SmartSlog Knowledge Patterns: Initial Experimental Performance Evaluation

Pavel Vanag, Dmitry Korzun
Department of Computer Science,
Petrozavodsk State University
Petrozavodsk, Russia
{vanag, dkorzun}@cs.karelia.ru

Abstract

The Smart-M3 platform allows prototyping and experimenting with various smart space applications. The performance problem of such applications and the level of automation in the development process is, however, in its early phase. In this paper, we evaluate the knowledge pattern mechanism that the SmartSlog SDK provides for Smart-M3 applications. On one hand, the mechanism allows writing compact code to implement advanced search queries to the smart space. On the other hand, high-level mechanisms are for the cost of performance; it can essentially degrade when the size of search problem grows. Therefore, a tradeoff has to be determined. Our initial experimental study considers a worst case model of the knowledge pattern mechanism and we provide empirical performance upper bounds.

Index Terms: Smart-M3, SmartSlog, Knowledge patterns, Performance evaluation.

I. Problem statement

Smart-M3 provides a multi-agent distributed application with a common smart space to share dynamic knowledge and to make reasoning cooperatively [1]. A Smart-M3 semantic information broker (SIB) maintains its smart space and represents the content in such low-level terms as RDF triples. A Smart-M3 application consists of knowledge processors (KPs) that share the application smart space. Since the same SIB can be used by several application, the SIB supports several application smart spaces within its own space.

For development efficiency, KP code can be constructed on top of its ontology library, which provides data structures and API to access the required part of the application smart space. The code is written using such high-level OWL ontology terms as classes, properties, and individuals [2], a convenient way when the problem domain allows ontological modeling. SmartSlog is an ontology multilingual library generator for Smart-M3 applications [3]. Among other useful features, it is oriented to tunable optimization of KP code size, system dependencies, device CPU/memory consumption, network load, and data synchronization.

In our prior work [3] we introduced a mechanism for effective manipulation with smart space content on the KP side: filtering locally available objects or searching-and-retrieving objects from the smart space. The mechanism applies our knowledge pattern model. A knowledge pattern is a graph (K-graph) where nodes represent virtual objects (individuals) from the ontology. Nodes are augmented with datatype properties and linked by object properties. It is similar to a downsized variant of the OWL ontology instance graph; the latter represents the whole knowledge in the space. In a K-graph, nodes are not actual individuals; instead, they are interpreted as masks or variables for actual individuals. The developer specifies with a knowledge pattern only a part of properties for objects in the ontology. In filtering, these selected properties are compared with properties of locally stored individuals. In searching, these properties are used to retrieve specified individuals mapping the K-graph to the ontology instance graph in the smart space.

Recently, knowledge pattern-based searching in the smart space is one of the computationally expensive operations that a SmartSlog ontology library implements. In this short paper, we focus on performance evaluation of knowledge patterns. Unfortunately, no essential knowledge processing can be delegated to SIB, since Smart-M3 v. 0.9.5 (or lower) has support of WirbulQL (or WQL) with its simple queries only. Nevertheless, we expect that in further releases SIB will include SPAQRL support [4], and then SmartSlog can delegate more processing from the KP side to the SIB side.

II. K-GRAPH WORST-CASE MODEL

The knowledge pattern-based search is an iterative process. Each iteration makes data queries to SIB for the next part of knowledge from the smart space. Also, each iteration involves bidirectional online transformation between an RDF triple set (knowledge representation model at the SIB side) and an OWL instance graph (knowledge representation model at the KP side with SmartSlog ontology library). Figure 1 depicts the basic steps and the points of communication with SIB.

