Statement of Research

Paolo Bientinesi

My research interests center around the fields of scientific and high-performance computing, with emphasis on automation. Research in these fields is motivated by real-world scientific problems and deals with the development of fast and accurate numerical algorithms. Despite the inherent interdisciplinary nature of the research, algorithms have traditionally been sought in abstraction of both the scientific application originating the input data and the executing architecture. Abstraction here translates to separation of concerns, which is a limiting factor in the pursuit of superior algorithms. This is especially true today, with the push towards innovative high-performance architectures such as multi-cores processors, GPU, FPGAs and Cell Broadband Engine. I believe that nowadays, combining and exploiting knowledge from applications, algorithms and architectures, is not only possible, but critical, to the progress of scientific and high-performance computing. Such a synergistic and interdisciplinary approach is an integral component of my research efforts.

My long term research plan is twofold. On one hand I intend to lead a team of scientists and students to study algorithms not in isolation, but in the broader context of applications and architectures. I am fully convinced that encoding extra information and knowledge is the route towards higher-quality algorithms. In this context, quality shall not be measured strictly in terms of performance and accuracy; algorithms shall be evaluated by how they enable new science and by their impact on scientific applications. High-performance computations are to become the common language in this new research paradigm; crucial to success will be the identification of abstractions and tools to allow for both productive and efficient algorithm & software development.

The second goal in my research agenda is automation[1]. Beyond seeking efficient solutions to instances of a problem, I aim at uncovering the mechanisms leading to such solutions. These mechanisms, once exposed, make automatic computations possible. In my mind, a mathematical problem can be labeled as closed only when it is so well understood that any instance could be solved by an automated system with limited, if any, human intervention. In the years to come, I intend to pursue an even more ambitious research goal, stemming from the following provocative statement: in the context of scientific computations, a problem is closed only when it is so well understood that its properties can be automatically identified and exploited in the solution process. Advances in this research line would greatly impact both the fields of scientific and high-performance computing, and large-scale computations in particular.

In my career, I have initiated and contributed to research projects in many different areas of computer science. Specifically, the focus has been on theoretical and practical aspects of the following subjects: Digital Signal Processing [1,14], Sparse Linear Algebra [2], Automation [6,10], High-Performance Computing [3,4,5], Numerical Linear Algebra [11,9], Symbolic Computations [6,8], Parallel Computing [1,3,4,5,11], Software Engineering [12,13], Computational Geometry [15], and Formal Methods [7,8,6,9]. In the next section, I overview a

[1]The word "automation" is sometimes used as a synonym for "fine tuning"; in contrast, in my research, the word automation indicates the use of computers for replicating complex tasks with limited or no human intervention. To avoid confusion, in my publications I often adopted the word "mechanical" as an alternative to "automatic".

number of projects and research ideas, emphasizing their relevance and contributions towards my long term research plan. In my web page, `http://www.cs.utexas.edu/users/pauldj`, I showcase results from the most significant projects that I have completed.

Research Directions

Multi-core Architectures & Performance Prediction [4,5]. Multi-core processors have become commodity hardware; they are used in a variety of computer systems, ranging from large computing clusters to home desktops and even laptops. Consequently, these parallel processors represent the engines for tackling massively parallel simulations with millions of degrees of freedom as well as problems of modest size to be executed on a single desktop. Hence, it is obvious that linear algebra libraries need to be tuned not only for asymptoticly large matrices, but across the entire spectrum of problem sizes. The question is how to exploit the parallelism offered by multi-core architectures. My goal is to provide an answer for computations that involve dense linear algebra problems and/or eigensolvers.

As part of the FLAME team, we have tackled the problem, relative to linear algebra operations, following two different paradigms: 1) parallelism is obtained through high-performance multi-threaded BLAS; 2) parallelism comes from data blocking and careful task scheduling. In the first case, sequential algorithms are directly translated into parallel code by linking to multi-threaded BLAS. High-performance is a byproduct of the richness, in terms of algorithms, of the family generated by my automated system or by the systematic procedure. In the latter case, the computation is partitioned down to blocks of size for which the sequential BLAS is optimized. A graph of dependencies is created among the blocks, and clever scheduling of the blocks is the cause of high-performance. We have gathered evidence that for different problem sizes and under different conditions, both the approaches have merit.

The research direction that I intend to pursue is related to the capability of accurately model and predict the performance of an algorithm or a scheduling scheme. Crucial to the success of this project is the understanding of the performance signature of the BLAS library; to this end, I am collaborating with Kasushige Goto, the developer of the popular GotoBLAS. Preliminary investigation, performed with the aid of a prototype simulator that I developed, showed predictions that closely mirror the actual algorithmic performance. I believe that the same approach will yield highly accurate estimates even for architectures with complex multi-level memories.

