Annotated Bibiliography of Scientific Code Automation

October 8, 2007

1 Efficiency–expressiveness trade-off

Analysa [2] combined efficiency and expressiveness by using a functional programming language (AlScheme) as a scripting language which linked with C, C++ and Fortran code for efficiency.

The Broadway compiler [18] supports domain-specific compiler optimizations. It provides compiler support for a wide range of domains and in the context of existing programming languages using a technique called 'library-level optimization' which recognizes and exploits the domain-specific semantics of software libraries.

Ken Kennedy proposed the use of 'telescoping languages' [22] in which fully optimized low-level code would be included as high-level operations in an extended language.

2 Compilers

Automatic generation of code has been performed in compiler design [15, 21].

3 Solving PDE's: the FEM

Optimization of code for solving differential equations has been studied widely [31, 32, 45, 46]. Many of these approaches have been based on the finite element method (FEM).

3.1 FIAT and SyFi

The evaluation of finite element basis functions, and related inner-product data has been automated by FIAT [24, 25] and by SyFi [37].

3.2 FErari

In [26], efficient evaluation of finite element matrices was addressed. This paper posed a complex optimization problem that was further studied in [29]. In [30], a different type of optimization, based on the geometric properties of certain tensors, was explored.

The paper [23] examines the effect of using complexity-reducing relations to generate optimized code for the evaluation of finite element variational forms. The optimizations are implemented in a prototype code named FErari. The authors demonstrate that by invoking FErari as an optimizing backend to the form compiler FFC, they obtain reduced local operation counts by as much as a factor 7.9 and speedups for the assembly of the global sparse matrix by as much as a factor 2.8.

3.3 Finite element form compilers

The FEniCS Form Compiler (FEC) was introduced in [27] and further developed in [28]. Analysa [2] also included a form compiler.

4 Quantum chemistry

See [1, 4, 5, 41] Also [6, 10, 11, 12, 16, 19, 34, 35, 36, 42]

5 Dense linear algebra

[7, 8, 9]

6 Signal processing

Signal processing algorithms have been studied extensively in the Spiral project [40].

7 Distributed and parallel computing

[13, 14]

8 Program analysis and transformation

There are automatic tools that extract information from existing codes. Two areas are performance analysis and sensitivity analysis (a.k.a., differentiation).

8.1 Automating performance analysis

The paper [20] presents a framework for parallel performance data mining and knowledge discovery. The PerfExplorer framework is part of the authors' ongoing research into automatic parallel performance analysis. PerfExplorer adresses the need to manage large-scale data complexity using techniques such as clustering and dimensionality reduction, and the need to perform automated discovery of relevant data relationships using comparitive and correlation analysis techniques. The intended uses of the framework include, but are not limited to, benchmarking, procurement evaluation, modeling, prediction and application optimization.

8.2 Automatic Differentiation

Algorithmic, or automatic, differentiation (AD) is concerned with the accurate and efficient evaluation of derivatives for functions defined by computer programs [17]. No truncation errors are incurred, and the resulting numerical derivative values can be used for all scientific computations that are based on linear, quadratic, or even higher order approximations to nonlinear scalar or vector functions. In particular, AD has been applied to optimization, parameter identification, equation solving, the numerical integration of differential equations, and combinations thereof. Apart from quantifying sensitivities numerically, AD techniques can also provide structural information, e.g., sparsity pattern and generic rank of Jacobian matrices.

9 Mesh generation

All discretization methods (finite element, finite difference, finite volume, spectral element, boundary element) require a mesh. This is an industry unto itself, so we do not try to survey it extensively, but give a few examples.

Triangle [44] is a program for efficiently generating 2D triangulations and Voronoi diagrams. Features include user-specified constraints on angles and triangle areas, user-specified holes and concavities, and the economical use of exact arithmetic to improve robustness. The paper [44] discusses many of the key implementation decisions, including the choice of triangulation algorithms and datastructures, the steps taken to create and refine a mesh, a number of issuesthat arise in Ruppert's algorithm, and the use of exact arithmetic.

NETGEN [43] is an advancing front 2D/3D-Mesh generator based on abstract rules. The process of tetrahedral mesh generation is broken up into four steps: special point calculation, edge following, surface meshing, and volume mesh generation. Several techniques of mesh optimization are tested for quality.

The Bank-Holst adaptive meshing paradigm [3] is an efficient approach for parallel adaptive meshing of elliptic partial differential equations. It is designed to keep communication costs low and to take advantage of existing sequential adaptive software.

