Scalable Incremental Query Evaluation for Large Models*

Gábor Szárnyas[1], István Ráth[1] and Dániel Varró[1,2,3]

[1] Budapest University of Technology and Economics,
Department of Measurement and Information Systems,
H-1117 Magyar tudósok krt. 2, Budapest, Hungary
{szarnyas, rath, varro}@mit.bme.hu
[2] DIRO, Université de Montréal
[3] MSDL, Dept. of Computer Science, McGill University

Abstract. Model query operations form the basis of model-driven software engineering tools and transformations. While the last decade brought considerable improvements in scalable, distributed storage technologies, known as NoSQL systems, evaluation of graph-like queries on large models requires further research. Unlike typical NoSQL workloads, e.g. key-value lookups or queries utilising full-text search, model queries often include lots of join, antijoin and complex filtering operations. The goal of our research is to propose an approach for scalable incremental query evaluation in a distributed system. This includes designing an architecture which allows querying on databases using different data representation formats. To achieve high performance, such a system should be capable of automatically allocating its resources (e.g. data shards and query processing nodes) in the cloud. To support this, we plan to define query and model metrics for estimating the resources required for the evaluation of each query.

1 Introduction

Nowadays, model-driven software engineering (MDE) plays an important role in the development processes of critical embedded systems. Advanced modelling tools provide support for a wide range of development tasks such as requirements and traceability management, system modelling, early design validation, automated code generation, model-based testing and other validation and verification tasks. With the dramatic increase in complexity that is also affecting critical embedded systems in recent years, modelling toolchains are facing scalability challenges as the size of design models constantly increases, and automated tool features become more sophisticated. Models from other MDE applications are also getting larger [3]. Reverse engineering software code to models, collecting sensor data and building geo-spatial models of cities can yield to models with model elements in the magnitude of 10^9 or more [15]. Modern web applications,

* This work was partially supported by the CERTIMOT (ERC_HU-09-01-2010-0003) and MONDO (EU ICT-611125) projects.

such as social networks and analytical services are also producing large graphs with diverse structural characteristics.

Most of these applications, especially software modelling and geo-spatial models include complex queries. These queries differ from the ones used in typical transactional databases as they often feature lots of join and antijoin operations along with recursive patterns and transitive closures [18]. Previous results have shown that current state-of-the-art tools from various technological spaces (including model frameworks, relational databases, triplestores, graph databases) are not able to run such complex queries on models consisting of more than 10^7 models elements [19,9], leaving plenty of room for improvements.

Many scalability issues can be addressed by improving query performance [3]. A well-known approach for increasing query performance is *incremental query evaluation* which aims to reduce query response time by storing partial results of the query and only calculating the impact of modifications. This technique has been proven to improve performance dramatically in several scenarios (e.g. on-the-fly well-formedness validation or model synchronisation) [18], at the cost of increasing memory consumption. Since single-computer heaps cannot grow arbitrarily (due hardware and operating system limitations), memory consumption is the most significant scalability limitation.

An alternative approach to tackling MDE scalability issues is to make use of advances in persistence technology. As the majority of model-based tools uses a graph-oriented data model, recent results of the NoSQL and Linked Data movement are straightforward candidates for adaptation to MDE purposes. Unfortunately, this idea poses difficult conceptual and technological challenges as it requires mapping from traditional metamodelling languages (such as Ecore[4] to the domain property graphs or RDF. Additionally, while there are initial efforts to overcome the mapping issues between the MDE and Linked Data worlds [7], even the most sophisticated NoSQL storage technologies lack efficient and mature support for executing expressive queries incrementally.

The main goal during this PhD research is to find a solution for scaling complex queries on large models. We plan to determine the key challenges of incremental graph pattern matching in a distributed system and combine existing NoSQL technologies with a distributed graph pattern matcher algorithm. In this article, we introduce the IncQuery-D approach, an architecture that is capable of scaling up from a single-workstation tool to a cluster to handle very large models and complex queries efficiently. This paper is based on our earlier publications [8,16] about the IncQuery-D system and aims to present the research landscape for a scalable incremental query engine.

2 Related Work

A wide range of special languages have been developed to support *graph-based* representation and querying of computer data. The Ecore metamodel of the

[4] `http://www.eclipse.org/emf`

Eclipse Modeling Framework (EMF[4]) provides a modelling language, where classes along with their references and attributes are used to describe the domain. Extensive tooling helps the creation and transformation of such domain models. For EMF models, OCL is a declarative constraint description and query language that can be evaluated with the local-search based Eclipse OCL[5] engine. To address scalability issues, impact analysis tools [5] have been developed as extensions or alternatives to Eclipse OCL.