Automatic Derivation of Algorithms [6,3,7,8,10]. High performance libraries for dense linear algebra operations are one of the fundamental building blocks in the area of scientific computing. Even applications that yield sparse linear systems often require the manipulation of smaller dense subproblems. As part of the Formal Linear Algebra Method Environment (FLAME) project, conducted at The University of Texas at Austin, we set out not to build a linear algebra library, but to build systematic and automated tools to build libraries.

The first milestone towards automation was the discovery of a systematic procedure for generating loop-based algorithms. We realized that, from the mathematical description of a linear algebra operation belonging to a certain class, it is possible to identify –*a priori*– one or more loop invariants. These are loop invariants for loop-based algorithms that compute the specified operation. Given that a loop invariant fully dictates the structure of a loop, we were able to lay out a sequence of algebraic manipulations to transform the mathematical description of an input operation into provably correct loop-based algorithms.

This generating procedure can be applied multiple times to the same operation, giving rise to a family of algorithms, each of which with its own performance and accuracy signature. The scope of applicability of the procedure includes, but is not limited to, all the Basic Linear Algebra Subprograms (BLAS) operations, all the operation supported by RECSY, a library for control theory, and many of the operations supported by the Linear Algebra Package (LAPACK).

In my Ph.D. dissertation, in addition to this systematic procedure, I showcased an automated system that, given a formal description of a target linear algebra operation, with only limited human intervention, returns algorithms and code for the operation. Thanks to this system, a user is able to generate dozens of algorithms and routines, even for advanced operations like the coupled (generalized) triangular Sylvester equation, in matter of minutes. As part of a collaboration with Prof. Brian Gunter, I used the system to generate algorithms and code for inverting a symmetric positive definite matrix, operation needed for the accurate estimation of the gravitational Earth field; the same operation is used in tomography. We measured a performance improvement of 20% to 30% with respect to the existing libraries, allowing a tangible advancement in the scientific investigation.

I want to stress the role of automation in the generation of algorithms. One interpretation is that an automated system assists the user in performing extremely complicated symbolic manipulations in a timely manner. While this statement is correct, it is incomplete. My system also generates results (in this case algorithms) that may have not been known in advance. In fact, in many occurrences of our research, the system has found more algorithms than we would have, had we tackled the problem by hand. In summary, this automated system is a tool assisting the user in making new discoveries.

The ultimate objective of this research effort is an automated system that, instead of returning a family of algorithms, returns the best algorithm, in terms of a given metric, for a target architecture and target settings. As I mentioned in the former section, this is one problem that I plan to investigate further. An orthogonal question is whether it is possible to single out, within a family of algorithms, all the ones that are numerically stable. The last chapter of my dissertation and [9] shows preliminary results in this direction.

Sparse Direct Solvers for Unassembled Hyper-Matrices [2]. The solution of the linear system $Ax = b$, with sparse matrix A, is undoubtedly the most common operation among scientific computations. The purpose of this project is to efficiently compute the solution of a sequence of linear systems $A_i x_i = b_i$, where the matrix A_i is tightly related to A_{i-1}.

In the setting of *hp*-adaptive Finite Element Methods, the sequence of matrices $\{A_i\}$ results from successive local refinements of the problem domain. The process of local refinements is recursive —domains are partitioned into subdomains— therefore the interactions among domains are naturally captured by an "HyperMatrix" (a matrix whose entries are matrices). Since the domain refinements are only local, modifications to one subdomain affect only a small, and known, number of entries in the HyperMatrix. Thus, the matrices A_i and A_{i+1} are strongly correlated; also to be noted, at any step $i > 1$, a factorization of a matrix A_{i-1} is available for reuse.

Given this formulation of the problem, it is not the solution of one instance of the linear system $Ax = b$ that should be optimized, as traditional solvers do, but the solution of a sequence of such systems. Standard solvers do not have a way of exploiting the factorization available from the former iteration, nor they accept hypermatrices as input. Since they only operate with flat matrices, any information relative to the hierarchy is lost, and graph partitioning algorithms are needed in the attempt to reconstruct the information in approximate form.

As part of a collaboration with the Texas Advanced Computing Center (TACC) and the Institute for Computational Engineering and Sciences (ICES), we have designed a new hierarchical data structure, the Unassembled Hyper-Matrix (UHM), and a solver, which take advantage of both the properties of the problem and the knowledge available in the application. In detail, we allow the matrix to be stored as a tree of unassembled element matrices, hierarchically ordered to mirror the refinement history of the physical domain. The factorization of an UHM assembles the nodes only when they need to be eliminated and makes use of the previously computed factorization.

This project, that received NSF funding, is still in its infancy. My first objective is to produce an hierarchical Cholesky solver. I believe that this will result in both space and performance improvements with respect to traditional solvers. Extensions to non-positive definite matrices (pivoting) and parallel architectures (multi-core) are research directions that I intend to explore in the future.