The 'isosurface stuffing' algorithm [43] fills an isosurface with a tetrahedral mesh whose dihedral angles are bounded. It is fast and robust as it generates tetrahedra from a small number of precomputed stencils It is the first algorithm that rigorously guarantees the suitability of tetrahedra for finite element methods in domains whose shapes are substantially more challenging than boxes. If the isosurface is a smooth 2-manifold with bounded curvature, and the tetrahedra are sufficiently small, then the boundary of the mesh is guaranteed to be a geometrically and topologically accurate approximation of the isosurface.

One issue is to be able to partition a given mesh for parallel calculation The paper [38] describes an efficient approach to partitioning unstructured meshes that occur naturally in the finite element and finite difference methods. The approach makes use of the underlying geometric structure of a given mesh and finds a provably good partition in random O(n) time. It applies to meshes in both two and three dimensions. The new method has applications in efficient sequential and parallel algorithms for large-scale problems in scientific computing. This is an overview paper written with emphasis on the algorithmic aspects of the approach.

Many detailed proofs can be found in companion papers.

Another issue is to maintain shape regularity as meshes are subdivided. The papers [39, 47] present efficient techniques for constructing simplicial meshes in which each simplex is small enough, according to an application specific error test, and the number of distinctly-shaped simplices in the mesh is small, depending only on the dimension of the problem. The techniques include the decomposition of simplices of a certain kind into smaller similar simplices, the refinement of a hierarchy of simplices to enforce an element size continuity condition, and the processing of such a refined hierarchy to produce a simplicial, tree-structured mesh.

10 Algebra of compiler optimization

In [33], the authors (according to their abstract) "use Kleene algebra with tests to verify a wide assortment of common compiler optimizations, including dead code elimination, common subexpression elimination, copy propagation, loop hoisting, induction variable elimination, instruction scheduling, algebraic simplification, loop unrolling, elimination of redundant instructions, array bounds check elimination, and introduction of sentinels. In each of these cases, we give a formal equational proof of the correctness of the optimizing transformation."

References

[1] Auer, A. A., Baumgartner, G., Bernholdt, D. E., Bibireata, A., Choppella, V., Cociorva, D., Gao, X., Harrison, R., Krishnamoorthy, S., Krishnan, S., Lam, C.-C., Lu, Q., Nooijen, M., Pitzer, R., Ramanujam, J., Sadayappan, P., and Sibiryakov, A. Automatic code generation for many-body electronic structure methods: the tensor contraction engine. *Molecular Physics 104* (2006), 211–228.

[2] Bagheri, B., and Scott, L. R. About Analysa. Research Report UC/CS TR-2004-09, Dept. Comp. Sci., Univ. Chicago, 2004.

[3] Bank, R. E. Some variants of the bank–holst parallel adaptive meshing paradigm. *Comput. Vis. Sci. 9*, 3 (2006), 133–144.

[4] Baumgartner, G., Auer, A., Bernholdt, D. E., Bibireata, A., Choppella, V., Cociorva, D., Gao, X., Harrison, R. J., Hirata, S., Krishanmoorthy, S., Krishnan, S., Lam, C.-C., Lu, Q., Nooijen, M., Pitzer, R. M., Ramanujam, J., Sadayappan, P., and Sibiryakov, A. Synthesis of high-performance parallel programs for a class of ab initio quantum chemistry models. *Proceedings of the IEEE 93*, 2 (2005). special issue on "Program Generation, Optimization, and Adaptation".

[5] Bernholdt, D., Nieplocha, J., and Sadayappan, P. Raising the Level of Programming Abstraction in Scalable Programming Models. *IEEE International Conference on High Performance Computer Architecture (HPCA), Workshop on Productivity and Performance in High-End Computing (P-PHEC)*, 76–84.

[6] Bibireata, A., Krishnan, S., Baumgartner, G., Cociorva, D., Lam, C., Sadayappan, P., Ramanujam, J., Bernholdt, D., and Choppella, V. Memory-constrained data locality optimization for tensor contractions. *Lecture notes in computer science Volume 2958: Languages and Compilers for Parallel Computing*, 93–108.

[7] Bientinesi, P., Dhillon, I. S., and van de Geijn, R. A. A parallel eigensolver for dense symmetric matrices based on multiple relatively robust representations. *SIAM J. Sci. Comput. 27*, 1 (2005), 43–66.

[8] Bientinesi, P., Gunnels, J. A., Myers, M. E., Quintana-Ortí, E. S., and van de Geijn, R. A. The science of deriving dense linear algebra algorithms. *ACM Trans. Math. Softw. 31*, 1 (2005), 1–26.

