From High-Level Specification to High-Performance Code

By FRANZ FRANCHETTI, ***Senior Member IEEE***
Guest Editor

JOSÉ M. F. MOURA, ***Fellow IEEE***
Guest Editor

DAVID A. PADUA, ***Fellow IEEE***
Guest Editor

JACK DONGARRA, ***Fellow IEEE***
Guest Editor

I. INTRODUCTION

Computer architectures and systems are becoming ever more powerful but increasingly more complex. With the end of frequency scaling (about 2004) and the era of multicores/manycores/accelerators, it is exceedingly hard to extract the promised performance, in particular, at a reasonable energy budget. Only highly trained and educated experts can hope to conquer this barrier that, if not appropriately dealt with, can translate into multiple orders of magnitude of underutilization of computer systems when programmed by less specialized programmers or domain scientists. To overcome this challenge, the last ten years have seen a flurry of activity to automate the design and generation of highly efficient implementations for these multicore/manycore architectures, and to translate high level descriptions of programs into high performance and power efficiency.

This month's special issue brings together leading approaches for developing high-performance code which is required today for delivering high performance and power efficiency due to the complex nature of computer architectures and systems.

This special issue brings together the leading approaches to tackle this remarkably hard problem. We feature articles on program translation, software synthesis, and automatic performance tuning in the context of compilers and library generators, performance engineering, program generation, domain-specific languages, and hardware synthesis for multicore/manycore architectures. We invited authors from leading world institutions (academic, industry, and national laboratories,) including from the United States, Europe, and Japan. We cover the range of machines from small scale level to the largest supercomputer installations and show that the underlying problems and techniques are remarkably similar.

II. COMPILATION

Computer architects are experimenting with ever more complex systems containing manycore processors, graphics processors, field-programmable gate arrays, and a high level of speculation techniques to keep Moore's law on track and to keep the systems within their power envelope. This enormous growth of computing power is a boon to scientists; however, it comes at a high cost: development of computing applications has become a nightmare. For today's commodity computers, even computer science and computer engineering graduate students produce suboptimal programs that may underperform peak performance by two orders of magnitude on a single central processing unit (CPU) and will not take advantage of multiple CPUs or accelerators. This section discusses compilation approaches that enable

Digital Object Identifier 10.1109/JPROC.2018.2875253

performance portability across current machines.

The paper "Machine learning in compiler optimization" by Wang and O'Boyle discusses the impact of machine-learning-based compilation having moved from an obscure research niche to a mainstream activity. The article describes the relationship between machine learning and compiler optimization and introduces the main concepts of features, models, training, and deployment. It provides a comprehensive survey and provides a road map for the wide variety of different research areas and concludes with a discussion on open issues in the area and potential research directions.

The paper "Domain-specific optimization and generation of high-performance GPU code for stencil computations" by Rawat *et al.* discusses stencil computations, which arise in a number of computational domains and exhibit significant data parallelism. They are well suited for execution on graphical processing units (GPUs), but can be memory-bandwidth limited unless temporal locality is utilized via tiling. The paper describes how effective tiled code can be generated for GPUs from a domain-specific language for stencils. Experimental results demonstrate the benefits of such a domain-specific optimization approach over state-of-the-art general-purpose compiler optimization.

The paper "The sparse polyhedral framework: Composing compiler-generated inspector–executor code" by Strout *et al.* discusses targeting irregular applications such as big graph analysis, material simulations, molecular dynamics simulations, and finite element analysis that have performance problems due to their use of sparse data structures. This paper reviews the history and current state of the art for inspector–executor strategies and reviews how the sparse polyhedral framework (SPF) enables the composition of inspector–executor transformations. Furthermore it describes a research vision to combine this generality in SPF with specialization to achieve composable and high-performance inspectors and executors, producing a powerful compiler framework for sparse matrix computations.

III. PROGRAM SYNTHESIS AND PROGRAM GENERATION

This section discusses program synthesis and program generation methods for a number of application domains to overcome the challenges posed by modern hardware.

The paper "SPIRAL: Extreme performance portability" by Franchetti *et al.* addresses the question of how to automatically map computational kernels to highly efficient code for a wide range of computing platforms and establishes the correctness of the synthesized code. More specifically, it focuses on performance portability across the ever-changing landscape of parallel platforms. The paper discusses the approach implemented in the SPIRAL system and demonstrates it with a selection of computational kernels from the signal and image processing domain, software-defined radio, and robotic vehicle control for target platforms ranging from mobile devices, desktops, and server multicore processors to large-scale high-performance and supercomputing systems.

The paper "Automating the development of high-performance multigrid solvers" by Schmitt *et al.* discusses domain-specific program generation, which has lately received increasing attention in the area of computational science and engineering. It introduces the new code generation framework Athariac. Its goal is to support the quick implementation of a language processing and program optimization platform for a given domain-specific language (DSL) based on stepwise term rewriting. The paper demonstrates the framework's use on the DSL ExaSlang for the specification and optimization of multigrid solvers. The paper provides evidence of Athariac's potential for making domain-specific software engineering more productive.