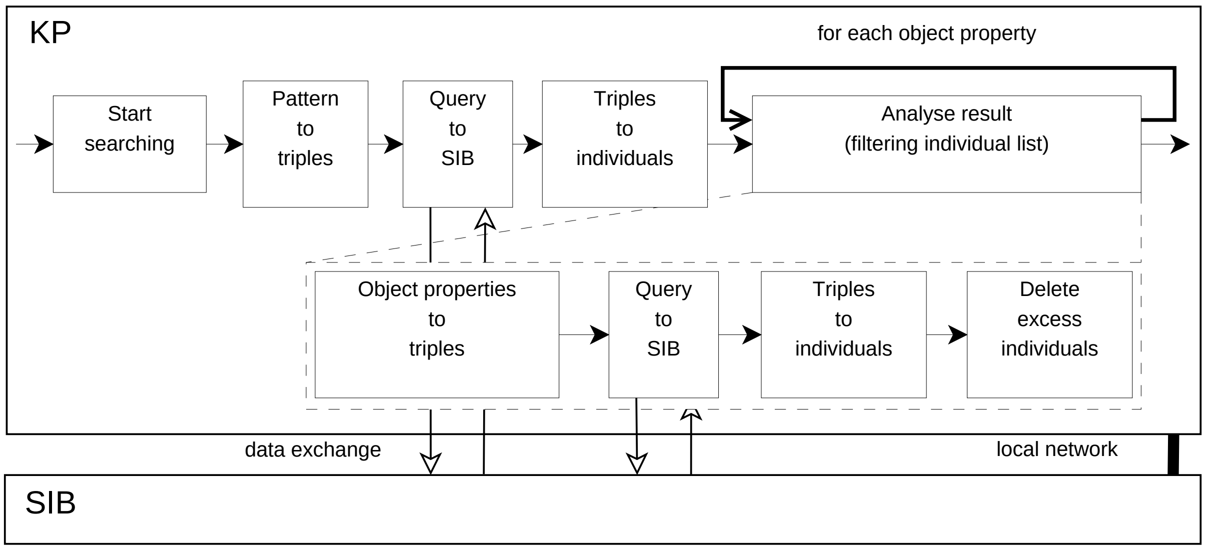

Fig. 1. Iterative steps of knowledge pattern-based search at the KP side.

Consequently, the performance essentially depends on the size of K-graph. Let us consider the following size parameters.

1) s_{wg} is the number of datatype properties that every object has (graph weight).
2) s_{wd} is the number of object properties that every object has (graph width).
3) s_{hg} is the longest path from a fixed node to other nodes (graph height).

They characterize the class of K-graphs for our performance evaluation. The names “weight”, “width”, and “height” support the intuition that a typical case for a K-graph is a tree, see Fig. 2. The definition of s_{wg} and s_{wd} restricts the class with regular graphs: the same number of edges per node. If a K-graph is a tree then the root is used as the fixed node in the definition of s_{hg}.

Although a K-graph can be different from a regular tree, our K-graph model is an approximation for the worst-case scenario. Given a K-graph, to estimate a performance upper bound,

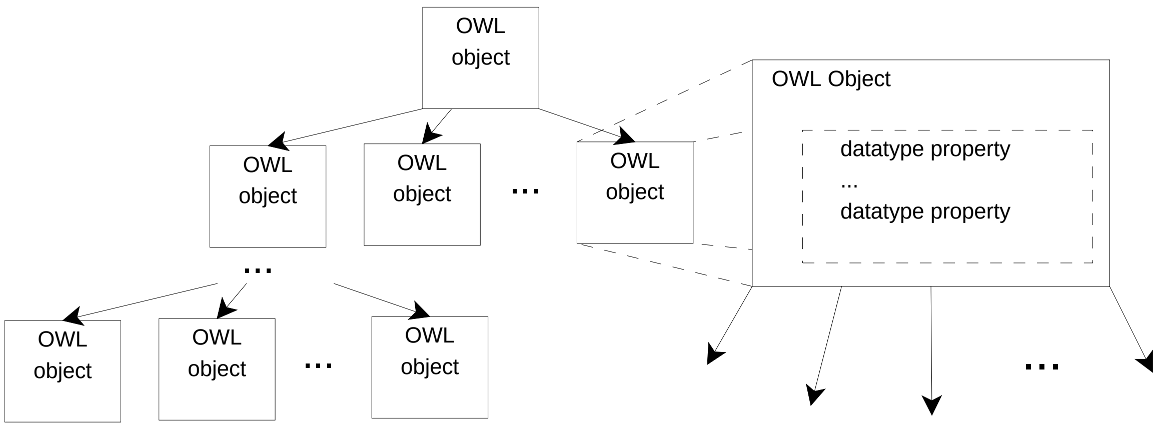

Fig. 2. A knowledge pattern forming a tree-like K-graph. In our experiments, we artificially generated only K-graphs from this class of regular trees.

