ExaStencils Advanced Stencil-Code Engineering

The German Research Foundation's German Priority Programme 1648 "Software for Exascale Computing" (SPPEXA) is nearing the end of its second of six years. 13 projects started in January 2013 to address various challenges of exascale computing. In this issue, we present project ExaStencils.

ExaStencils is being pursued by eight principal investigators of five research groups at three locations. At the University of Passau, there are the Chairs of Programming (Christian Lengauer and Armin Größlinger) and of Software Product Lines (Sven Apel), at the Friedrich-Alexander-Universität Erlangen-Nürnberg the Chairs of System Simulation (Ulrich Rüde and Harald Köstler) and of Hardware-Software-Co-Design (Jürgen Teich and Frank Hannig), and at the University of Wuppertal the Applied Computer Science Group (Matthias Bolten).

The central goal of ExaStencils is to develop a radically new software technology for applications with exascale performance. To reach this goal, the project focuses on a comparatively narrow but very important application domain. The aim is to enable a simple and convenient formulation of problem solutions in this domain. The software technology developed in ExaStencils shall facilitate the highly automatic generation of a large variety of efficient implementations via the judicious use of domain-specific knowledge in each of a sequence of optimization steps such that, at the end, exascale performance results.

The application domain chosen is that of stencil codes, i.e., compute-intensive algorithms in which data points in a grid are redefined repeatedly as a combination of the values of neighboring points. This neighborhood pattern is called a stencil. Stencil codes are used for the solution of discrete partial differential equations and the resulting linear systems. To obtain a performance-competitive, highly automated software technology, the domain is restricted further to multigrid methods [1].

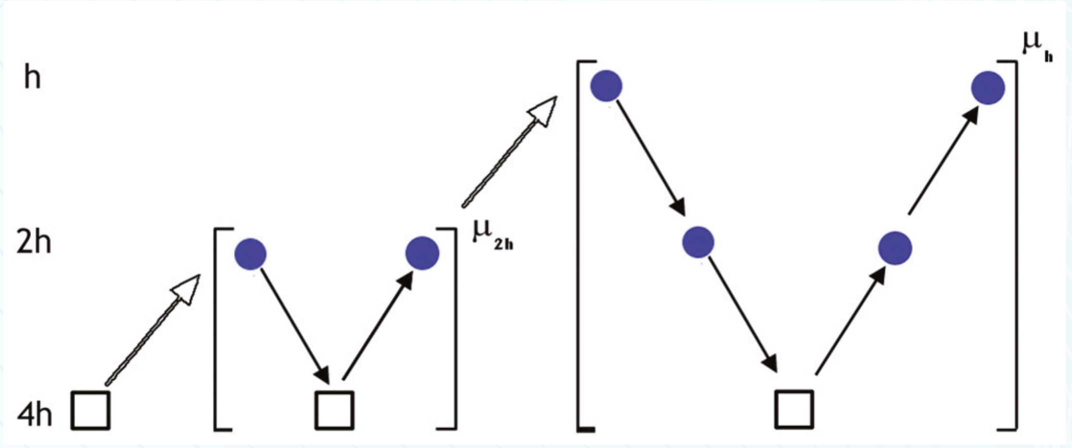

Figure 1: An F-cycle as a succession of V-cycles.

