set 1 is described in this paper because we encountered a consistent problem with the dynamic data set and did not see the value in presenting more data in this study. The result is listed in

Table 3. Summary of load time in dynamic situation

Cycle time of updating context data	Cycle time of query processing	Load Time (%)	
		Minerva	DLDB-OWL
1200 SEC	600sec	33.3 0.0 66.7	99.8 0.2 0.0
	60sec	5.0 6.7 88.3	89.5 10.5 0.0
	10sec	0.6 3.6 95.8	91.3 8.7 0.0
900 SEC	600sec	33.3 16.7 50.0	93.3 6.7 0.0
	60sec	5.0 6.7 88.3	84.0 16.0 0.0
	10sec	0.8 6.7 92.5	89.1 10.9 0.0
600 SEC	600sec	50.0 0.0 50.0	90.0 10.0 0.0
	60sec	6.7 5.0 88.3	71.0 29.0 0.0
	10sec	0.8 6.7 92.5	85.7 14.3 0.0

Table 3 where the first number of the three denotes the rate of yielding correct answers to the query aforementioned. The second and third numbers denote the rate of yielding wrong answers and no response to the query, respectively.

4. CONCLUSION

We first studied the responsiveness to the user request using both memory-based and DBMS-based inference engines in static context setting. The result indicated that database-based inference engines far outperform the other in this criterion owing to the fact that memory based inference engines pre-process context data while loading and thus is able to respond without further calculation at query time.

Then, using DBMS-based inference engines, we analyzed how they perform in conducting a context-aware MyEntrance service under the setting and environment of a major university. By varying the cycle of context changes and knowledge loading, performance measures such as correctness and completeness were examined. We noticed that engines do not respond to queries in the middle of their inferences and sometimes generate invalid or no responses.

We conclude that current state-of-the-art inference engines do not fulfill responsiveness, scalability and high-update frequency requirements demanded for ubiquitous computing environments with lots of fast moving users and other transient computing entities. And, evolutionary algorithms or inference mechanisms to deal with enormous amount of data, frequency of updates and respond to user's needs in time are needed for the inference engines to be improved.

5. ACKNOWLEDGEMENT

Thanks to the Institute of Information Technology Assessment and Ministry of Information and Communication of Republic of Korea for providing an opportunity to participate in the IT839 project. And, thanks to IBM China Research Laboratory for providing detailed information about Integrated Ontology Development Toolkit available on IBM AlphaWorks.

REFERENCES

[1] Carroll, J.J., Dickinson, I., Dollin, C., Reynolds, D., Seaborne, A., Wilkinson, K. Jena: Implementing the Semantic Web Recommendations. Proceedings of the 13th International World Wide Web Conference. ACM Press, New York (2004) 74-83

[2] Guo, Y., Pan, Z., Heflin., J. An Evaluation of Knowledge Base Systems for Large OWL Datasets. Proceedings of the 3rd International Semantic Web Conference, Hiroshima. LNCS, Vol. 3298. (2004) 274-288

[3] Haarslev, V., M¨oller, R., Wessel, M. Querying the Semantic Web with Racer + nRQL, In: Proc. of the KI-04 Workshop on Applications of Description Logics. (2004)

[4] IBM's IODT/Minerva team: Minerva Reasoner, See, http://www.alphaworks.ibm.com/tech/semanticstk or http://www.ifcomputer.com/MINERVA/

[5] Kevin, W., Sayers, C., Kuno, H. Efficient RDF Storage and Retrieval in Jena2. Proceedings of First International Workshop on Semantic Web and Databases (2003) 131-151

[6] Sirin, E., Parsia, B. Pellet: An owl dl reasoner. Proceedings Of the Int. Third International Semantic Web Conference (ISWC 2004) - Poster (2004)

[7] Wessel, M., Möller, R. A High Performance Semantic Web Query Answering Engine. International Workshop on Description Logics (DL2005), Edinburgh, Scotland, UK. (2005)

[8] MC Lee, HK Jang, YS Paik, SE Jin, S Lee A Ubiquitous Device Collaboration Infrasturcture: Celadon. Third Workshop on Software Technologies for Future Embedded & Ubiquitous Systems (SEUS 2006)

[9] Li Ma, Yang Yang, Zhaoming Qiu, Guotong Xie, Yue Pan, Shengping Liu: Towards A Complete OWL Ontology Benchmark. 3rd European Semantic Web Conference (ESWC06) – 2006

[10] Y. Guo, Z. Pan, and J. Heflin. LUBM: A Benchmark for OWL Knowledge Base Systems. Journal of Web Semantics 3(2), 2005, pp158-182.

[11] T. Liebig, H. Pfeifer, F. von Henke, Reasoning Services for an OWL Authoring Tool: An Experience Report, in: Proceedings of the 2004 International