one can take the maximum over all the objects for the number of data and object properties per node. If a K-graph is a set of trees then the performance upper bound estimation can be straightforwardly reduced to a single tree case.

We expect that the case of K-graphs with cycles is uncommon in simple smart applications. Such K-graphs appear in advanced scenarios with non-trivial knowledge deduction. We leave this topic to our further research.

III. Performance evaluation

For experimental evaluation, we implemented a special KP to generate knowledge patterns for different size parameters and to make search queries to the smart space. The test content (entire ontology instant graph) is published in the smart space in advance. In the worst case, K-graph and the ontology instant graph coincide since the full search is performed.

Let N_{ind} be the number of individuals in a generated graph. They are stored in the smart space (the SIB side) using N triples. It requires N_{rdf} RDF triples to keep facts about each individual and N_{ont} RDF-scheme triples to keep the high-level ontology declarations, i.e.,

$$N = N_{\mathrm{ont}} + N_{\mathrm{ind}} N_{\mathrm{rdf}},$$

where N_{ont} is a constant for a fixed ontology. Every individual is represented using $N_{\mathrm{rdf}} = 1+s_{\mathrm{wg}}+s_{\mathrm{wd}}$ triples. Due to the well-known property on the number of nodes in a tree of height $s_{\mathrm{hg}} \geq 1$ and degree $s_{\mathrm{wd}} \geq 2$, the total number of individuals is $N_{\mathrm{ind}} = (s_{\mathrm{wd}}^{s_{\mathrm{hg}}} - 1)/(s_{\mathrm{wd}} - 1)$. Consequently, the smart space content size is characterized by the equation

$$N = N_{\mathrm{ont}} + \frac{s_{\mathrm{wd}}^{s_{\mathrm{hg}}} - 1}{s_{\mathrm{wd}} - 1}(1 + s_{\mathrm{wg}} + s_{\mathrm{wd}}).$$

We measure the time $T = T(s_{\mathrm{wg}}, s_{\mathrm{hg}}, s_{\mathrm{wd}})$ that a search query spends (in microseconds) on average. The empirical dependencies are shown in Fig. 3. We vary $s_{\mathrm{wg}}, s_{\mathrm{wd}}$ from 1 to 10 and s_{hg} from 1 to 5 because higher values for s_{hg} cause rapid growth of experiment time.

The experiment results show that the search performance can be described by a polynomial model with high constant exponents or even by a exponential model. We assume the worse case and consider the exponential dependence

$$T(s_{\mathrm{wg}}, s_{\mathrm{hg}}, s_{\mathrm{wd}}) = b_0 \exp\left(b_1 s_{\mathrm{wg}} + b_2 s_{\mathrm{hg}} + b_3 s_{\mathrm{wd}}\right).$$

