Mogwaï: a Tool to Query Very Large Models on NoSQL Backends

Gwendal Daniel[1], Gerson Sunyé[1], and Jordi Cabot[3]

[1] AtlanMod Team
Inria, Mines Nantes & Lina
{gwendal.daniel,gerson.sunye}@inria.fr
[2] ICREA, UOC
jordi.cabot@icrea.cat

Abstract. Scalability of existing modeling frameworks has become a major issue of MDE adoption in the industry. Specifically, scalable model persistence and efficient query frameworks are two of the challenges that have to be addressed to deal with very large models. This paper introduces Mogwaï, an efficient query framework designed to compute and execute OCL (Object Constraint Language) queries over models stored in NeoEMF, a persistence layer that stores very large models into NoSQL databases. Mogwaï relies on a model-to-model transformation that maps OCL queries to Gremlin, a generic NoSQL query language supported by several databases. Queries are computed on the database side, benefiting of all the query optimizations and embedded indexes / caches, improving execution time and reducing memory footprint. This tool is fully available online.

Keywords: Model Driven Engineering, Scalability, OCL, Model Query

1 Introduction

With the progressive adoption of modeling techniques in the industry [8], existing tools have to deal with large models [2], and the scalability of available solutions to persist, edit, transform, and query models has become a major issue [7].

In the last decade, the Eclipse Modeling Framework (EMF)[3] has become the *de-facto* standard framework for building modeling tools, offering a strong foundation to implement model navigation, queries, and storage solutions. Nevertheless, the EMF was first designed to handle simple modeling activities, and its default serialization mechanism–XMI[4]–has shown clear limitations to handle very large models [1, 9].

These limitations have led to the creation of several persistence frameworks built on top of the EMF to achieve model scalability using databases [5, 9] and advanced mechanisms such as application-level caches and *lazy-loading*. NeoEMF [1] is one of these solutions. It provides a multi-store implementation allowing to represent models as instances of key-value stores, graph databases, and distributed column database.

[3] https://eclipse.org/modeling/emf/
[4] http://www.omg.org/spec/XMI/2.4.1/

While this evolution of model persistence backends has improved the support for managing large models, they are just a partial solution to the scalability problem in current modeling frameworks. In its core, all frameworks are based on the use of low-level model handling APIs which are then used by most other MDE tools in the framework ecosystem. This approach is clearly inefficient when used on top of persistence frameworks because (i) the API granularity is too fine-grained to benefit from the advanced query capabilities of the backend and (ii) an important time and memory overhead is necessary to construct navigable intermediate objects that are needed by the API.

To overcome this situation, we propose Mogwaï [3], an efficient and scalable query framework for large models. Mogwaï consists of a translation component that maps model queries written in OCL into expressions of a graph traversal language, Gremlin [10]. Generated queries are then sent to the NeoEMF/Mogwaï component, which is in charge of the computation of these queries on the database side, bypassing the EMF API.

The paper is organized as follows: Section 2 gives a short overview of the Mogwaï query infrastructure. Section 3 presents the tool implementation, Section 4 introduces the highlights of the demonstration, and finally Section 5 summarizes the key points of the paper.

2 The Mogwaï Framework

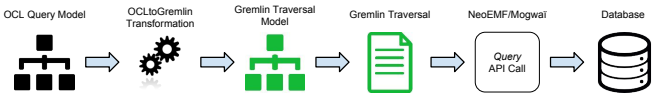
The Mogwaï framework is composed of two components: (i) the `OCL2Gremlin` model-to-model transformation, which maps OCL expressions on to Gremlin traversals, and (ii) the `NeoEMF/Mogwaï` persistence layer, an extension of NeoEMF that provides an advanced query API for graph databases. We choose OCL as our input language because it is a well-known OMG standard used to complement graphical (meta) modeling languages with textual descriptions of invariants, operation contracts, derivation rules, and query expressions. Gremlin is a NoSQL query language designed to query databases implementing the Blueprints API, an abstraction layer on top of graph stores which has been implemented by several databases. Therefore, we choose Gremlin as our target language, because it is the most mature and generic solution to query a wider variety of NoSQL databases.

Figure 1 shows the overall query process of (a) the Mogwaï query framework and compares it with (b) standard EMF API based approaches. An initial textual OCL expression is parsed and transformed into an OCL query model. This model constitutes the input of the `OCL2Gremlin` Mogwaï component, which consists of a model-to-model transformation generating the corresponding Gremlin traversal model.

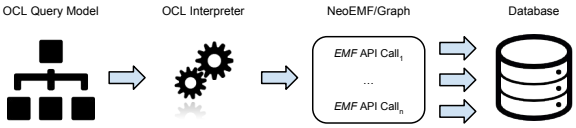
This transformation is composed of a mapping from OCL on to Gremlin and a translation algorithm implementing that mapping. The Gremlin model is then converted to a textual expression and sent to the `NeoEMF/Mogwaï` component, that computes it on the database side. Query results are then reified as standard EMF objects by `NeoEMF/Mogwaï`, making them usable in any EMF-based scenario.

