PAMM · Proc. Appl. Math. Mech. **14**, 945 – 946 (2014) / **DOI** 10.1002/pamm.201410453

Platform-independent Description of Imaging Algorithms

Harald Köstler[1,*]

[1] FAU, Department Informatik, Lehrstuhl für Systemsimulation, Cauerstr. 11, 91058 Erlangen

In the last years real-time imaging and more complex models for the various imaging applications became feasible mostly because of the progress made in parallel computer architectures like found in GPUs and modern multi-core CPUs. However, the implementation effort increases, if one wants to achieve good performance. A solution to this problem is to formulate the imaging algorithms in an abstract way in a domain-specific language (DSL) and then automatically generate efficient C++ or CUDA code. A multi-layered approach is sketched that allows users to describe applications in a natural way from the mathematical model down to the program specification.

1 Introduction

Variational models are a common class of mathematical models used for imaging applications like image denoising, image segmentation, or image registration. Typically, one formulates an energy functional or a (time-dependent) partial differential equation (PDE) that have to be discretized and minimized resp. solved numerically. Often, models for different applications require similar building blocks like (sparse) matrix vector products or other linear algebra operations. However, efficient implementations have to take into account the concrete hardware platform and the context within the numerical algorithms and thus one often has to re-implement similar operations.

Within the project *Advanced Stencil-Code Engineering* (EXASTENCILS)[1] we propose a multi-layered domain-specific language that provides an interface to users and developers of imaging algorithms. In figure 1 on the left side these layers are summarized starting from the continuous variational model, its discretization, application and algorithm specific settings, down to a program specification that is close to source code.

2 Code Generation Prototype

A first prototype for this concept was realized in the Scala language. Here, DSL descriptions on the different layers are parsed, transformed into an implementation-oriented representation, and then pretty-printed to C++ or CUDA source code. An important feature is that the user only has to specify layer 1, all other layers can be automatically generated from layer 1 using domain-specific knowledge. This becomes possible due to the limited scope of the domain, i.e. currently variational models are supported that require the numerical solution of an elliptic PDE via multigrid. However, the users can adapt the descriptions on all layers manually to include further applications. Figure 1 shows part of the DSL programs for doing HDR compression in the gradient domain as described in [1]. Here, one has to solve the PDE

$$\Delta u = f \quad \text{in } \Omega \tag{1a}$$

$$u = 0 \quad \text{on } \partial\Omega \tag{1b}$$

A second application is image denoising via a nonlinear isotropic complex diffusion process modeled via a time-dependent PDE [2, 3]. Note that it results in completely different source code compared to solving the simple PDE from eq. (1). Table 1 lists some preliminary solver runtimes for generated codes running on a quad-core Intel Xeon Processor E5-1620 v2 and an NVIDIA GeForce GTX 680.

Table 1: Measured runtimes in milliseconds for one multigrid V(2,2)-cycle with a Jacobi smoother ($\omega = 0.8$). The image size is $N = 4096^2$ resp. $N = 256^3$. Direct coarsening down to less than three pixels per direction on the coarsest grid is applied.

	HDR Compression		Image Denoising	
	CPU	GPU	CPU	GPU
2D	156	60	3023	224
3D	202	63	3217	275

* Corresponding author: e-mail harald.koestler@fau.de, phone +49 9131 85 28359, fax +49 9131 85 28928

