Improving the performance of graph transformation execution in the Bulk Synchronous Parallel model

1 Introduction

The relevance and importance of graph-based analysis and modelling has increased with the growth of services such as social networks (for example Facebook) and context-aware search as well as continuously being valuable in applications such as product lifecycle management and supply chain management (SCM) [1]. Such services store large amount of data in the form of graphs and as graph sizes grows the need for the execution of graph transformations with high performance and low resource utilization increases. To reach high performance, parallelizing the execution is a viable approach. The theory of algebraic graph transformations [2] can be used as a high-level language addressing the problem of performing high performance, parallelized execution of transformations on graphs while still persevering high expressiveness. [1].

A technology which can be used for solving such problems is the Eclipse Modelling Framework (EMF) [3]. EMF is a modelling framework and code generation toolbox for building Java applications based on structured data models [4]. EMF offers a plethora of tools related to modelling. One of those, Henshin [5] provides a toolset and a model transformation language with formal graph transformation semantics [2] which offers the possibility of formal reasoning. Henshin further offers amalgamation transformation units which are especially suitable for parallel application of transformation rules by use of one of its operators [4].

Some existing solutions utilize EMF for addressing the problem of large graphs and the need for parallel execution with high performance [1] [6]. In [1], Henshin is used to create high level graph transformations with the expressiveness of algebraic graph transformations. A code generator for generating runnable code from the transformations to run on a cluster has been created [1]. The solution addresses the problem of parallel execution of transformations with high performance on graphs with millions of nodes. It is also proven to be scalable (horizontal and vertical). However, with growing data sizes, the need for adding more computing nodes increases. It is then also crucial to take extra care in the efficiency and resource utilization of the algorithm which is currently a limiting factor [1].

To address the demand of high performance on large graphs the code generator generates code adapted for the Bulk Synchronous Parallel (BSP) abstract machine [1]. BSP provides a bridging model for designing parallel algorithms which can be executed on distributed computer systems [7]. By mapping graph transformations to this model, graphs with millions of nodes can be iteratively searched and transformed [1].

A technology which is inspired by BSP is the iterative graph processing system Apache Giraph [8]. Apache Giraph is used by [1] and is built for high scalability, used at Facebook and adds several features including shared aggregators and master computations [8]. The open source software Apache Giraph builds upon the MapReduce programming model [9] implementation Apache Hadoop [10]. Another similar BSP inspired graph processing framework is Pregel developed by Google [11].

2 Statement of the problem

However the authors of [1] has seen some limitations to their approach. Over a certain number of added computation nodes the execution time does not decrease with the same rate when adding new nodes to the cluster. This can be seen by figure 1 which shows their results. As can be seen, when reaching approximately 14 nodes the benefit in shorter execution time when adding more nodes starts decreasing while the total used node time keeps increasing.

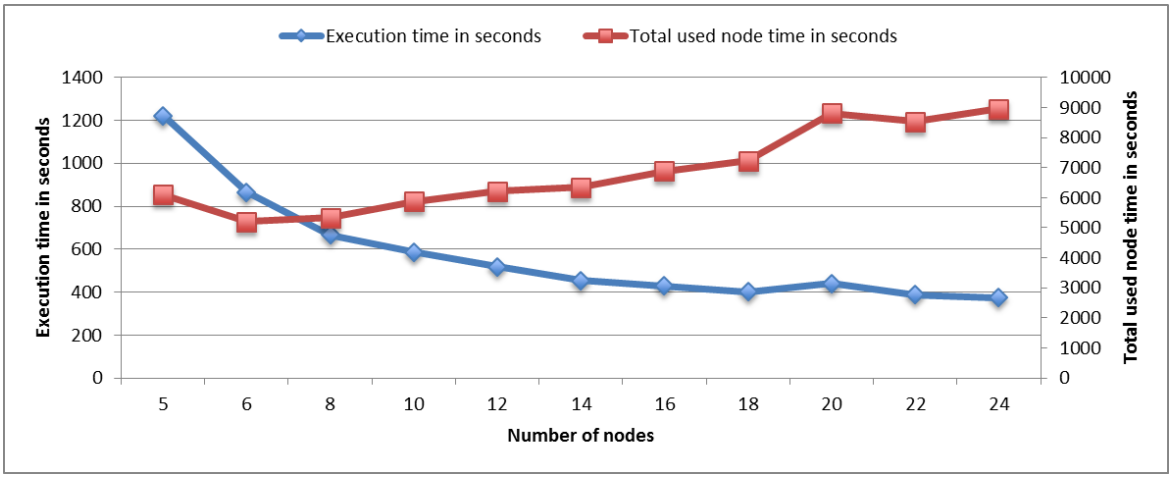

Figure 1. Shows the execution of a transformation rule on a millions of vertices large graph [1].