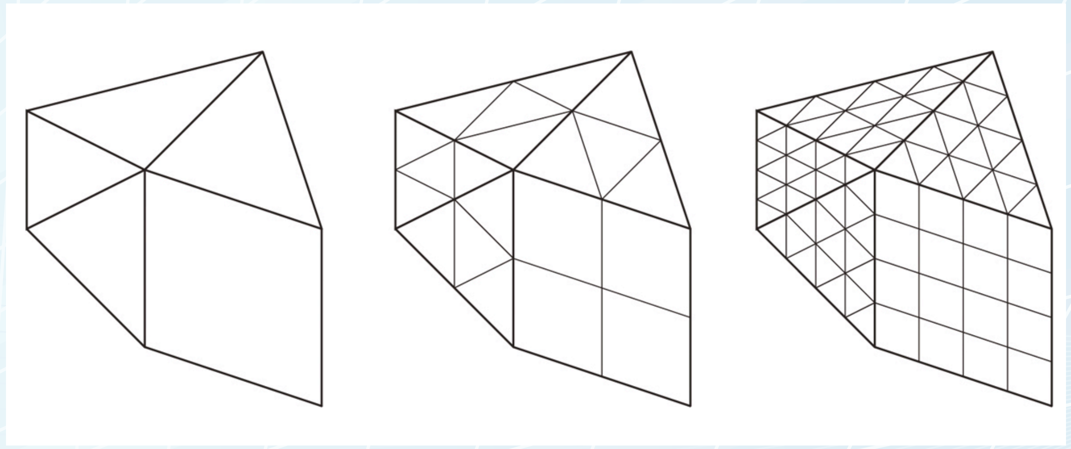

Figure 2: Successive refinement of a hierarchical hybrid grid (left to right).

Multigrid methods involve stencil computations on a hierarchy of very fine to successively coarser grids. On the coarser grids, less processing power is required and communication dominates. A multigrid method is characterized by two strategies: (1) a smoothing strategy, which is used to smooth the sampling error of the grid at hand, and (2) a coarsening strategy, which transfers data from one grid to the next coarser grid. Once one arrives at the coarsest level, one refines the grid again via some form of interpolation. This cycle of coarsening and refining is called a V-cycle. Various cycling strategies are commonly used. For instance, an F-cycle multigrid method consists of a sequence of progressively deeper V-cycles (see Fig. 1). The technology for the efficient implementation and a systematic performance engineering of parallel multigrid methods is a major current research topic [2].

ExaStencils also restricts the structure of the grids. Considered are so-called hierarchical hybrid grids: at the coarsest level, the grid is unstructured, but refinements of each segment must be homogeneous, though each segment may exhibit a different structure (see Fig. 2).

Present-day stencil codes are implemented in general-purpose programming languages, such as Fortran, C, or Java, or derivates thereof, and harnesses for parallelism, such as MPI, OpenMP or OpenCL. ExaStencils favors a much more domain-specific approach with languages at several layers of abstraction, the most abstract being the mathematical formulation, the most concrete the optimized target code. At every layer, the corresponding language expresses not only computational directives but also domain knowledge of the problem and platform to be leveraged for optimization. This approach will enable a highly automated code

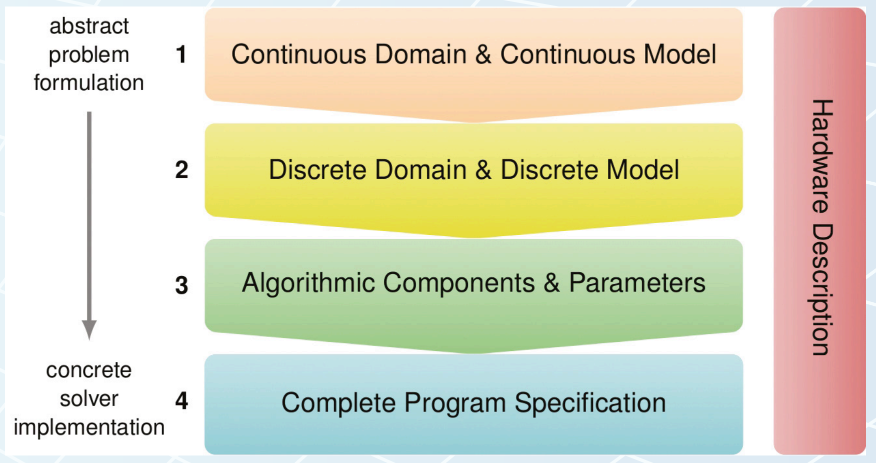
abstract problem formulation
concrete solver implementation
1
Continuous Domain & Continuous Model
2
Discrete Domain & Discrete Model
3
Algorithmic Components & Parameters
4
Complete Program Specification
Hardware Description

Figure 3: The ExaStencils DSL with four layers of abstraction. Layers 1-2 address the concerns of application scientists, Layers 2-3 those of mathematicians, and Layers 3-4 those of computer scientists.

generation at all layers and has been demonstrated successfully before in the U.S. projects FFTW [3] and SPIRAL [4] for certain linear transforms.