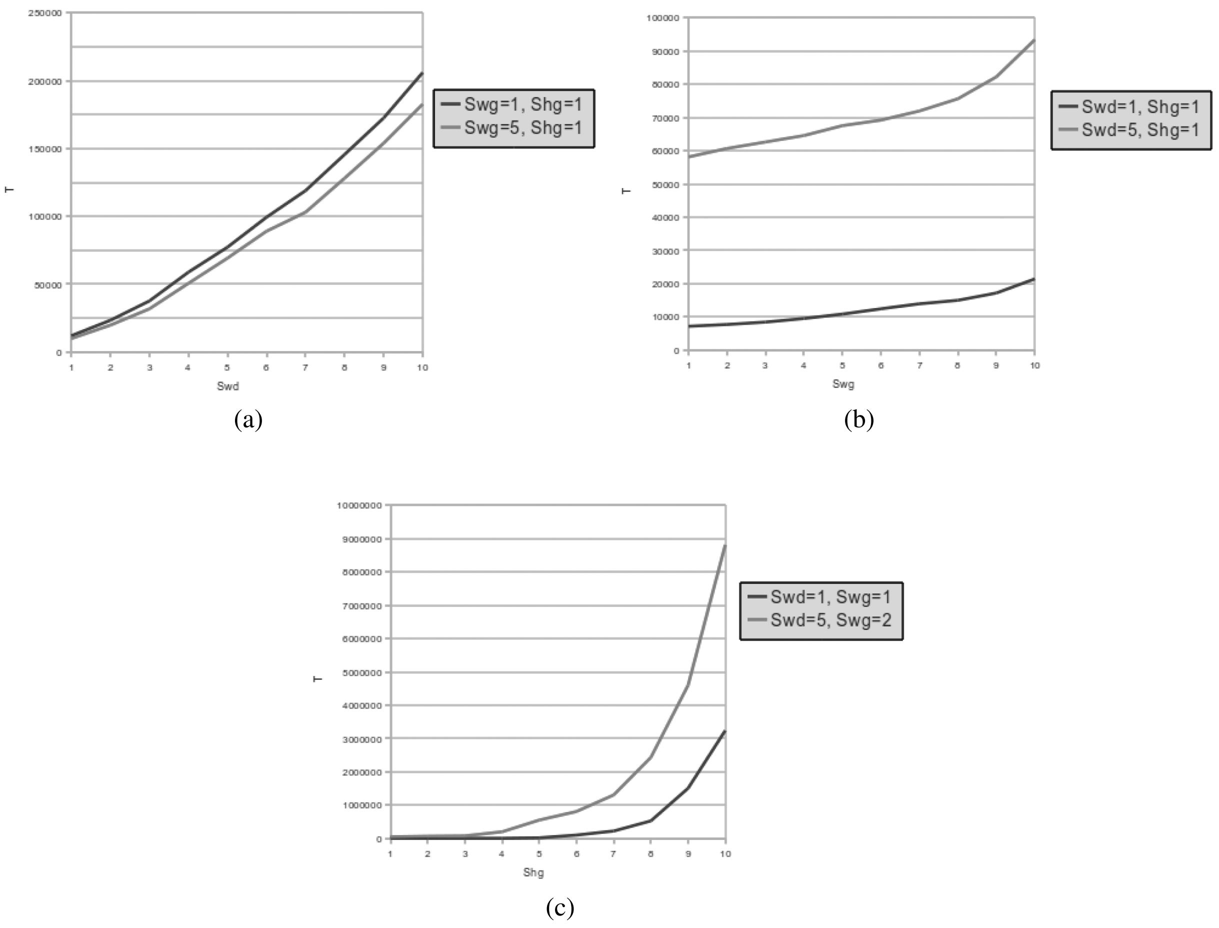

Fig. 3. The measured dependency $T(s_{\rm wg}, s_{\rm wd}, s_{\rm hg})$. (a) Variation of $s_{\rm wd}$ with fixed $s_{\rm wh}$ and $s_{\rm hg}$. (b) Variation of $s_{\rm wg}$ with fixed $s_{\rm wd}$ and $s_{\rm hg}$. (c) Variation of $s_{\rm hg}$ with fixed $s_{\rm wg}$ and $s_{\rm wd}$.

Applying multiple non-linear regression analysis [5] we determined the explanatory variables: $b_0 \approx 11.582$, $b_1 \approx 0.034$, $b_2 \approx 5.538$, $b_3 \approx 0.388$. More than 83% of the variance is accounted by the statistical model.

