Towards a High-Performance Modelica Compiler

Giovanni Agosta[1] Emanuele Baldino[1] Francesco Casella[1] Stefano Cherubin[1] Alberto Leva[1]
Federico Terraneo[1]

[1]DEIB, Politecnico di Milano, Italy, `given_name.family_name@polimi.it`

Introduction

Large-scale models are becoming increasingly common in a variety of relevant application fields: *smart grids*, *detailed thermal simulations* and *coarse-scale fluid dynamics* are just some examples.

Mainstream Modelica toolchains, however, scale poorly for systems approaching or exceeding the one-million equation target. The code generation time is unacceptably large. The memory footprint of the generated simulation code is another scalability concern, as it has an impact on simulation speed due to CPU cache misses. Although the need to extend existing Modelica implementations to support large models is recognized by the Modelica community (Frenkel et al., 2011; Casella, 2015), significant effort is still required to effectively support large-scale systems.

Even considering the recent introduction of sparse solvers (Braun et al., 2017), existing Modelica toolchains perform heavy structural analysis optimization passes that scale poorly for large-scale system, and lose structural information during the flattening phase, such as arrays and looping constructs. As a consequence the generated C code does not exploit CPU architectures effectively. Furthermore, the C code generation phase does not take into account architectural optimizations, and simply generates unoptimized C code. This approach passes the burden of optimization to the C compiler, to the detriment to the overall translation efficiency and runtime performance.

New approach to Modelica compilation

In this paper we argue that significant performance improvements of Modelica toolchains could be achieved if an integrated approach is adopted, where high-level information from the Modelica source, instead of being transferred to an imperative language compiler, is used directly to produce architecture-optimized machine code. To scale the Modelica language to large-scale problems, a change of perspective is thus required, where a Modelica *compiler* – not just a translator – can perform model-specific and architectural-specific optimizations in an integrated way.

Our proposal aims at improving the code generation process without any impact on the Modelica syntax and semantics, thus being fully compatible with existing Modelica models and libraries.

A high-performance Modelica compiler needs to preserve arrays, `for` loops, and in general the object-oriented structure of the models as much as possible, in order to factor out common behaviour (= equations) in large-scale models. Thus it is possible to achieve much faster code generation, a much smaller memory fooprint, and hence much faster code execution thanks to the vastly reduced chances of cache misses.

The modularization of *flow* variables in connection sets and improvements to structual analysis for the compilation of systems of index greater than one are also areas of future research to achieve our goal.

We argue that to achieve this result in a cost-effective way, said Modelica compiler has to be integrated in an existing compiler framework.

For this reason, we propose to design a Modelica compiler integrated in LLVM (Lattner and Adve, 2004), which is a state of the art compiler framework, designed with the explicit goals of modularity and extensibility.

The authors form an inter-disciplinary research group within the Dipartimento di Elettronica, Informazione e Bioingegneria of Politecnico di Milano, which includes strong competences in the areas of Modelica and object-oriented modelling and simulation, Computer Architectures and Compiler Design.

This on-going work is today at a very early stage of development. The main goal of this paper is thus to present this group's vision and roadmap, as well as to present some initial results of a very early prototype.

References

W. Braun, F. Casella, and B. Bachmann. Solving large-scale Modelica models: new approaches and experimental results using OpenModelica. In *Proc. 12th International Modelica Conference*, pages 557–563, Prague, Czech Republic, 2017. doi:10.3384/ecp17132557.

F. Casella. Simulation of large-scale models in Modelica: State of the art and future perspectives. In *Proc. 11th International Modelica Conference*, pages 459–468, Versailles, France, 2015.

J. Frenkel, C. Schubert, G. Kunze, P. Fritzson, M. Sjölund, and A. Pop. Towards a benchmark suite for Modelica compilers: Large models. In *Proc. 8th International Modelica Conference*, pages 143–152, Dresden, Germany, 2011.

C. Lattner and V. Adve. LLVM: A compilation framework for lifelong program analysis & transformation. In *Proc. 2004 International Symposium on Code Generation and Optimization*, pages 75–86, Palo Alto, CA, USA, 2004.