At the center of the project is a Scala-based code generator for a wide range of stencil codes in the domain, which is currently under development. It takes code formulated in an external domain-specific language at four different layers of abstraction (see Fig. 3). The ultimate vision is that application scientists will program at the most abstract layers, and the additional information specified at the lower layers will be generated automatically based on an analysis of the specific problem to be solved and information on the execution platform at hand. A preliminary version of the code generator in existence already demonstrates the feasibility of the ExaStencils approach [5]. It produces target code in C++ with OpenMP and CUDA. Distributed architectures are one of the main focuses of the next version of the generator framework, which is currently under development.

One major innovation in ExaStencils is that it views stencil codes not as individuals but as members of a family. The domain-specific specification pinpoints the commonalities that the code shares with the other codes of the family, and the variabilities in which it departs from the other codes. Each point of variability comes with a number of options or alternatives. The idea is that application scientists, and the ExaStencils compiler and run-time system, choose suitable options from these variabilities – and no more has to be specified to obtain a custom-optimized implementation.

In first, yet hand-coded, experiments on Jülich's BlueGene/Q JUQUEEN, we employed the Highly Scalable Multigrid Solver [6] for hierarchical hybrid grids. Commonalities and variabilities are usually specified in terms of a variability model. The variability model for the Highly Scalable Multigrid Solver is illustrated in Fig. 4. Each node denotes a configuration option - in our case, the choice of a coarse grid solver, a smoother, and pre- and post-smoothing parameter values which must satisfy the condition that their sum is greater than zero. A selection of configuration options gives rise to an executable variant of the stencil code.

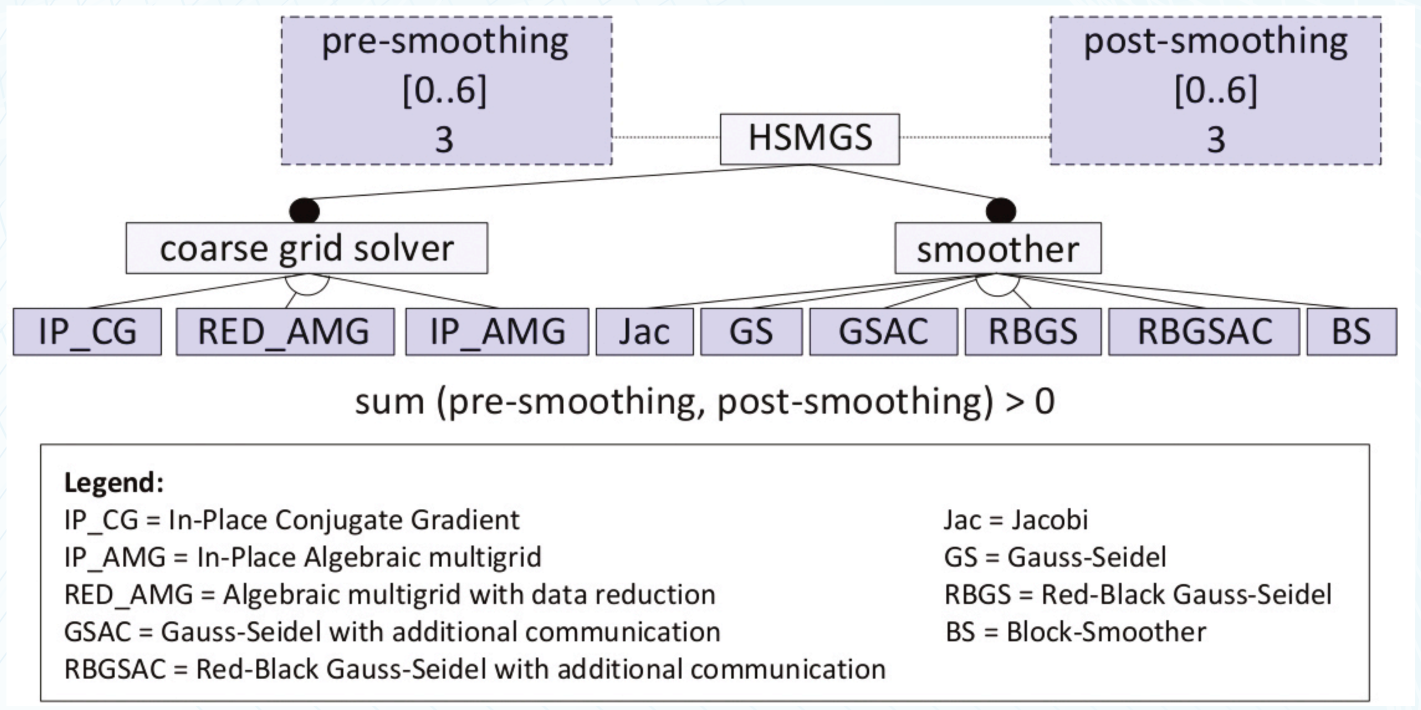
pre-smoothing
[0..6]
3
post-smoothing
[0..6]
3
HSMGS
coarse grid solver
smoother
IP_CG
RED_AMG
IP_AMG
Jac
GS
GSAC
RBGS
RBGSAC
BS
sum (pre-smoothing, post-smoothing) > 0
Legend:
IP_CG = In-Place Conjugate Gradient
IP_AMG = In-Place Algebraic multigrid
RED_AMG = Algebraic multigrid with data reduction
GSAC = Gauss-Seidel with additional communication
RBGSAC = Red-Black Gauss-Seidel with additional communication
Jac = Jacobi
GS = Gauss-Seidel
RBGS = Red-Black Gauss-Seidel
BS = Block-Smoother

