Towards Distributed Model Transformations with LinTra

Loli Burgueño[1], Manuel Wimmer[2], and Antonio Vallecillo[1]

[1] Universidad de Málaga. Spain. {loli,av}@lcc.uma.es
[2] TU Wien, Austria. wimmer@big.tuwien.ac.at

Abstract. Performance and scalability of model transformations are becoming prominent topics in Model-Driven Engineering. In previous works we introduced LinTra, a platform for executing model transformations in parallel. LinTra is based on the Linda model of a coordination language and is intended to be used as a middleware where high-level model transformation languages are compiled. In this paper we present the initial results of our analyses on the scalability of out-place model-to-model transformation executions in LinTra when the models and the processing elements are distributed over a set of machines.

1 Introduction

The performance and scalability of model transformations is gaining interest as industry is progressively adopting model-driven techniques, multicore computers are becoming commonplace and the cloud is extensively used. However, existing model transformation engines are mostly based on sequential and in-memory execution strategies, and thus their capabilities to transform large models in parallel and distributed environments are limited.

In previous works we introduced LinTra [2,3], a platform for executing model transformations in parallel. LinTra is based on the Linda model of a coordination language and is intended to be used as a middleware where high-level model transformation languages are compiled. Currently LinTra outperforms existing sequential and parallel model transformation engines, but we also wanted to study how the distribution aspects affect the performance of LinTra. In this paper we present the initial results of our analyses on the scalability of out-place model-to-model transformation executions in LinTra when the models and the processing elements are distributed over a set of machines.

This paper is organized as follows. After this introduction, Sect. 2 briefly describes the background of our proposal and the running example we use in the paper to illustrate our approach. Then, Sect. 3 describes an implementation and evaluation based on the case study and, finally, Sect. 4 draws some conclusions and oulines future work.

2 Backgound

2.1 LinTra

LinTra [2, 3] is a framework for the parallel execution of model transformations. It uses data-parallelism and the Blackboard paradigm [4] to store the input and output

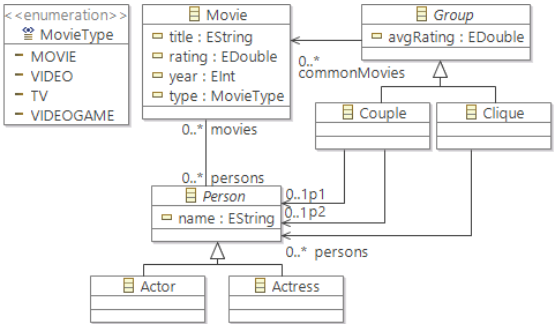

Fig. 1. IMDb Metamodel taken from [5].

models, as well as the required data to keep track of the MT execution that coordinates the agents that are involved in the transformation process. One of the key aspects of LinTra is the model and metamodel representation, wherein every entity is assigned an identifier, which is used to reference objects and to represent relationships between them. Relationships between entities are represented by storing in the source entity the identifier(s) of its target entity(ies).

In LinTra, traceability is implemented implicitly using a bidirectional function that receives as a parameter the entity identifier of the input model and returns the identifier of the output entity, regardless whether the output entities have already been created or not. This means that LinTra does not store information about traces explicitly; thus, the performance is not affected by the search for trace information.

In order to carry out the transformation process in parallel, LinTra uses data-parallelism and the Master-Slave design pattern [4]. The master's job is to launch slaves and coordinate their work. Each slave is in charge of applying the transformation to submodels of the input model (i.e., partitions) as if each partition is a complete and independent model. Since in LinTra's out-place mode the complete input model is always available, in case the slaves have data dependencies with elements that are not in the submodels they were assigned, they only have to query the Blackboard to retrieve them.

2.2 Running example

The example that we have selected uses the "Movie Database" (IMDb) proposed in the Transformation Tool Contest (TTC) 2014 [5], whose metamodel is shown in Figure 1. For exercising the distribution aspects we simply use the identity transformation, which permits checking how fast the complete model graph can be traversed and copied, i.e., focusing on the communication aspects more than on the computational aspects of the transformation itself.

3 Implementation and Evaluation

The goal is to be able to know how the distribution of data and processes affects the performance of the model transformation in LinTra. Instead of using one parallel machine,

what happens when the model is distributed among several machines, or the slaves are executed in different nodes?

In general, the analysis of the effect of data and processes distribution on the performance of a model transformation is a complex task, and a comprehensive study should deserve its own line of research. However, to initially answer this research question we have conducted some experiments in order to estimate some limits of the execution times of a model transformation when either the model, the slaves, or both, are distributed among several machines. For that we used two machines (A and B), with 4 cores each, connected in a LAN via TCP/IP. We used a very basic communication platform to implement the distributed blackboard, based on Java HashMaps to store the models and sockets for communicating the machines—hence avoiding the layers that any technological solution such as LevelDB or Cassandra would add. Our goal was to obtain the lower bounds for the execution times, independently from any solution.