The Resource Description Framework (RDF[6]) is developed to support the description of instances of the semantic web, assuming sparse, ever-growing and incomplete data. Semantic models are built up from triple statements, which can be queried using the SPARQL graph pattern language with tools like Sesame[7]. Property graphs [14] provide a more general way to describe graphs by annotating vertices and edges with key-value properties. They can be stored in graph databases like Neo4j[8] which provides the Cypher language for defining patterns and the Gremlin language for defining graph traversals.

In the context of event-based systems, distributed evaluation engines were proposed earlier [1], scaling up in the number of rules rather than in the number of data elements. As a recent development, caching approaches based on the Rete algorithm have been proposed for the processing of Linked Data. INSTANS [13] also uses this algorithm to perform complex event processing (formulated in SPARQL) on RDF data, gathered from distributed sensors. Diamond [12] evaluates SPARQL queries on Linked Data, but it lacks an indexer so their main challenge is efficient data traversal. Trinity [17] is a closed source, pure in-memory solution, which executes a highly optimized local-search based algorithm on top of the Trinity distributed key-value store with low response time. However, the effect of data updating on query performance is currently not investigated. A recent development is a graph transformation system based on the Apache Giraph framework, which utilises the Bulk Synchronous Parallel programming paradigm [10].

As web-based collaborative modelling environments are also gaining momentum [11], they could also benefit from a distributed backend and query engine.

3 Research Plan

3.1 Research Questions

Our research proposes numerous questions, the most important among which are the following.

Architecture and data representation. Is it possible to serve multiple users concurrently? Is it possible to build a query engine which works on various backends using different data representation formats?

[5] `http://eclipse.org/modeling/mdt/?project=ocl`
[6] `http://www.w3.org/standards/techs/rdf/`
[7] `http://www.openrdf.org/`
[8] `http://neo4j.org`

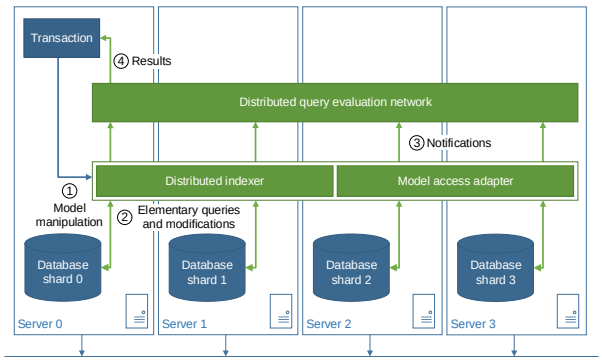

Fig. 1. The architecture of IncQuery-D, an incremental query framework.

Scalable incremental query evaluation. Is it possible to utilise an increment query evaluation algorithm in a distributed system for high-performance query evaluation?

Optimisation and dynamic reconfiguration. How can we scale and optimise such a system? How can the system adapt to the changes, both in the system (e.g. changing models and queries) and in the cloud environment of the system? How can we estimate the resources required by a certain setup?

3.2 Architecture and Data Representation

The primary goal of our research is to design an architecture that can make use of the distributed cloud infrastructure to scale out memory-intensive incremental query evaluation techniques. In our earlier work, we proposed a three-tiered architecture [16]. To maximize the flexibility and performance of the system, model persistence, indexing and incremental query evaluation are delegated to three independently distributable asychronous components (Figure 1). Consistency is ensured by synchronized construction, change propagation and termination protocols.

3.3 Scalable Incremental Query Evaluation

The Rete algorithm in a distributed environment. IncQuery-D constructs a distributed and asynchronous network of communicating nodes that are capable of producing the results set of the defined queries. Our prime candidate for this layer is the *Rete algorithm*, however, the architecture is capable of incorporating other incremental and search-based query evaluation algorithms as well. The Rete algorithm was originally proposed for rule-based expert systems [4] and later improved and adapted for EMF models in [2]. The algorithm defines a *query evaluation network*, an asynchronous network of communicating

nodes capable of producing the results of the query and incrementally maintaining the result set. The nodes of the query evaluation network can be distributed in a cluster of machines which provides a way to scale of the system.

Dimensions of scalability. The scalability of the system is affected by a number of different parameters. To implement a horizontally scalable system, we should consider the following parameters.

- *Infrastructure.* The number of machines, the available memory and computing power of each machine, the performance of the network in the cloud, the number of concurrent users of the system.
- *Model.* The size of the model, the characteristics of the model (e.g. number of nodes, edges, the structure of the graph).
- *Queries.* The number and complexity of the queries in the system.

The design of the system should consider these aspects and provide ways to scale along these parameters. Defining precise metrics for instance models and queries is an ongoing research topic.

3.4 Optimisation and Dynamic Reconfiguration

Allocation of resources. Allocating the nodes of the distributed query evaluation network to the machines in the cloud is a nontrivial task and offers the possibility for using various optimisation techniques. For example, the problem can be formalized as a constraint satisfaction problem (CSP). Furthermore, the parameters of the problem are continuously changing, including not just the memory consumption of the Rete nodes but also the number of machines (due to demands for elastic scalability) and the number of Rete nodes (due to queries being added to and removed from the system). The highly dynamic nature of such a system gives way for using more advanced optimisation techniques, such as design space exploration (DSE) [6], an optimisation technique capable of providing detailed steps for the dynamic reconfiguration of the system.