Figure 4: Concrete variability model for the Highly Scalable Multigrid Solver.

Which configuration options (i.e., which choices of algorithmic components, alternatives of data structures, and parameter values) contribute to maximal performance is obvious in some cases and very surprising in others. To make this problem tractable, ExaStencils will provide a capability of recommending suitable combinations of configuration options, based on a machine-learning approach [7].

With project ExaStencils, we hope to provide proof of the application relevance of the ExaStencils paradigm of domain-specific stencil code engineering and to encourage experts of other suitable domains to take a similar approach. For up-to-date information, please visit the project's Web site at www.exastencils.org.

Acknowledgements

We gratefully acknowledge the financial support fo the Priority Research Initiative 1648 "Software for Exascale Computing", funded by the German Research Foundation.

References

[1] Trottenberg, U., Osterlee, C. W., Schüller, A.
Multigrid, Academic Press, 2000

[2] Gmeiner, B., Köstler, H., Stürmer, M., Rüde, U.
Parallel multigrid on hierarchical hybrid grids: A performance study on current High Performance Computing clusters. Concurrency and Computation: Practice and Experience 26(1), pp.217-240, Jan. 2014

[3] Frigo, M., Johnson, S. G.
The design and implementation of FFTW3, Proc. IEEE 93(2), pp.216-231, Feb. 2005

[4] Püschel, M., Franchetti, P., Voronenko, Y.
Spiral. In: Encyclopedia of Parallel Computing, 1920–1933, Padua, D. A. et al. (eds.), Springer, 2011

[5] Köstler, H., Schmitt, C., Kuckuk, S., Hannig, F., Teich. J., Rüde, U.
A Scala Prototype to Generate Multigrid Solver Implementations for Different Problems and Target Multi-Core Platforms. Computing Research Repository (CoRR), pp.18, arXiv:1406.5369, June 2014

[6] Kuckuk, S., Gmeiner, B., Köstler, H., Rüde, U.
A generic prototype to benchmark algorithms and data structures for hierarchical hybrid grids. In: Proc. Int. Conf. on Parallel Computing (ParCo), pp.813-822, IOS Press, 2013

[7] Grebhahn, A., Siegmund, N., Apel, S., Kuckuk, S., Schmitt, C., Köstler, H.
Optimizing Performance of Stencil Code with SPL Conqueror. In: Proc. Int. Workshop on High-Performance Stencil Computations (HiStencils), Größlinger, A. and Köstler, H. (eds.), pp.7-14, www.epubli.de, Jan. 2014

contact: Prof. Dr. Christian Lengauer, lengauer@fim.uni-passau.de

- Sven Apel[1]
- Matthias Bolten[2]
- Armin Größlinger[1]
- Frank Hannig[3]
- Harald Köstler[3]
- Christian Lengauer[1]
- Ulrich Rüde[3]
- Jürgen Teich[3]

[1] University of Passau

[2] University of Wuppertal

[3] Friedrich-Alexander-Universität Erlangen-Nürnberg