We selected the IMDb-Identity model transformation case study[3], which basically traverses a 5 million elements model and produces a copy of it as the target model. We considered five configurations:

- **Config.0**: One local machine (A). All slaves are executed locally. Both source and target models are stored locally.
- **Config.1**: Two machines (A and B). All slaves are executed in A. Source and target models are stored (mingled) in both machines: elements with even identifiers in one and with odd identifiers in the other, in order to exacerbate data distribution.
- **Config.2**: Two machines (A and B). All slaves are executed in A. Source model stored in A and target model in B (best case for distribution).
- **Config.3**: Two machines (A and B). Half of the slaves are executed in each machine. Source and target models are stored both machines, mingled as in *Config.1*.
- **Config.4**: Two machines (A and B). Half of the slaves are executed in each machine. Source and target models are stored in both machines, but taking care that each slave only handles data stores locally.

For each configuration we executed the transformation on subsets of the complete model of increasing size, emulating different sizes of the database model to check how different model transformation engines scale up. The resulting execution times are shown in the left table of Fig. 2. The right table shows the speed-ups when compared with Config.0.

It is interesting to observe how data distribution has a significant impact on the model transformation execution performance. Resulting times for Config.1 and Config.3, in which the data is evenly spread and forces all slaves to access half of the data in each machine, suffer from heavy network access. Config.2 and Config.4, on the contrary, offer good results. Of course, when all slaves are executed in one machine and the target model resides remotely, the network access introduces some delay (an average speed-up of 3.8 according to our results). Config.4, in turn, provides the best case because it gets the two machines working in parallel and handling just local data (apart from the access to the shared area to gets the jobs, which resides only in machine A). The average speed-up of 0.39 is a significant achievement for this best case.

[3] http://atenea.lcc.uma.es/index.php/Main_Page/Resources/LinTra#IMDb

Execution times (seconds)					
# Elems	Config.0	Config.1	Config.2	Config.3	Config.4
200	0.05	6.65	0.03	11.85	0.01
2,000	0.04	26.85	0.09	43.24	0.02
20,000	0.11	292.84	0.39	422.51	0.03
100,000	0.84	1,602.08	3.02	3,343.25	0.09
200,000	1.53	3,087.76	6.20	3,781.01	0.74
300,000	2.14	5,364.64	12.24	3,460.83	1.01
400,000	3.00	7,230.36	13.29	7,246.77	1.44
1,000,000	16.50	24,855.59	40.02	27,726.10	3.99
2,000,000	18.38	49,711.19	90.97	47,134.36	11.53
5,000,000	52.97	104,393.49	314.42	113,122.47	26.38

Speed-up					
# Elems	Config.0	Config.1	Config.2	Config.3	Config.4
200	1.00	141.43	0.68	252.13	0.30
2,000	1.00	745.69	2.61	1,201.00	0.42
20,000	1.00	2,762.65	3.69	3,985.91	0.30
100,000	1.00	1,911.79	3.60	3,989.55	0.11
200,000	1.00	2,020.79	4.06	2,474.48	0.48
300,000	1.00	2,511.53	5.73	1,620.24	0.47
400,000	1.00	2,410.92	4.43	2,416.39	0.48
1,000,000	1.00	1,506.03	2.43	1,679.96	0.24
2,000,000	1.00	2,704.93	4.95	2,564.72	0.63
5,000,000	1.00	1,970.95	5.94	2,135.76	0.50
Average	***1.00***	***1,868.67***	***3.81***	***2,232.01***	***0.39***

Fig. 2. Execution times (left) and relative speedups (right) of distributed configurations.

Data and process distribution may have a significant impact on the performance of a model transformation. As expected, the results are very sensitive to the way in which data and processes are distributed among the nodes. We have run some experiments to show some initial results, but this issue deserves further investigations and more detailed analysis to provide more generalized results.

4 Conclusions and Future Work

This position paper has presented some initial experiments that try to estimate the effect of both data and process distribution on the performance of LinTra model transformations. The study conducted here is specific for that solution. We plan to compare these results with the emerging MT engines that also provide distribution (e.g. [1]). We also want to use different technological platforms that offer data distribution to evaluate their performance when connected with LinTra. More precisely, we want to check Infinispan with LevelBD as persistence layer whose results with LinTra are better than other data-grids when executed on a single machine [3], and the combination of Apache Spark and Cassandra. Finally, we also want to use UDP instead of TCP to see the speed-up obtained by simplifying the communication protocol used.

References

1. Benelallam, A., Gómez, A., Tisi, M., Cabot, J.: Distributed model-to-model transformation with ATL on MapReduce. In: Proc. of SLE 2015. pp. 37–48. ACM (2015)
2. Burgueño, L.: On the Quality Properties of Model Transformations: Performance and Correctness. Ph.D. thesis, Universidad de Málaga (April 2016)
3. Burgueño, L., Wimmer, M., Vallecillo, A.: A Linda-based platform for the parallel execution of out-place model transformations. Information & Software Technology. To appear
4. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M.: Pattern-Oriented Software Architecture: A System of Patterns. Wiley, Chichester, UK (1996)
5. Horn, T., Krause, C., Tichy, M.: The TTC 2014 Movie Database Case. In: Proc. of the 7th Transformation Tool Contest (TTC 2014). vol. 1035, pp. 93–97. CEUR Workshops (2014)