# Integrating STELLA & MODESTO: Definition and Optimization of Complex Stencil Programs

Tobias Gysi
Department of Computer Science
ETH Zurich
tobias.gysi@inf.ethz.ch

Torsten Hoefler (Advisor)
Department of Computer Science
ETH Zurich
htor@inf.ethz.ch

## I. Introduction

Stencil computations are an important algorithmic motif that commonly appears in various application domains such as atmospheric modeling, seismic imaging, and electromagnetic simulations. Rather than applying a single stencil iteratively, many of these applications execute different dependent stencils in succession. We can represent such complex stencil programs using a directed acyclic graph whose nodes and edges represent the stencils and their data dependencies respectively.

Due to their low arithmetic intensity stencil codes are typically heavily memory bandwidth bound. An effective optimization of complex stencil programs therefore requires data-locality transformations. For example, fusion of successive stencil evaluations allows to buffer intermediate results in the cache.

STELLA [1] is a domain specific language embedded (DSL) in C++ that allows to optimize the execution of successive stencils. A single source code written using STELLA can be compiled for different target architectures such as multi-core or many-core processors.

MODESTO [2] is a tool that allows to select data-locality transformations for a given stencil program, target architecture combination. A stencil algebra allows to enumerate different stencil program implementation variants whose performance is evaluated using a compile-time performance model.

By integrating STELLA and MODESTO, we can fully automate the tuning of complex stencil programs. We demonstrate the effectiveness of our approach using example kernels from the COSMO [3] atmospheric model.

## II. Data-locality Transformations

Data-locality transformations facilitate a memory bandwidth efficient implementation of complex stencil programs. Using a combination of loop tiling and loop fusion, we apply multiple dependent stencils to a single tile of the stencil evaluation domain and buffer intermediate results in the cache. Thereby, we satisfy data dependencies at the tile boundaries using redundant computation or halo exchange communication. Furthermore, we employ hierarchical tiling to target multiple memory hierarchy levels.

In STELLA, we can fuse consecutive stencils on virtual tiling hierarchy levels using the DSL elements highlighted in Figure 1. For example, we can either put two consecutive stencils in separate sweeps or fuse their execution using a single sweep. To support different target architectures STELLA provides back-ends that implement hardware architecture specific data-locality transformations. In particular, each back-end defines its own tiling hierarchy including a mapping to the virtual tiling hierarchy of STELLA. Such a mapping might omit or insert tiling hierarchy levels to match the requirements of the particular target architecture. Using the concepts of a virtual tiling hierarchy and target architecture specific back-ends STELLA allows to write performance portable stencil codes.

```
// stencil definition (functors)
struct Lap { ... };
struct Fli { ... };
...
// stencil program initialization (DSL)
Stencil stencil;
StencilCompiler::Build(
  stencil,
  pack_parameters(
    ...
  ),
  define_temporaries(
    StencilBuffer<lap, double>(),
    StencilBuffer<fli, double>(),
    ...
  ),
  define_loops(
    define_sweep(
      StencilStage<Lap, IJRange<-1,1,-1,1> >(),
      StencilStage<Fli, IJRange<-1,0,0,0> >(),
      ...
)));
// stencil program execution
stencil.Apply();
```

Fig. 1: Illustration of the STELLA syntax

## III. Analysis & Performance Modeling

Given a stencil program, MODESTO enumerates different implementation variants using our stencil algebra. An algebra element defines a stencil execution order and fuses the stencils on different tiling hierarchy levels. We represent algebra elements using nested bracket expressions. For example, the following expression fuses the stencils $lap$ and $fli$ respectively $flj$ and $out$ on the second tiling hierarchy level:

$$[[lap, fli], [flj, out]].$$

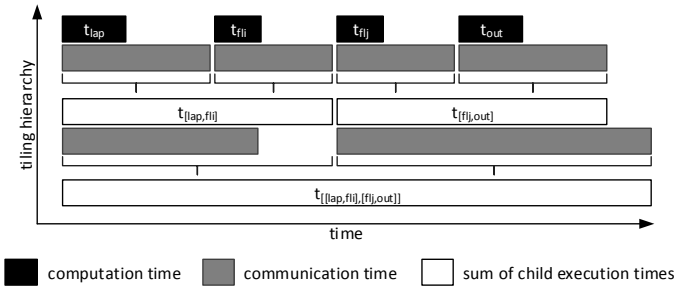
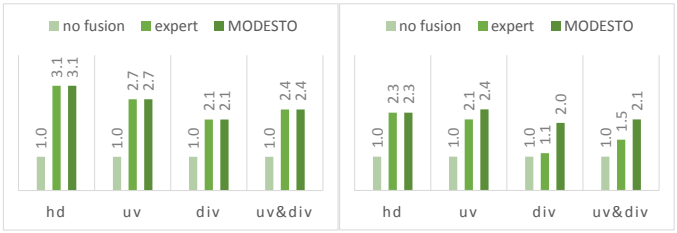



Fig. 2: Stencil program execution time estimation




(a) Speedup Intel Core i5-3330 (b) Speedup Nvidia Tesla K20c

Fig. 3: Speedup of expert- and MODESTO-tuned kernels

We analyze the performance of a stencil program implementation variant using the concept of affine sets and maps. Thereby, we represent tiling optimizations using maps that relate each point of the stencil evaluation domain to a tile identifier. Additional maps model the data dependencies between different stencil evaluations. By following the data dependencies starting from the output points, MODESTO computes the amount of computation and communication performed by a stencil program.

Similar to the Roofline model [4], we estimate the stencil program execution time using peak throughputs. Thereby, we consider the bandwidth requirements on all tiling hierarchy levels as shown by Figure 2.

## IV. Optimization

Using a combination of exhaustive search and dynamic programming MODESTO searches the optimal element in the set of all stencil program implementation variants $I$. Thereby, we aim at minimizing the execution time $t(x)$ while respecting the memory size limits $M^l$ on all tiling hierarchy levels.

$$\begin{aligned} &\underset{x \in I}{\text{minimize}} \quad t(x) \\ &\text{subject to} \quad m^l(x) \leq M^l \qquad l = 1, \ldots, m \end{aligned}$$

## V. Implementation & Evaluation

We evaluated our approach using four example kernels from the COSMO atmospheric model. All experiments were performed on an Intel Core i5-3330 CPU and a Nvidia Tesla K20c GPU. Figure 3 shows that MODESTO-tuned kernels achieve the same or better performance than expert-tuned kernels. Furthermore, due to loop fusion the tuned kernels attain a 2.0–3.1x speedup compared to their naive counterparts. This demonstrates the effectiveness of the data-locality transformations implemented by STELLA.

## References

[1] T. Gysi, C. Osuna, O. Fuhrer, M. Bianco, and T. C. Schulthess, "Stella: A domain-specific tool for structured grid methods in weather and climate models," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '15, 2015.

[2] T. Gysi, T. Grosser, and T. Hoefler, "Modesto: Data-centric analytic optimization of complex stencil programs on heterogeneous architectures," in *Proceedings of the 29th ACM on International Conference on Supercomputing*, ser. ICS '15. New York, NY, USA: ACM, 2015, pp. 177–186.

[3] G. Doms and U. Schättler, "The nonhydrostatic limited-area model LM (Lokal-Modell) of the DWD. Part I: Scientific documentation," German Weather Service (DWD), Germany, Tech. Rep., 1999. [Online]. Available: http://www.cosmo-model.org/

[4] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65–76, Apr. 2009.