Kai Germaschewski

Using automated code generation to support high performance implicit extended MHD integration

University of New Hampshire
8 College Rd
Durham
NH 03824
kai.germaschewski@unh.edu
Amitava Bhattacharjee
Joachim Raeder

Recently, significant progress has been made in developing scalable implicit methods for time integration of resistive and extended MHD time integration. These methods use inexact Newton solves and physics-based preconditioning for the inner linear solves. As opposed to algebraic linear solvers, for which optimized high-performance kernels exist and are reusable across a large class of problems, these new methods employ matrix-free (or matrix-light) algorithms. As a simple example, GMRES only needs to perform Jacobian-vector products, which can be approximated as finite-difference directional derivatives. Therefore, this new class of algorithms spends a major part of its computational time in problem-specific r.h.s. evaluations.

Automatic code generation is a technique that takes the specification of an algorithm at a higher abstraction level and creates a well-tuned implementation from it. For finite-volume / finite-difference based discretizations, this higher abstraction level can be a stencil computation, which needs to be performed at every grid point in the integration domain. It could also be specified at an even higher level closer to the mathematical description of the problem, for example by defining the fluxes and leaving the implementation of the discrete divergence to the code generator. At the backend, a code generator can feature different modules which generate optimal code for specific hardware architectures, for example conventional architectures (x86) but using SIMD instructions (e.g. SSE2), or accelerator-based architectures like the Cell processor or GPGPUs. The problem-specific code (in this example, the discretized system of equations) can be agnostic to the actual hardware used, a high-performance implementation tailored to the specific architecture will be generated automatically. While a conventionally written code will typically only achieve a couple percent of peak performance, the automatically generated version can improve performance by a factor of four or more even on conventional hardware, and provide even higher gains on accelerator-based machines.