Parameter b_0 is a scaling coefficient. It depends on the power of machines where KP and SIB are located as well as the network bandwidth between them. More importantly, the model shows that the K-graph height, width and weight are arranged with separation in magnitude of the one-degree order, i.e., the following performance-impact proportion can be used for coarse-grain estimates:

$$s_{\rm hg} : s_{\rm wd} : s_{\rm wg} \approx 1 : 10 : 10^2.$$

The most significant factor is the height $s_{\rm hg}$, see also Fig. 3 (c). As for a practical conclusion, it is better to use more datatype properties than object properties a knowledge pattern. The developer should carefully make decisions on use of those object properties that leads to lengthy paths.

IV. Related work

The performance problem for Smart-M3 application and the level of automation in the development process is in its early phase [2]. Works [6] and [7] considered the Smart-M3 performance, focusing primarily on the SIB side. In contrast, we studied the performance available at the KP side and its relation to automated KP development.

D'Elia *et al.* [6] proposed an algorithm for SIB to provide basic access control functionality at RDF triple pattern level. The insert operation performance is evaluated as a function of active protections in SIB. The key finding is that the impact on SIB response time is small. The algorithm was included to Smart-M3 from version 0.9.5.

Kantola [7] studied the problem of synchronizing data between a RESTfulWeb Service and a smart space (SIB, Smart-M3 v. 0.9.2). The study indicated that the SIB performance could become a problem even in case of a few simultaneous data updating agents.

V. Conclusion

Our early measurements showed the basic trends in performance upper bound when non-trivial search is performed in the smart space and the search is controlled at the KP side. They indicate that this type of search is possible from the performance point of view. Our evaluation also provides coarse estimates for a developer to decide the size limit of knowledge patterns for her/his KP code. We plan to continue this research applying other benchmarks and models, with further focus on typical scenarios of real-life Smart-M3 applications.

Acknowledgment

This research is a part of grant KA179 “Complex development of regional cooperation in the field of open ICT innovations” of Karelia ENPI programme, which is co-funded by the European Union, the Russian Federation and the Republic of Finland. We are grateful to the Open Innovations Association FRUCT for its support and R&D infrastructure. We would also like to thank Sergey I. Balandin and Iurii A. Bogoiavlenskii for their feedback and expertise.

References

[1] J. Honkola, H. Laine, R. Brown, and O. Tyrkkö, “Smart-M3 information sharing platform,” in *Proc. IEEE Symp. Computers and Communications*, ser. ISCC ’10. IEEE Computer Society, Jun. 2010, pp. 1041–1046.

[2] D. G. Korzun, S. I. Balandin, V. Luukkala, P. Liuha, and A. V. Gurtov, “Overview of Smart-M3 principles for application development,” in *Proc. Congress on Information Systems and Technologies (IS&IT’11), Conf. Artificial Intelligence and Systems (AIS’11)*, vol. 4. Moscow: Physmathlit, Sep. 2011, pp. 64–71.

[3] D. G. Korzun, A. A. Lomov, P. I. Vanag, J. Honkola, and S. I. Balandin, “Multilingual ontology library generator for Smart-M3 information sharing platform,” *International Journal on Advances in Intelligent Systems*, vol. 4, no. 3&4, 2011.

[4] E. Prud’hommeaux and A. Seaborne, “SPARQL query language for RDF,” W3C Recommendation, Jan. 2008. [Online]. Available: http://www.w3.org/TR/rdf-sparql-query/

[5] G. A. F. Seber and C. J. Wild, *Nonlinear Regression*, ser. Wiley Series in Probability and Statistics. John Wiley & Sons, 1989.

[6] A. D’Elia, D. Manzaroli, J. Honkola, and T. S. Cinotti, “Access control at triple level: Specification and enforcement of a simple RDF model to support concurrent applications in smart environments,” in *Proc. 11th Int’l Conf. Next Generation Wired/Wireless Networking (NEW2AN’11) and 4th Conf. Smart Spaces (ruSMART’11)*. Springer-Verlag, 2011.

[7] E. Kantola, “Synchronizing data between a social networking service and an RDF store via publish/subscribe,” Faculty of Information and Natural Sciences, Aalto University, Helsinki, Finland, Master’s Thesis, Jun. 2010.