Performance-Portable Finite-element Computations from High-level Specifications with FFC and PyOP2

Florian Rathgeber*, Graham R. Markall*, Lawrence Mitchell‡, Nicolas Loriant*, David A. Ham*†, Carlo Bertolli§ and Paul H.J. Kelly*

* Department of Computing, Imperial College London, London SW7 2AZ, United Kingdom
{f.rathgeber, graham.markall08, n.loriant, david.ham, p.kelly}@imperial.ac.uk

† Grantham Institute for Climate Change, Imperial College London, London SW7 2AZ, United Kingdom

‡EPCC, The University of Edinburgh, Edinburgh EH9 3JZ, United Kingdom - lawrence.mitchell@ed.ac.uk

§IBM T.J. Watson Research Center, NY, USA - cbertol@us.ibm.com

***Abstract*—We present a tool chain for the fully automated synthesis of performance-portable finite-element solvers for multicore and GPGPU platforms from high-level specifications. Our runtime code generation and just-in-time compilation pathway takes finite-element forms in the domain-specific language UFL to low-level code. Automatically generated finite-element assembly kernels are passed to PyOP2, a domain-specific language for mesh-based simulation codes, which acts as an intermediate abstraction layer for executing the numerical kernels in parallel over an unstructured mesh. Easy integration of our tool chain allows transparently adding performance portability to existing simulation codes.**

PyOP2 [1], [2] is a Python implementation of the unstructured mesh computation framework OP2 [3], which applies numerical kernels in parallel over an unstructured mesh. Kernels and parallel loop invocation code are just-in-time (JIT) compiled at runtime and cached. Subsequent parallel loops using the same kernel do not incur additional compilation overhead. PyOP2 delivers performance portability across a range of hardware platforms: OpenMP, OpenCL and MPI on multi-core CPUs and CUDA or OpenCL on GPGPU. Data layout, mesh partitioning, traversal, parallel scheduling and efficient execution of parallel loops are automatically managed.

We demonstrate the finite-element tool chain shown in Figure 1, using the domain-specific Unified Form Language UFL [4] and the form compiler FFC [5] from the FEniCS project and PyOP2 as a suitable intermediate representation for parallel kernel execution. Finite-element methods are widely used to approximately solve partial differential equations on unstructured domains. UFL allows the weak forms of PDEs to be expressed in near-mathematical notation. The local assembly operation executes the same kernel for every entity of the mesh and is therefore a natural fit for the PyOP2 computation model. We show how these kernels are generated automatically from the weak form of an equation given in UFL. Global assembly and linear solves are passed through to platform-specific linear algebra backends integrated into PyOP2 through a modular interface.

Using this tool chain, scientists can drive finite-element computations from an input notation very close to the mathematical model and transparently benefit from performance-portable parallel execution on their hardware architecture of choice without requiring specialist knowledge in numerical analysis or parallel programming.

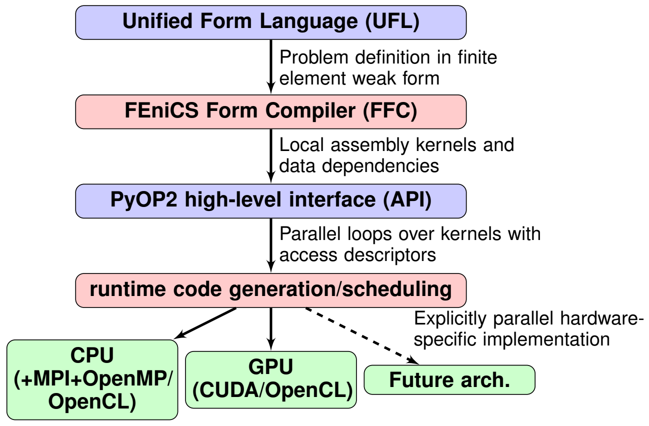
Unified Form Language (UFL)
Problem definition in finite element weak form
FEniCS Form Compiler (FFC)
Local assembly kernels and data dependencies
PyOP2 high-level interface (API)
Parallel loops over kernels with access descriptors
runtime code generation/scheduling
Explicitly parallel hardware-specific implementation
CPU (+MPI+OpenMP/ OpenCL)
GPU (CUDA/OpenCL)
Future arch.

Fig. 1. Overview of the UFL/PyOP2 finite-element code synthesis tool chain

Automatic generation of low-level code facilitates rapid development and allows generation of variants that support optimized performance and explore alternative code generation schemes. We plan to look at hybrid CPU/GPU execution, improved communication overlap, intra-kernel vectorization and warp-wide parallelization, and variants of local vs. global assembly. The key in each case is to adapt the implementation to the application context and the hardware capabilities.

References

[1] F. Rathgeber, G. R. Markall, L. Mitchell, N. Loriant, D. A. Ham, C. Bertolli, and P. H. J. Kelly, "PyOP2: A High-Level Framework for Performance-Portable Simulations on Unstructured Meshes," in *WOLFHPC2012: Workshop on Languages for High-Performance Computing at SC '12*, November 2012, in press.

[2] G. R. Markall, F. Rathgeber, L. Mitchell, N. Loriant, C. Bertolli, D. A. Ham, and P. H. J. Kelly, "Performance portable finite element assembly using PyOP2 and FEniCS," in *Proceedings of the International Supercomputing Conference (ISC) '13*, ser. Lecture Notes in Computer Science, vol. 7905, June 2013, in press.

[3] G. R. Mudalige, I. Reguly, M. B. Giles, C. Bertolli, and P. H. J. Kelly., "OP2: An Active Library Framework for Solving Unstructured Mesh-based Applications on Multi-Core and Many-Core Architectures," in *Innovative Parallel Computing conference (InPar '12)*, 2012.

[4] M. S. Alnaes, A. Logg, K. B. Oelgaard, M. E. Rognes, and G. N. Wells, "Unified form language: A domain-specific language for weak formulations of partial differential equations," *arXiv*, Nov. 2012.

[5] R. C. Kirby and A. Logg, "A compiler for variational forms," *ACM Transactions on Mathematical Software*, vol. 32, no. 3, pp. 417–444, 2006.