IV. MULTICORE, MANYCORE, AND EXASCALE

Programming accelerators and multicores are challenging. There is a wealth of emerging languages and programming models. A number of these have established themselves as *de facto* standards. This section provides an overview on the state of the art of accelerator and multicore programming and sketches the future direction to be expected.

The paper "The long and winding road toward efficient high-performance computing" by Jalby *et al.* discusses a major challenge to Exaflop computing, and, more generally, efficient high-end computing: finding the best "matches" between advanced hardware capabilities and the software used to program applications, so that top performance will be achieved. It is important that key performance enablers at the software level—autotuning, performance analysis tools, full application optimization—be understood. For each area the paper highlights major limitations and most promising approaches to reaching better performance and energy levels. It concludes by analyzing hardware and software design, trying to pave the way for more tightly integrated hardware and software codesign.

The paper "The ongoing evolution of OpenMP" by de Supinski *et al.* presents an overview of the past, present, and future of the OpenMP application programming interface (API). While the API originally specified a small set of directives that guided shared memory fork-join parallelization of loops and program sections, OpenMP now provides a richer set of directives that capture a wide range of parallelization strategies that are not strictly limited to shared memory. The paper argues that future OpenMP versions will support a range of parallelization strategies and the addition of direct support for debugging and performance analysis

tools. Furthermore, the paper predicts that future OpenMP specifications will likely include support for more parallelization strategies and to embrace closer integration into its Fortran, C, and, in particular, C++ base languages.

The paper "Navigating the landscape for real-time localization and mapping for robotics and virtual and augmented reality" by Saeedi *et al.* discusses the challenges of visual understanding of 3-D environments in real time, at low power. It describes the results of a major research effort to assemble the algorithms, architectures, tools, and systems software needed to enable delivery of simultaneous localization and mapping (SLAM), by supporting applications specialists in selecting and configuring the appropriate algorithm and the appropriate hardware, and compilation pathway, to meet their performance, accuracy, and energy consumption goals.

V. AUTOTUNING AND EXASCALE

Automatic performance tuning (autotuning) is an established technique to automate the tedious task of extracting performance from complicated architectures. Applying these techniques to large-scale computer systems (petascale and exascale) comes with its own set of challenges. This section gives an overview of the necessary techniques.

The paper "Autotuning numerical dense linear algebra for batched computation with GPU hardware accelerators" by Dongarra *et al.* addresses computational problems in engineering and scientific disciplines that rely on the solution of many instances of small systems of linear equations. It targets graphics processing units (GPUs), which over the past decade have become an attractive high-performance computing (HPC) target for solvers of linear systems of equations. The paper shows how an automated exploration of the implementation space as well as new data layouts lead to large performance gains over the state of the art.

The paper "Japanese autotuning research: Autotuning languages and FFT" by Katagiri and Takahashi introduces the current research on automatic performance tuning by the Japanese research community and focuses on two main research thrusts. First, it introduces the Japanese autotuning frameworks, including methodology, computer languages, and performance models. In addition, it discusses target algorithms and applications. The second focus is on the fast Fourier transform (FFT) that has been extensively targeted by autotuning techniques. The paper explains the state-of-the-art algorithms for advanced multicore architectures, with focus on Japanese target machines.

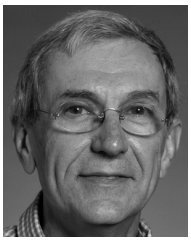
The paper "Autotuning in high-performance computing applications" by Balaprakash *et al.* draws on the authors' extensive experience applying autotuning to high-performance applications, describing both successes and future challenges. It argues that if autotuning is to be widely used in the HPC community, then researchers must address the software engineering challenges, manage configuration overheads, and continue to demonstrate significant performance gains and portability across architectures. The paper proposes that tools that configure the application must be integrated into the application building process so that tuning can be reapplied as the application and target architectures evolve.

ABOUT THE GUEST EDITORS

Franz Franchetti (Senior Member, IEEE) received the Dipl.-Ing. (M.Sc.) degree in technical mathematics and the Dr. techn. (Ph.D.) degree in computational mathematics from the Vienna University of Technology, Vienna, Austria, in 2000 and 2003, respectively.

Currently, he is a Professor at the Department of Electrical and Computer Engineering, Carnegie Mellon University, Pittsburgh, PA, USA. His research focuses on automatic performance tuning and program generation for emerging parallel platforms and algorithm/hardware cosynthesis. He targets multicore CPUs, clusters and high-performance systems (HPC), graphics processors (GPUs), field-programmable gate arrays (FPGAs), FPGA acceleration for CPUs, and logic-in-memory and 3DIC chip design. Within the SPIRAL effort, his research goal is to enable automatic generation of highly optimized software libraries for important kernel functionality. In other collaborative research threads, he is investigating the applicability of domain-specific transformations within standard compilers and the application of HPC in material sciences. He led three DARPA projects in the BRASS, HACMS, and PERFECT programs and is a Principal Investigator/Principal Co-Investigator (PI/Co-PI) on a number of federal and industry grants.