As the data sizes grows further the need for fast execution time and low resource utilization can be expected to increase. Therefore it exists a need to further improve the scalability of the solution provided by [1]. This while still preserving the expressiveness of a declarative high-level modelling language, which abstracts away the basic graph operations.

To understand the possible reasons for the behaviour it is important to understand the underlying computation model. Because the code generator outputs code to be run on Apache Giraph, which is inspired by the BSP computation model, the model needs to be

understood. It is also essential to understand the transformation features of Henshin as well as the by [1] constructed code generator.

BSP is an approach for efficiently processing large-scale graph data. The approach is used for designing highly parallelized algorithms. When designing the algorithms a specific scheme must be followed. The scheme consists of a series of supersteps. Each superstep consists of computation and communications which are organized in the following four steps (illustrated by figure 2):

1. Master Computation – The master node initializes the superstep by performing a single computation. The step can be used for bookkeeping and coordination of the computations performed in the vertexes.
2. Vertex Computation – Parallel step. The active vertexes performs local computations. During this step the vertex can modify the state of itself or outgoing edges as well as mutate the topology. Incoming messages from the previous step are available during the computation.
3. Communication – During vertex computations the vertex can send messages to other vertices (adjacent or by ID). The messages are received during the next superstep. Vertexes can be requested to be inactive for the next step.
4. Barrier synchronization – The local computations and the communication must be finished for every vertex before the next superstep can start.

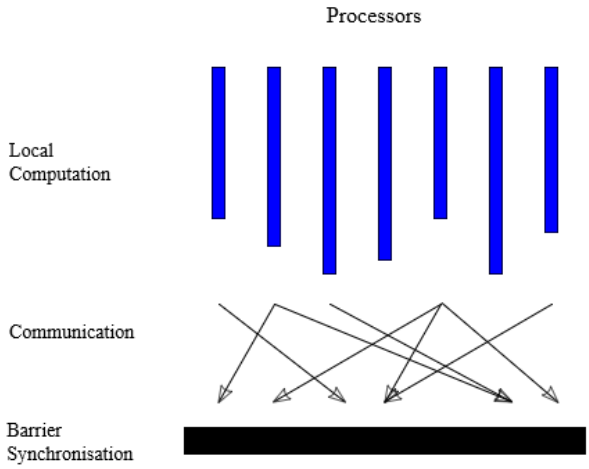

Figure 2. An Illustration of the BSP model [12].

As can be seen in figure 2 computations are typically performed in parallel to be followed by message passing. It is therefore reasonable to conclude that the communication peak somewhere in the middle of the third phase. The usage of processing power and memory is instead believed to peak during the vertex computation phase. One of those is likely to be the limiting factor and needs further research [1].

The code solution constructed by [1], which creates a code generator for BSP, works with directed graphs with typed vertexes and edges as well as primitive-typed attributes at the

vertexes. The transformations are specified using Henshin which provides declarative transformation rules and operational transformation units.

Operational transformation rules are in Henshin created by extending the graph with stereotypes as well as providing conditions for the attribute values [4]. Stereotypes, which can be placed on both edges and vertexes, can be used to create, delete, preserve, require or forbid an entity. Declarative operational transformation units are on the other hand used for defining control-flows such as fixed number iterations, for all matches, sequential execution, non-deterministic execution [4].

In the approach provided by [1], the execution of rules is applied to all matches to provide the possibility for maximum parallelization at the modelling stage. A rule means that the host graph (the graph to search and transform) are searched for a match and when a match is found the operations specified by the stereotypes are executed. To achieve maximum parallelization an iterative parallel approach is used to search the graphs [1].

The approach consists mainly of a graph pattern matching stage and a rule application (modification) stage. The matching process is further divided into local steps, which are executed according to the BSP model, where local constraints are checked. A list of partial matches are also created, extended or combined. After the matching stage transformation rule can be executed in parallel. The matching phase is said to be the most performance intensive and it is also here the room for improvement are located [1]. One way to do so is to attempt to decrease the number of partial matches generated.

3 Purpose of the study

By investigating, implementing and testing different approaches for limiting the communication overhead and the memory consumption a higher rate of performance increase when adding more computation nodes could possibly be gained. The purpose of the study is thereby to investigate and document the suitability of the different approaches.