[1] http://www.exastencils.org/

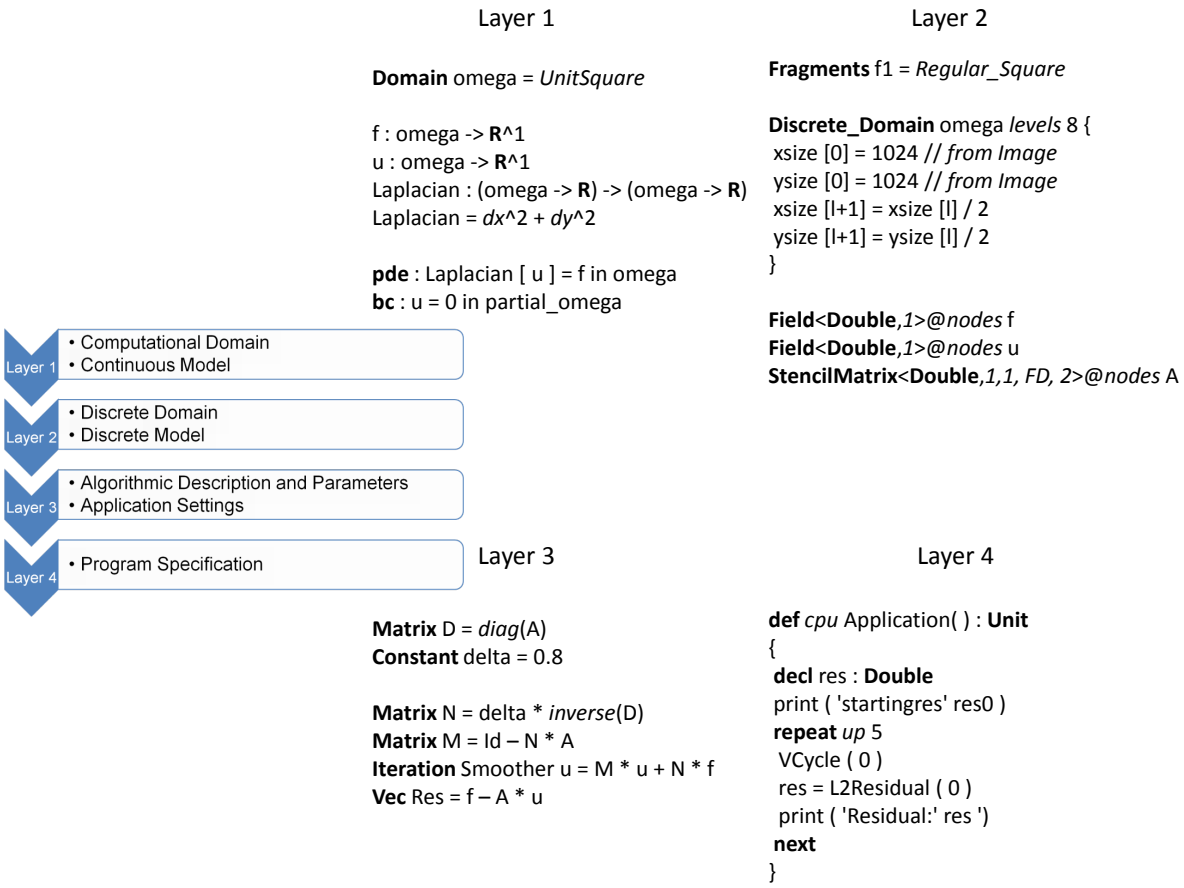

Fig. 1: Different layers for describing a prototypical variational imaging application (left) and exemplary DSL description of parts of the HDR compression application.

3 Future Work

Clearly, it is a very challenging project to fully develop such a multi-layered DSL. Currently, we optimize the generated code [4,5], add more multigrid components, improve the DSLs and the transformations framework in Scala [6], add distributed memory parallelization for larger applications [7], and apply domain-specific optimizations [8].

Acknowledgements This work was supported by the German Research Foundation (DFG) through the Priority Programme 1648 "Software for Exascale Computing" (SPPEXA).

References

[1] H. Köstler, M. Stürmer, and T. Pohl, Journal of Real-Time Image Processing pp. 1–13 (2013).
[2] G. Gilboa, N. Sochen, and Y. Zeevi, IEEE Transactions on Pattern Analysis and Machine Intelligence **26**(8), 1020–1036 (2004).
[3] I. Dietrich, R. German, H. Köstler, and U. Rüde, Modeling multigrid algorithms for variational imaging, in: Proceedings of 21st Australian Software Engineering Conference (ASWEC2010), (IEEE Computer Society Washington, DC, USA, 2010), pp. 224–234.
[4] S. Kronawitter and C. Lengauer, Optimization of two jacobi smoother kernels by domain-specific program transformation, in: Proceedings of the 1st International Workshop on High-Performance Stencil Computations (HiStencils), (2014), pp. 75–80.
[5] R. Membarth, F. Hannig, J. Teich, and H. Köstler, Towards Domain-specific Computing for Stencil Codes in HPC, in: High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion, (2012), pp. 1133–1138.
[6] C. Schmitt, S. Kuckuk, H. Köstler, F. Hannig, and J. Teich, An evaluation of domain-specific language technologies for code generation, in: Proceedings of the 14th International Conference on Computational Science and Its Applications (ICCSA 2014), (2014), to appear.
[7] S. Kuckuk, B. Gmeiner, H. Köstler, and U. Rüde, A generic prototype to benchmark algorithms and data structures for hierarchical hybrid grids, in: Proceedings of the International Conference on Parallel Computing (ParCo), (2013), pp. 813–822.
[8] A. Grebhahn, N. Siegmund, S. Apel, S. Kuckuk, C. Schmitt, and H. Köstler, Optimizing performance of stencil code with spl conqueror, in: Proceedings of the 1st International Workshop on High-Performance Stencil Computations (HiStencils), (2014), pp. 7–14.