Fault-tolerance. To support high availability, the system has to be designed with fault-tolerance in mind. The most common problems are hardware failures and resource exhaustion. By careful *replication* of data and the processing nodes of the query evalution network, the system should be capable of continuing its operation. Replication can also improve the performance of the queries. Developing sharding and replication strategies is a major goal of the research.

4 Conclusion

In this paper, we outlined the current state of research for scalable incremental graph pattern matching and proposed a solution for the problem. The proposed system, IncQuery-D, is designed from the ground up for scalable query evaluation. The layered architecture of the system offers flexibility as the system is not tied to any particular data representation format or platform.

References

1. Acharya, A. et al. Implementation of Production Systems on Message-Passing Computers. *IEEE Trans. Parallel Distr. Syst.*, 3(4):477–487, July 1992.
2. G. Bergmann. *Incremental Model Queries in Model-Driven Design*. Ph.D. dissertation, Budapest University of Technology and Economics, Budapest, 2013.
3. Dimitrios S. Kolovos et al. A Research Roadmap Towards Achieving Scalability in Model Driven Engineering. In *Proceedings of the Workshop on Scalability in Model Driven Engineering*, BigMDE, New York, NY, USA, 2013. ACM.
4. C. Forgy. Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem. *Artificial Intelligences*, 19(1):17–37, 1982.
5. T. Goldschmidt and A. Uhl. Efficient OCL Impact Analysis, 2011.
6. Á. Hegedüs, Á. Horváth, I. Ráth, and D. Varró. A Model-driven Framework for Guided Design Space Exploration. In *26th IEEE/ACM Int. Conf. on ASE)*, Lawrence, Kansas, USA, 11 2011. IEEE Computer Society.
7. G. Hillairet, F. Bertrand, J. Y. Lafaye, et al. Bridging EMF applications and RDF data sources. In *Proceedings of the 4th International Workshop on Semantic Web Enabled Software Engineering, SWESE*, 2008.
8. B. Izsó, G. Szárnyas, I. Ráth, and D. Varró. IncQuery-D: Incremental Graph Search in the Cloud. In *Proceedings of the Workshop on Scalability in Model Driven Engineering*, BigMDE '13, pages 4:1–4:4, New York, NY, USA, 2013. ACM.
9. B. Izsó, G. Szárnyas, I. Ráth, and D. Varró. MONDO-SAM: A Framework to Systematically Assess MDE Scalability. In *BigMDE '14*, 2014. Accepted.
10. C. Krause, M. Tichy, and H. Giese. Implementing Graph Transformations in the Bulk Synchronous Parallel Model. In *FASE*, volume 8411 of *LNCS*, pages 325–339. Springer Berlin Heidelberg, 2014.
11. Miklos Maroti et al. Online Collaborative Environment for Designing Complex Computational Systems. In *ICCS*, Cairns, Australia, 2014. Elsevier Procedia.
12. Miranker, Daniel P. et al. Diamond: A SPARQL query engine, for linked data based on the Rete match. *AImWD*, 2012.
13. M. Rinne. SPARQL update for complex event processing. In *ISWC'12*, volume 7650 of *LNCS*. 2012.
14. M. A. Rodriguez and P. Neubauer. Constructions from dots and lines. *CoRR*, abs/1006.2361, 2010.
15. M. Scheidgen, A. Zubow, J. Fischer, and T. H. Kolbe. Automated and Transparent Model Fragmentation for Persisting Large Models. In *Proceedings of the 15th Int. Conf. on MODELS*, pages 102–118, Berlin, Heidelberg, 2012. Springer.
16. G. Szárnyas, B. Izsó, I. Ráth, D. Harmath, G. Bergmann, and D. Varró. IncQuery-D: A Distributed Incremental Model Query Framework in the Cloud. In *ACM/IEEE 17th Int. Conf. on MODELS*, Valencia, Spain, 2014. Accepted.
17. K. Zeng, J. Yang, H. Wang, B. Shao, and Z. Wang. A distributed graph engine for web scale RDF data. In *Proceedings of the 39th international conference on Very Large Data Bases*, PVLDB'13, pages 265–276. VLDB Endowment, 2013.
18. Zoltán Ujhelyi et al. Anti-pattern detection with model queries: A comparison of approaches. In *IEEE CSMR-WCRE 2014 Software Evolution Week*. IEEE, 2014.
19. Zoltán Ujhelyi et al. EMF-IncQuery: An Integrated Development Environment for Live Model Queries. *Science of Computer Programming*, 2014. Accepted.