4 Review of the literature

4.1 Foundation

Henshin: Advanced Concepts and Tools for In-Place EMF Model Transformations [4]

Authors: Arendt, Thorsten; Bierman, Enrico; Jurack, Stefan; Krause, Christian & Taentzer, Gabriele
Summary: The paper describes Henshin which is a part of the Eclipse Modelling Framework (EMF). EMF provides modelling and code generation abilities for structured data models written in Java. Henshin is a language and toolset designed for performing in-place model transformations of models created in the EMF. The language consists of structured, nested transformation units with operational semantics, which is built up using pattern-matching rules. The language has its roots within the theory of graph transformations. The presentation is used using the cases of model refactoring and meta-model evolution.

Significance: As Henshin is going to be used as the graph transformation language it is essential to understand it.

Implementing Graph Transformations in the Bulk Synchronous Parallel Model [1]

Authors: Krause, Christian; Tichy, Matthias & Giese, Holger
Summary: In this paper Graph transformation rules and units are used as a high level modelling language containing operational and declarative features. To be able to apply the language on large scale graphs and a mapping to the Bulk Synchronous Parallel model is introduced. The approach is highly distributed and parallelized. The solution is evaluated with a large dataset using a large cluster. The tools used are Henshin modelling tool and the Apache Giraph BSP framework.
Significance: This paper lays the foundation for the knowledge this thesis will build on as well as suggesting the future research which this thesis will conduct.

4.2 Related work

IncQuery-D: Incremental Graph Search in the Cloud [6]

Authors: Izsó, Benedek; Szárnyas, Gábor; Ráth, István & Varró, Dániel
Summary: The authors have identified a problem with scalability of current Model-driven engineering (MDE) tools. They have also identified solutions in other fields, although those solution create more complex and low-level queries. To address the problem of scalable MDE tools while preserving high abstraction they adapt techniques for incremental graph search. The techniques are known from the EMF project IncQuery and adapted for a distributed cloud infrastructure. The solution is claimed to be highly scalable.
Significance: The paper provides another approach for doing the study suggested. Although the authors claim the solution to be horizontal scalable, to work on very large models and efficiently handle complex queries. However it is unclear how it scales for rules with a high amount of partial matches as in the example handled by [1].

Parallelization of graph transformation based on incremental pattern matching [13]

Authors: Bergmann, Gábor; Ráth, István & Varró, Dániel
Summary: The paper presents an incremental pattern matching technique for graph transformations. The approach stores all matches as a cache which is updated upon model changes. The approach sacrifices model manipulation performance and memory consumption for faster pattern queries. Possibilities for parallelizing are also discussed.
Significance: The paper can offer ideas regarding performance versus memory usage and similar considerations. However large-scale graphs distributed over a cluster is not considered and it also uses a high amount of memory.

5 Research question and/or Hypotheses

RQ1: What problem(s) limit the scalability of the current approach?
RQ2: What methods exists for addressing the problem(s)?
For each and every possibly found method:

RQ3: How much performance gain and memory consumption decrease does it give?
RQ4: What kind of graphs is the solution suitable for?

6 The Design – Methods and Procedures

The research will be conducted in the form of design science and follow the framework with the seven principles presented by [14].

The first guideline,*"Design as an Artifact"* [14], will be approached by the production of a viable instantiation artifact. The instantiation will be in the form of a software tool which aims to improve the by [1] already constructed tool with respect to trying to give a positive answer on RQ3. The software tools will be based on the technology presented in Section 1 and 2. The design also has to take into consideration the basic rules of parallel computing, the ACID (Atomic, Consistent, Isolated and Durable) properties

To address the second guideline, *"Problem relevance"* [14], the work will develop a solution which is aimed to solve the relevant business problem stated in section 1. The solution will attempt to answer the for the guideline relevant research questions RQ1 and RQ2.

Further guideline 3, *"Design evaluation"* [14], will be fulfilled by the rigorous demonstration of the about the different instantiations utility, quality and efficacy using both a real data case and purposefully designed artificial benchmarks. Dynamic analysis, using the real data set and the artificial benchmarks, will evaluate for the dynamic qualities asked for by research questions RQ3 and RQ4.

A possible choice for the real data set is the one used by [14], the IMDB movie database. Then the solution will be possible to more easily relate to the already available statistics. It will also be beneficial to use a real data set to evaluate the benefit of the instantiation has on real data.

On the other hand. the set of artificial benchmarks will be purposefully designed for use to evaluate on which cases the solution performs better and on which cases (e.g. sparse vs dens graphs) it will have worse performance. Thus for aiming to responding to RQ4. It will also be possible to relate the different solutions to each other and possibly come up with a hybrid solution.

The fourth guideline, *"Research Contribution"* [14], is going to be achieved by making the contribution clear and verifiable in the area of the design instantiation (the EMF framework and the BSP model) and the design foundation by aiming to improve and extend the available knowledge.