Compared to existing query frameworks, Mogwaï does not rely on the EMF API to perform a query. In general, API based query frameworks translate OCL queries into a sequence of low-level API calls, which are then performed one after another

on the persistence layer (in this example `NeoEMF/Graph`). While this approach has the benefit to be compatible with every EMF-based application, it does not take full advantage of the database structure and query optimizations. Furthermore, each object fetched from the database has to be reified to be navigable, even if it is not going to be part of the end result. Therefore, execution time of the EMF-based solutions strongly depends on the number of intermediate objects fetched from the database.



(a) Mogwaï Query Framework



(b) EMF-based Query Frameworks

Fig. 1. Comparison of OCL execution

3 Implementation

Mogwaï consists of a set of open-source Eclipse plugins released under the EPL license[5]. Source code and benchmark materials are fully available on Github[6]. An Eclipse update site containing the last stable version of the framework is also available online.

Initial OCL queries are parsed using Eclipse MDT OCL [4] and the output OCL models constitute the input of a set of 70 ATL [6] transformation rules and helpers implementing the mapping and the transformation process presented in previous work [3].

The resulting script is sent to NeoEMF/Mogwai, which contains a Gremlin execution engine that computes the traversal through the Blueprints API on top of Neo4j. Results are then reified to create a navigable EMF model. Resulting model elements are created only from the query results, removing the memory overhead implied by intermediate objects creation during the query execution on EMF-based approaches.

4 Demonstration Overview

The demonstration will present a typical use case where the Mogwaï framework improves significantly the execution time and memory consumption of OCL queries. First, we will introduce model persistence in the NeoEMF architecture and how it is possible to use NoSQL backends to store very large models efficiently. In particular, we will emphasize the graph component, which stores models using property graph primitives (such as *vertices*, *edges*, *properties*). In this initial step it will be demonstrated

[5] `https://www.eclipse.org/legal/epl-v10.html`
[6] `https://github.com/atlanmod/Mogwai`

that NeoEMF/Mogwaï is able to store efficiently and with a small memory footprint a model containing two million elements.

After persisting the model as a graph, we will present the query features of the Mogwaï framework, this step includes the design of typical OCL queries (extracted from real-world software modernization use cases), an analysis of the produced Gremlin traversals, and a comparison with standard EMF query frameworks in terms of design, execution time, and memory consumption.

Finally, the key points of the tool will be summarized and some remarks about integration into existing modeling applications will be provided. Materials used during the presentation will be publicly available on the Mogwaï Github repository.

5 Conclusion

In this paper we introduced Mogwaï, a framework that generates Gremlin traversals from OCL queries in order to maximize the benefits of using a NoSQL backend to store large models. Gremlin traversals are created using a model-to-model transformation and are then delegated to a specific persistence layer which offers query facilities, bypassing EMF API's limitations. We have presented an open source implementation of the Mogwaï integrated in NeoEMF, and previous work [3] has shown a clear benefit of using it in terms of memory consumption and execution time.

References

1. Amine Benelallam, Abel Gómez, Gerson Sunyé, Massimo Tisi, and David Launay. Neo4EMF, a Scalable Persistence Layer for EMF Models. In *Proc. of the 10th ECMFA*, pages 230–241, York, United Kingdom, 2014.
2. Gábor Bergmann, Ákos Horváth, István Ráth, Dániel Varró, András Balogh, Zoltán Balogh, and András Ökrös. Incremental evaluation of model queries over EMF models. In *Proc. of the 13th MoDELS conference*, pages 76–90. Springer, 2010.
3. Gwendal Daniel, Gerson Sunyé, and Jordi Cabot. Mogwaï: a framework to handle complex queries on large models. In *Proc of the 10th RCIS Conference (to appear)*. IEEE, 2016. Available Online at http://tinyurl.com/jgopmvk.
4. Eclipse Foundation. MDT OCL, 2016. URL: www.eclipse.org/modeling/mdt/?project=ocl.
5. Eclipse Foundation. The CDO Model Repository (CDO), 2016. URL: http://www.eclipse.org/cdo/.
6. Frédéric Jouault, Freddy Allilaire, Jean Bézivin, and Ivan Kurtev. ATL: A model transformation tool. *SCP*, 72(1–2):31 – 39, 2008.
7. Dimitrios S Kolovos, Louis M Rose, Nicholas Matragkas, Richard F Paige, Esther Guerra, Jesús Sánchez Cuadrado, Juan De Lara, István Ráth, Dániel Varró, Massimo Tisi, et al. A research roadmap towards achieving scalability in model driven engineering. In *Proc. of 1st BigMDE Workshop*, pages 2–12. ACM, 2013.
8. Parastoo Mohagheghi, Miguel A Fernandez, Juan A Martell, Mathias Fritzsche, and Wasif Gilani. MDE adoption in industry: challenges and success criteria. In *Models in Software Engineering*, pages 54–59. Springer, 2009.
9. Javier Espinazo Pagán and Jesús García Molina. Querying large models efficiently. *IST*, 56(6):586–622, 2014.
10. Tinkerpop. The Gremlin Language, 2016. URL: www.gremlin.tinkerpop.com.