DOE Project Final Report

Domain Specific Language Support for Exascale

OSU PI: Ponnuswamy Sadayappan
Project PI: Daniel Quinlan (LLNL)

OSU Project #: DE-SC0008844
Project Start: 09/01/2012
Project End: 08/31/2016

In this final report, we summarize accomplishments along multiple directions from the work done at the Ohio State University towards the larger goals of the project.

A) Multi-Target Code Generation for Stencil Computations:

We developed a framework to automatically generate high-performance code for multi-core CPUs, GPUs and FPGAs from a single DSL stencil program input. Several code generators were built and evaluated. The framework enables performance portability and high productivity, and relies on advanced compilation techniques to generate target-specific code. The approach can be integrated in C/C++ programs, and Matlab (via Mex calls). The code generation techniques capture domain-specific information to enable aggressive optimization (e.g., check convergence in iterative solvers only every xx time steps), achieve data locality and parallelization via spatial/temporal tiling, and employ target-specific knowledge (e.g., for short-vector SIMD, a dedicated data layout transformation is used to avoid stream alignment conflicts).

Relevant Publications

SDSLc: A Multi-Target Domain-Specific Compiler for Stencil Computations, P. Rawat, M. Kong, T. Henretty, J. Holewinski, K. Stock, L.-N. Pouchet, J. Ramanujam, A. Rountev, and P. Sadayappan. *International Workshop on Domain-Specific Languages and High-Level Frameworks for High Performance Computing (WOLFHPC'15).*

Released Software: Available at http://hpcrl.cse.ohio-state.edu/wiki/index.php/SDSL

B) Domain-Specific Optimization for High-Order Stencils:

We developed a novel technique based on associative reordering to greatly enhance performance of high-order stencil computations. High-order stencils arise in high-accuracy numerical solution approaches for PDEs in various domains. Their performance decreases as order is increased, due to excessive register pressure and

consequent sub-optimal SIMD code. A new optimization was developed to enable significantly enhanced performance for high-order stencils, on multi-core processors, by exploiting associativity of stencil operations and global semantics information to perform instruction reordering. The approach generates high-performance codes of > 10,000 lines automatically from < 20 lines DSL description, using ROSE/PolyOpt and dedicated SIMD compiler.

Relevant Publications

A Framework for Enhancing Data Reuse via Associative Reordering, K. Stock, M. Kong, L.-N. Pouchet, T. Grosser, F. Rastello, J. Ramanujam, and P. Sadayappan. *ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'14).*

Released Software: Available from the ROSE/D-TEC branch and http://hpcrl.cse.ohio-state.edu/wiki/index.php/HOSTS

C) High-Performance Automatic Code Generation for Stencils on GPUs:

We developed a new approach for GPU code generation for 3D Stencils. Scarce shared-memory capacity poses severe challenges for parallel time-tiled GPU execution of 3D stencils. To address this limitation, we developed new resource-centric approach to determining effective fusion+tiling for 3D stencils, by exploiting associative reordering of stencil operations and use of registers to reduce demands on shared memory. The developed stencil optimizer achieves higher performance on 3D stencils than existing ones.

Relevant Publications

Effective resource management for enhancing performance of 2D and 3D stencils on GPUs, P. Rawat, C. Hong, M. Ravishankar, V. Grover, L.-N. Pouchet, and P. Sadayappan. *9th Annual Workshop on General Purpose Processing using Graphics Processing Unit (GPGPU '16).*

Resource Conscious Reuse-Driven Tiling for GPUs, P. Rawat, C. Hong, M. Ravishankar, V. Grover, L.-N. Pouchet, and P. Sadayappan. *2016 International Conference on Parallel Architectures and Compilation (PACT '16).*

Released Software: Not yet publicly released; alpha version available upon request.

D) PAdvisor Padding Advisor Tool:

We developed a new approach for optimal array padding. Padding is a well-known practically used technique to add dummy array locations to avoid conflict misses in cache. Current practice uses heuristics for extent of padding. PAdvisor is a tool based

on a new approach for compiler analysis for optimal conflict-free array padding. Many adaptively refined meshes naturally have power-of-two sizes; padding significantly affects performance. Our efficient padding tool can enable co-tuning of padding with tile size selection, which is sub-optimal if decoupled.

Relevant Publications

Effective padding of multidimensional arrays to avoid cache conflict misses, Changwan Hong, Wenlei Bao, Albert Cohen, Sriram Krishnamoorthy, Louis-Noël Pouchet, Fabrice Rastello, J. Ramanujam, and P. Sadayappan*. 2016 ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI '16).*

Released Software: Available at http://hpcrl.cse.ohio-state.edu/wiki/index.php/PAdvisor.

E) Task-based Programming on Distributed-Memory Clusters:

We introduced a new macro-dataflow programming environment for distributed-memory clusters, based on the Intel Concurrent Collections (CnC) runtime. Our language extensions let the user define virtual topologies, task mappings, task-centric data placement, task and communication scheduling, etc. along with the macro-dataflow graph describing the application. We developed a new compiler to automatically generate programs using the Intel CnC C++ run-time, with key automatic optimizations including task coarsening and coalescing to adapt the granularity of tasks to the program/target machine. We experimentally validated our approach on a variety of scientific computations mostly from dense linear algebra, showing our approach can match and even outperform Scalapack in certain situations, while requiring only a few lines of DSL programming to describe the parallel algorithm strategy.

Relevant Publications

M. Kong, L.-N. Pouchet, V. Sarkar and P. Sadayappan, PIPES: A Language and Compiler for Task-based Programming on Distributed-Memory Clusters*. IEEE/ACM Conference on Supercomputing (SC'16).*

Released Software: Not yet publicly released; alpha version available upon request.

F) PolyCheck Verification Tool:

We developed a tool to verify the correctness of program transformations. Guaranteeing the correctness of program transformations is essential, and to date three main approaches have been developed: proof of equivalence of affine programs, matching the execution traces of programs, and checking bit-by-bit equivalence of program outputs. Each technique suffers from limitations in the kind of transformations supported, space complexity, or the sensitivity to the testing dataset. We developed a novel approach that addresses all three limitations to provide an automatic bug checker

to verify any iteration reordering transformations on affine programs, including non-affine transformations, with space consumption proportional to the original program data and robust to arbitrary datasets of a given size. We achieve this by exploiting the structure of affine program control- and data-flow to generate at compile-time lightweight checker code to be executed within the transformed program. Experimental results assess the correctness and effectiveness of our method and its increased coverage over previous approaches.

Relevant Publications

W. Bao, S. Krishnamoorthy, L.-N. Pouchet, F. Rastello, and P. Sadayappan, PolyCheck: dynamic verification of iteration space transformations on affine programs. *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL '16).*

Released Software: Not yet publicly released; alpha version available upon request.