Dr. Franchetti was a member of the team winning the Gordon Bell Prize (Peak Performance Award) in 2006, and he was a member of the team winning the HPC Challenge Class II Award (most productive system) in 2010. In 2013, he was awarded the CIT Dean's Early Career Fellowship by the College of Engineering of Carnegie Mellon University.



José M. F. Moura (Fellow, IEEE) received the Electrical Engineering degree from the Instituto Superior Técnico (IST), Lisboa, Portugal and the Electrical Engineering and M.Sc. degrees and the D.Sc. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA.

Currently, he is the Philip L. and Marsha Dowd University Professor at Carnegie Mellon University, Pittsburgh, PA, USA, with the Departments of Electrical and Computer Engineering and, by courtesy, of Biomedical Engineering. He has held visiting Professor appointments at MIT and New York University and the Center for Urban Science & Progress (CUSP), and he was on the faculty of IST. His research interests are in statistical signal image, and data processing, and data science, with particular emphasis on distributed decision and inference in networked systems and graph based data.

Dr. Moura is a member of the U.S. National Academy of Engineers, a Fellow of the U.S. National Academy of Inventors, a corresponding member of the Portugal Academy of Science, and a Fellow of the American Association for the Advancement of Science (AAAS). He has received several awards including the IEEE Signal Processing Society Award for outstanding technical contributions and leadership in signal processing and the IEEE Signal Processing Society Technical Achievement Award for fundamental contributions to statistical signal processing. He has served in several volunteering capacities the IEEE, being currently the 2018 President Elect of IEEE, to serve as IEEE President in 2019.

David A. Padua (Fellow, IEEE) is a Donald Biggar Willet Professor in Engineering at the University of Illinois at Urbana-Champaign, Urbana, IL, USA, where he has been a Faculty Member since 1985. His has done research in parallel computing, autotuning, and compilers and has published more than 170 papers in these areas. He has participated in the organization of more than 70 conferences and workshops and served on the editorial board of the *ACM Transactions on Programming Languages and Systems* (ACM TOPLAS), *Journal of Parallel and Distributed Computing* (JPDC), the IEEE Transactions on Parallel and Distributed Systems (IEEE TPDS), and the *International Journal of Parallel Programming* (IJPP), and as Editor-in-Chief of the *Encyclopedia of Parallel Computing* (Springer-Verlag).

Prof. Padua is a Fellow of the Association for Computing Machinery (ACM) and the recipient of the 2015 IEEE Computer Society Harry H. Goode Award. In 2017, he received a doctorate honoris causa from the University of Valladolid, Spain.

Jack Dongarra (Fellow, IEEE) holds an appointment as the University Distinguished Professor of Computer Science in the Electrical Engineering and Computer Science Department at the University of Tennessee, Knoxville, TN, USA and holds the title of Distinguished Research Staff in the Computer Science and Mathematics Division at Oak Ridge National Laboratory (ORNL), Oak Ridge, TN, USA; Turing Fellow at Manchester University, Manchester, U.K.; and an Adjunct Professor in the Computer Science Department at Rice University, Houston, TX, USA. He specializes in numerical algorithms in linear algebra, parallel computing, the use of advanced-computer architectures, programming methodology, and tools for parallel computers. His research includes the development, testing and documentation of high-quality mathematical software. He has contributed to the design and implementation of the following open source software packages and systems: EISPACK, LINPACK, the BLAS, LAPACK, ScaLAPACK, Netlib, PVM, MPI, NetSolve, Top500, ATLAS, and PAPI. He has published approximately 200 articles, papers, reports and technical memoranda and he is coauthor of several books.

Prof. Dongarra was awarded the IEEE Sid Fernbach Award in 2004 for his contributions in the application of high-performance computers using innovative approaches; in 2008, he was the recipient of the first IEEE Medal of Excellence in Scalable Computing; in 2010, he was the first recipient of the SIAM Special Interest Group on Supercomputing's award for Career Achievement; in 2011, he was the recipient of the IEEE Charles Babbage Award; in 2013, he was the recipient of the ACM/IEEE Ken Kennedy Award for his leadership in designing and promoting standards for mathematical software used to solve numerical problems common to high-performance computing; and in 2019, he received the SIAM/ACM Prize in Computation Science and Engineering. He is a Fellow of the American Association for the Advancement of Science (AAAS), the Association for Computing Machinery (ACM), and the Society for Industrial and Applied Mathematics (SIAM), and a foreign member of the Russian Academy of Sciences and a member of the U.S. National Academy of Engineering.