[9] Bientinesi, P., Quintana-Ortí, E. S., and van de Geijn, R. A. Representing linear algebra algorithms in code: the flame application program interfaces. *ACM Trans. Math. Softw. 31*, 1 (2005), 27–59.

[10] Cociorva, D., Baumgartner, G., Lam, C.-C., Sadayappan, P., Ramanujam, J., Nooijen, M., Bernholdt, D. E., and Harrison, R. Space-time trade-off optimization for a class of electronic structure calculations. In *PLDI '02: Proceedings of the ACM SIGPLAN 2002 Conference on Programming language design and implementation* (New York, NY, USA, 2002), ACM Press, pp. 177–186.

[11] Cociorva, D., Wilkins, J., Baumgartner, G., Sadayappan, P., Ramanujam, J., Nooijen, M., Bernholdt, D., and Harrison, R. Towards automatic synthesis of high-performance codes for electronic structure calculations: Data locality optimization. *Lecture Notes in Computer Science 2228* (2001), 237–??

[12] Cociorva, D., Wilkins, J. W., Lam, C.-C., Baumgartner, G., Ramanujam, J., and Sadayappan, P. Loop optimization for a class of memory-constrained computations. In *ICS '01: Proceedings of the 15th international conference on Supercomputing* (New York, NY, USA, 2001), ACM Press, pp. 103–113.

[13] de Carvalho, F. H., and Lins, R. D. The # model: separation of concerns for reconciling modularity, abstraction and efficiency in distributed parallel programming. In *SAC '05: Proceedings of the 2005 ACM symposium on Applied computing* (New York, NY, USA, 2005), ACM Press, pp. 1357–1364.

[14] de Carvalho Junior, F., Lins, R., CorrÃ?a, R., AraÃ?jo, G., and de Santiago, C. 2007, ch. Design and Implementation of an Environment for Component-Based Parallel Programming, pp. 184–197. 10.1007/978-3-540-71351-7_15.

[15] Fraser, C. W. *Automatic generation of code generators.* PhD thesis, 1977.

[16] Gao, X., Sahoo, S. K., Lam, C.-C., Ramanujam, J., Lu, Q., Baumgartner, G., and Sadayappan, P. Performance modeling and optimization of parallel out-of-core tensor contractions. In *PPoPP '05: Proceedings of the tenth ACM SIGPLAN symposium on Principles and practice of parallel programming* (New York, NY, USA, 2005), ACM Press, pp. 266–276.

[17] Griewank, A. *Evaluating derivatives: principles and techniques of algorithmic differentiation.* Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2000.

[18] Guyer, S. Z., and Lin, C. Broadway: a compiler for exploiting the domain-specific semantics of software libraries. *Proceedings of the IEEE 93* (2005), 342–357.

[19] Hirata, S. Tensor Contraction Engine: Abstraction and Automated Parallel Implementation of Configuration-Interaction, Coupled-Cluster, and Many-Body Perturbation Theories. *J. Phys. Chem. A 107*, 46 (2003), 9887–9897.

[20] Huck, K. A., and Malony, A. D. Perfexplorer: A performance data mining framework for large-scale parallel computing. *sc 0* (2005), 41.

[21] Karuri, K. *A Framework for Automatic Generation of Code Optimizers.* PhD thesis, 2001.

[22] Kennedy, K., Broom, B., Chauhan, A., Fowler, R., Garvin, J., Koelbel, C., McCosh, C., and Mellor-Crummey, J. Telescoping languages: A system for automatic generation of domain languages. *Proceedings of the IEEE 93*, 2 (2005). special issue on "Program Generation, Optimization, and Adaptation".

[23] Kirby, R., and Logg, A. Benchmarking domain-specific compiler optimizations for variational forms. Preprint from The Finite Element Center, April 2007.

[24] Kirby, R. C. Algorithm 839: Fiat, a new paradigm for computing finite element basis functions. *ACM Trans. Math. Softw. 30*, 4 (2004), 502–516.

[25] Kirby, R. C. Optimizing fiat with level 3 blas. *ACM Trans. Math. Softw. 32*, 2 (2006), 223–235.

[26] Kirby, R. C., Knepley, M., Logg, A., and Scott, L. R. Optimizing the evaluation of finite element matrices. *SIAM J. Sci. Computing 27* (2005), 741–758.

[27] Kirby, R. C., and Logg, A. A compiler for variational forms. *ACM Trans. Math. Softw. 32*, 3 (2006), 417–444.

[28] Kirby, R. C., and Logg, A. Efficient compilation of a class of variational forms. *ACM Trans. Math. Softw. 33*, 3 (2007), 17.