By applying rigorous methods for the construction of the instantiation and when designing the methods for evaluation the fifth guideline, *"Research rigor"*, shall be taken into consideration. The steps undertaken shall be explicitly validated, using the above described evaluation plan, and purposefully described.

Guideline 6, *"Design as search process"* [14], is going to be expressed in the way of conducting the research. The research will be conducted in an iterative manner using the generate/test cycle presented by [14]. During the generate phase the research questions RQ1 and RQ2 will be attempted to answered and during the test cycle the RQ3 and RQ4 will be answered utilizing the described evaluation approaches. The result will serve as feedback for the next cycle.

Some thoughts have already been put into what methods are going to be researched during the first cycles. In short they are as follows:

- Store, in each vertex, information about the surrounding nodes. Then the vertex will have more information and therefore have the possibility to perform more local execution and is not in the same need of sending messages. This will also raise the memory consumption which will be traded off against the possible performance gain. Another possible outcome using this approach is that the memory consumption actually decreases due to being able create fewer partial matches.
- Another possibility is to spread the steps executed. The nodes will then not all be sending messages at the same superstep. Vertices with odd ID could be in one step and vertices with even ID could be in another step. It will then also possibly even out the memory consumption because different steps have different memory requirements.

The last guideline, *"Communication of research"* [14], is going to be followed by apart from describing the solution in a technical manner for practitioners and researcher within the technology also describe the solution with business aim. This will according to [14] enable behaviour science to research the use of the instantiation within an organization.

7 Limitations and Delimitations

The concepts underlying the solution will hopefully not be limited to Henshin and Apache Giraph but the solution will not be evaluated using other platforms. This because evaluating using other platforms will take too much time.

8 Significance of the study

The study will attempt to further develop the available work in this area and therefore the researcher involved in the area will benefit from it.

If the results are beneficial they could also be used for current distributed graph applications in the business. One example of this is doing transformations on the huge graphs of user data the social networks have gathered.

9 References

[1] C. Krause, M. Tichy and H. Giese, “Implementing Graph Transformations in the Bulk Synchronous Parallel Model,” in *Proc. of the 17th International Conference on Fundamental Approaches to Software Engineering (FASE), June 7-10,* Grenoble, France, 2014.

[2] H. Ehrig, K. Ehrig, U. Prange and G. Taentzer, Fundamentals of Algebraic Graph Transformation. Monographs in Theoretical Computer Science., Springer, 2006.

[3] “EMF: Eclipse Modeling Framework.,” [Online]. Available: http://www.eclipse.org/modeling/emf/.

[4] T. Arendt, E. Bierman, S. Jurack, C. Krause and G. Taentzer, “Henshin: Advanced Concepts and Tools for In-Place EMF Model Transformations,” *MoDELS'10: Proc. 13th international conference on Model Driven Engineering Languages and Systems,* vol. 6394, pp. 121-135, 2010.

[5] “Henshin,” [Online]. Available: http://www.eclipse.org/henshin/.

[6] G. S. I. R. a. D. V. Benedek Izsó, “IncQuery-D: Incremental Graph Search in the Cloud,” in *Proc. of BigMDE ’13, ACM*, 2013.

[7] L. Valiant, “A bridging model for parallel computation.,” *Commun. ACM,* vol. 33, no. 8, pp. 103-111, 1990.

[8] “Apache Giraph - Welcome to Apache Giraph!,” The Apache Software Foundation, 9 August 2014. [Online]. Available: http://giraph.apache.org/. [Accessed 4 October 2014].

[9] J. Dean and S. Ghemawat, “MapReduce:simplifieddataprocessingonlargeclusters.,” *Commun. ACM,* vol. 51, no. 1, p. 107–113, January 2008.

[10] T. A. S. Foundation, “Welcome to Apache™ Hadoop®!,” The Apache Software Foundation, 12 September 2014. [Online]. Available: http://hadoop.apache.org/. [Accessed 8 October 2014].

[11] G. Malewicz, M. Austern, A. Bik, J. Dehnert, I. Horn, N. Leiser and G. Czajkowski, “Pregel: a system for large-scale graph processing,” *SIGMOD 2010, ACM,* p. 135–146, 2010.

[12] Discboy, “Wikipedia,” 4 November 2006. [Online]. Available: http://en.wikipedia.org/wiki/File:Bsp.wiki.fig1.svg. [Accessed 21 October 2014].

[13] G. Bergmann, I. Ráth and D. Varró, “Parallelization of graph transformation based on incre- mental pattern matching,” in *ECEASST 18*, 2009.

[14] A. R. Hevner, S. T. March and J. Park, “Design science in information system research,” *MIS Quartely,* vol. 28, no. 1, pp. 75-105, 2004.