Architectural challenges in creating a high-performant model-transformation engine

Tamás Fekete, Gergely Mezei

Modeling is a frequently used concept in computer science. One of its main application areas is the Model-driven Engineering (MDE). Due to the essential role of model processing in MDE, it is important to find and apply efficient model-transformation algorithms. Several techniques exist, one of the most popular among them is the graph rewriting-based model transformation. Graph rewriting is based on subgraph isomorphism, which is an NP complete problem. Since MDE tools must often handle huge models, performance is a key feature. However, graph rewriting tends to slow down sharply as the size of the input model increases. The usage of strongly parallelized algorithms is a way to increase the performance. Nowadays, there are several kinds of hardware components supporting efficient evaluation of parallelized algorithms. OpenCL framework is the most popular interface supporting heterogeneous architectures (e.g. CPU, GPU, APU, DSP) [1]. The comparison of the OpenCL framework and other vendor specific solutions is studied in many papers e.g. in [2]. According to the classification of graph transformation approaches [3], none of them can use the advantages of the OpenCL framework. Our overall goal is to fill in this gap and create a high-performant general model-transformation engine based on the OpenCL framework. The engine we are creating is referred to as the GPGPU-based Engine for Model Processing (GEMP). In [4] we have worked out and evaluated the first part (pattern matching) of model transformations. However, model transformation is not complete without the ability of rewriting the pattern. In this paper, we take this step and describe the desinging and implementation challenges and our solution. The performance is studied on experimental way using a case study.

OpenCL versions are backwards compatible. Since NVIDIA GPUs support OpenCL 1.2 only, we have decided to use this version in order to maximize the hardware independence. The implementation is applied in C++11 (STL and Boost libraries are used), we are using the test-driven development methodology (TDD). The evaluation of the unit tests provided immediate feedback about the state of the implementation continuously from the beginning, which was a great help in judging our progress. **As the main goal is to achieve a high-performant model-transformation engine, the following four main points are discussed in the current paper:** (i) The description of an efficient OpenCL-based pattern matching algorithm. (ii) As far as we use several devices, we need to synchronize their work in order to achieve the best performance (e.g.: CPU + GPU). The synchronization algorithm is elaborated. (iii) Scalability is studied because of the limited memory usage (e.g.: dividing the input/output data). (iv) The data structures and the relations between those are also highly important because of the high performance and low memory usage.

The novelties of the current paper are the followings: (i) The detailed architecture of a high-performant model-transformation engine supporting both pattern matching and rewriting. (ii) A collection of acceleration techniques which are applied and (iii) optimization models to find the best configuration for the model-transformation. The efficiency of our solution is proved through case studies. Results are introduced and evaluated in the paper.

In general, model-transformation consists of three main steps: (i) Finding topological matches in the graph fulfilling both the negative and the positive constraints of the pattern. (ii) Processing the attributes of the result of the first step and thus evaluating the constraints referring to attributes. (iii) Rewriting the pattern. The first two steps are executed on OpenCL devices, while the third is executed directly on the host.

If the input model must be divided into partitions then the whole process is repeated as soon as processing of the partition of the input data is finished. Vertices, which cannot be processed on the OpenCL device (because of the partitions) are queued and evaluated on the host as soon as the host has free capacity.

Both the positive and negative constraints are evaluated together in two different steps. The reason for dividing the matching algorithms into two steps is that copying all vertices of the host model with all of their attributes to the OpenCL device memory would be inefficient. By dividing this process into two strongly connected steps, we can reduce the data to be copied. In this way, we need to copy the attributes only for those vertices, which are part of a topological matching structure. In the first two steps in GEMP, there are two separated kernel codes. The third step of graph rewriting is the rewriting of the graph, namely replacing tha pattern found in the input graph as specified by the rewriting rule. This step is managed on the host side, without the usage of any OpenCL device. This is necessary, since matches may overlap and threads are working in parallel. This means that the patterns found by the OpenCL devices may be invalid at this time (invalidated by other threads). Therefore, the host acts as a synchronization point and ensures the validity of the patterns before rewriting.

Our overall goal is to create a GPU-based model transformation engine referred to as GEMP. The main contribution of the paper is the rewriting part of the transformation. This part is elaborated in detail. Moreover, the paper presents the architecure of the solution and illustrates the machanisms, analyzes the results by case studies. As the domain for the case studies, the movie database, IMDB [5] was chosen. Although our goal is not fully complete, the preliminary results are promising which means that GEMP can be effectively used as an accelerator engine in all MDE tools.

References

[1] K. Group. [Online]. Available: https://www.khronos.org/opencl/. [Accessed 15 January 2016].

[2] Veerasamy, Bala Dhandayuthapani; Nasira, G. M.. *Exploring the contrast on GPGPU computing through CUDA and OPENCL*, Journal on Software Engineering vol. 9, issue 1, p. 1-8, (2014).

[3] E. Jakumeit, S. Buchwald, D. Wagelaar, L. Dan, A. Hegedus, M. Herrmannsdorfer, T. Hornf, E. Kalnina, C. Krause, K. Lano, M. Lepper, A. Rensink, L. Rose, S. Watzoldt and S. Mazanek, *A survey and comparison of transformation tools based on the transformation tool contest*, Special issue on Experimental Software Engineering in the Cloud(ESEiC), vol. 85, no. 1, pp. 41-99, 2014.

[4] F. Tamás and M. Gergely, *Creating a GPGPU-accelerated framework for pattern matching using a case study*, in EUROCON 2015, International Conference on Computer as a Tool, Salamance, Spain, 2015.

[5] *IMDb database source, Alternative Interfaces*, [Online]. Available: www.imdb.com/interfaces. [Accessed 15 January 2016].