[29] Kirby, R. C., Logg, A., Scott, L. R., and Terrel, A. R. Topological optimization of the evaluation of finite element matrices. *SIAM J. Sci. Computing 28* (2006), 224–240.

[30] Kirby, R. C., and Scott, L. R. Geometric optimization of the evaluation of finite element matrices. *SIAM J. Sci. Computing 29* (2007), 827–841.

[31] Korelc, J. Automatic generation of finite-element code by simultaneous optimization of expressions. *Theoretical Computer Science 187* (Nov 1997), 231–248.

[32] Korelc, J. Multi-language and multi-environment generation of nonlinear finite element codes. *Engineering with Computers 18* (Nov 2002), 312–327. 10.1007/s003660200028.

[33] Kozen, D., and Patron, M.-C. Certification of compiler optimizations using kleene algebra with tests. In *CL '00: Proceedings of the First International Conference on Computational Logic* (London, UK, 2000), Springer-Verlag, pp. 568–582.

[34] Krishnan, S., Krishnamoorthy, S., Baumgartner, G., Cociorva, D., Lam, C., Sadayappan, P., Ramanujam, J., Bernholdt, D., and Choppella, V. Data Locality Optimization for Synthesis of Efficient Out-of-Core Algorithms. *Proc. of the Intl. Conf. on High Performance Computing* (2003).

[35] Lam, C.-C., Cociorva, D., Baumgartner, G., and Sadayappan, P. Memory-optimal evaluation of expression trees involving large objects. In *HiPC* (1999), pp. 103–110.

[36] Lam, C.-C., Sadayappan, P., and Wenger, R. On optimizing a class of multi-dimensional loops with reductions for parallel execution. *Parallel Processing Letters 7*, 2 (1997), 157–168.

[37] Mardal, K. A., Skavhaug, O., Lines, G., Staff, G., and Odegard, A. Using python to solve partial differential equations. *Computing in Science and Engineering* (2007).

[38] Miller, G. L., Teng, S., Thurston, W., and Vavasis, S. A. Automatic mesh partitioning. Tech. rep., Ithaca, NY, USA, 1992.

[39] Moore, D., and Warren, J. Adaptive simplicial mesh quadtrees. *Houston J. Math 21*, 3 (1995), 525–540.

[40] Püschel, M., Moura, J. M. F., Johnson, J., Padua, D., Veloso, M., Singer, B. W., Xiong, J., Franchetti, F., Gačić, A., Voronenko, Y., Chen, K., Johnson, R. W., and Rizzolo, N. SPIRAL: Code generation for DSP transforms. *Proceedings of the IEEE 93*, 2 (2005). special issue on "Program Generation, Optimization, and Adaptation".

[41] Sahoo, S. K., Krishnamoorthy, S., Panuganti, R., and Sadayappan, P. Integrated loop optimizations for data locality enhancement of tensor contraction expressions. In *Supercomputing, 2005. Proceedings of the ACM/IEEE SC 2005 Conference* (2005), pp. 13–13.

[42] Sahoo, S. K., Krishnamoorthy, S., Panuganti, R., and Sadayappan, P. Integrated loop optimizations for data locality enhancement of tensor contraction expressions. In *SC '05: Proceedings of the 2005 ACM/IEEE conference on Supercomputing* (Washington, DC, USA, 2005), IEEE Computer Society, p. 13.

[43] Schöberl, J. Netgen: An advancing front 2d/3d-mesh generator based on abstract rules. *Computing and Visualization in Science* (1997).

[44] Shewchuk, J. R. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In *Applied Computational Geometry: Towards Geometric Engineeri= ng*, M. C. Lin and D. Manocha, Eds., vol. 1148 of *Lecture Notes in Computer Science*. Springer-Verlag, May 1996, pp. 203–222. From the First ACM Workshop on Applied Computational Geometry.

[45] van Engelen, R., Wolters, L., and Cats, G. CTADEL: a generator of multi-platform high performance codes for PDE-based scientific applications. In *ICS '96: Proceedings of the 10th international conference on Supercomputing* (New York, NY, USA, 1996), ACM Press, pp. 86–93.

[46] Wang, P. S., Tan, H.-Q., Saleeb, A. F., and Chang, T.-Y. P. Code generation for hybrid mixed mode formulation in finite element analysis. In *SYMSAC '86: Proceedings of the fifth ACM symposium on Symbolic and algebraic computation* (New York, NY, USA, 1986), ACM Press, pp. 45–52.

[47] Zhang, S. Successive subdivisions of tetrahedra and multigrid methods on tetrahedralmeshes. *Houston J. Math 21*, 3 (1995), 541–556.