Geophysical Research Abstracts
Vol. 16, EGU2014-8464-2, 2014
EGU General Assembly 2014

STELLA: A domain-specific embedded language for stencil codes on structured grids

Tobias Gysi (1,2), Oliver Fuhrer (3), Carlos Osuna (4), Benjamin Cumming (5), and Thomas Schulthess (5)

(1) Supercomputing Systems AG, Zurich, Switzerland, (2) Swiss Federal Institute of Technology (ETH Zurich), Scalable Parallel Computing Lab, Switzerland, (3) Swiss Federal Office of Meteorology and Climatology MeteoSwiss, Zurich, Switzerland, (4) Center for Climate Systems Modeling (C2SM), ETH Zurich, Switerland, (5) Swiss National Supercomputing Center (CSCS), Lugano, Switzerland

Adapting regional weather and climate models (RCMs) for hybrid many-core computing architectures is a formidable challenge. Achieving high performance on different supercomputing architectures while retaining a single source code are often perceived as contradicting goals. Typically, the numerical algorithms employed are tightly inter-twined with hardware dependent implementation choices and optimizations such as for example data-structures and loop order. While Fortran is currently the de-facto standard for programming RCMs, no single such standard for porting such models to graphics processing units (GPUs) has yet emerged. The approaches used can be grouped into three main categories: compiler directives (OpenACC, PGI compiler directives), custom programming languages (CUDA, OpenCL) and domain-specific libraries or languages.

STELLA (STencil Loop LAnguage) is a domain-specific embedded language (DSEL) built using generic programming in C++ which is targeted at stencil codes on structured grids. It allows a high-level specification of the algorithm while separating hardware dependent implementation details into back-ends. Currently, a back-end for multi-core CPUs using the OpenMP programming model and a back-end for NVIDIA GPUs using the CUDA programming mode has been developed. We will present the domain-specific language and its features such as software managed caching. With the example of an implementation of the dynamical core of a RCM (COSMO) we will compare performance with respect to the original Fortran implementation both on both CPUs and GPUs. Finally, we will discuss advantages and disadvantages of our approach as compared to other approaches such as source-to-source translators.