Multidimensional Fast Fourier Transforms on the Cell Processor [1]. The Cell Broadband Engine is a multi-core processor with low-power consumption and high computing performance; it was designed targeting gaming and multimedia applications on small electronic devices. The Cell is composed of one master processor (PPE), up to eight synergistic processing elements (SPEs) and a broadband interconnect bus. Because of its impressive theoretical peak performance (204.8 `Gflops` when operating in single precision floating point arithmetic), its release generated a general interest in trying to exploit such a computational power for other applications.

In collaboration with Prof. Nikos Pitsianis and Prof. Xiaobai Sun at Duke University, wet set out to evaluate the potential of the Cell when employed in real-time, low-power, high-resolution image processing systems, such as synthetic aperture radar imaging systems, tomographic imaging, and interactive volume data rendering. Our first target was the efficient computation of 2D FFTs on streaming data. After a first period of investigation of the architecture, in which we studied the cost of memory accesses, data movements and arithmetic vector operations, we designed an algorithm that would benefit from the features of the Cell.

Our algorithm for 2D FFTs consists of two phases: a coupling phase in which pair-wise communications are interwoven with arithmetic operations, and an uncoupled phase, during which smaller 2D FFTs are computed independently on the SPEs, leaving the communication network free to transfer data to and from the main memory for the stream of successive FFTs. Thanks to the joint exploitation of the architectural features specific to the Cell and the algorithmic properties specific to the FFT, the algorithm achieves almost perfect overlap of computation and communication; when compared with other codes, including the acclaimed FFTW, our algorithm attains the best performance.

Having built an accurate computational model for the Cell processor, my goal has now shifted towards the efficient computation of three and higher dimensional FFTs.

Selected References

[1] Paolo Bientinesi, Nikos Pitsianis, Xiaobai Sun. *Parallel 2D FFTs on the Cell Broadband Engine.* Submitted to International Journal of High Performance Computing Applications.

[2] Paolo Bientinesi, Victor Eijkhout, Kyungjoo Kim, Jason Kurtz and Robert van de Geijn. *Sparse Direct Factorizations through Unassembled Hyper-Matrices.* Submitted to Computer Methods in Applied Mechanics and Engineering.

[3] Paolo Bientinesi, Brian Gunter and Robert van de Geijn. *Families of Algorithms Related to the Inversion of a Symmetric Positive Definite Matrix.* Accepted for publication in ACM Transactions on Mathematical Software.

[4] Paolo Bientinesi, Ernie Chan, Enrique Quintana-Ortí, Gregorio Quintana-Ortí, Robert van de Geijn and Field Van Zee. *SuperMatrix: a Multithreaded Runtime Scheduling System for Algorithms-by-Blocks.* ACM SIGPLAN 2008 Symposium on Principles and Practice of Parallel Programming (PPoPP'08), February 20-23, 2008.

[5] Paolo Bientinesi, Tze Meng Low, Robert van de Geijn and Field van Zee. *Scalable Parallelization of FLAME Code via the Workqueuing Model.* Accepted for publication in ACM Transactions on Mathematical Software.

[6] Paolo Bientinesi. *Mechanical Derivation and Systematic Analysis of Correct Linear Algebra Algorithms.* The University of Texas at Austin, Department of Computer Sciences. September 2006. (Ph.D. Dissertation). Also Technical Report TR-06-46.

[7] Paolo Bientinesi, John A. Gunnels, Margaret E. Myers, Enrique S. Quintana-Ortí and Robert A. van de Geijn. *The Science of Deriving Dense Linear Algebra Algorithms.* ACM Transactions on Mathematical Software, 31(1), March 2005.

[8] Paolo Bientinesi and Robert van de Geijn, *Formal Correctness and Stability of Dense Linear Algebra Algorithms.*, 17th IMACS World Congress: Scientific Computation, Applied Mathematics and Simulation, 2005.

[9] Paolo Bientinesi and Robert van de Geijn. *The Science of Deriving Stability Analyses.* In preparation.

[10] Paolo Bientinesi, Sergey Kolos and Robert van de Geijn, *Automatic Derivation of Linear Algebra Algorithms with Application to Control Theory.* In Proceedings of PARA'04 State-of-the-Art in Scientific Computing, June 20-23, 2004.

[11] Paolo Bientinesi, Inderjit S. Dhillon, Robert A. van de Geijn. *A Parallel Eigensolver for Dense Symmetric Matrices Based on Multiple Relatively Robust Representations.* SIAM Journal on Scientific Computing, 27(1), 43-66, 2005.

[12] Paolo Bientinesi, Enrique Quintana-Ortí and Robert van de Geijn. *Representing Linear Algebra Algorithms in Code: The FLAME APIs.* ACM Transactions on Mathematical Software, 31(1), March 2005.

[13] Paolo Bientinesi and Robert van de Geijn. *Representing Dense Linear Algebra Algorithms: A Farewell to Indices.* FLAME Working Note #17. The University of Texas at Austin, Department of Computer Sciences. Technical Report TR-2006-10. February 2006.

[14] Paolo Bientinesi and Xiaobai Sun. *Numerical Calculation of the Prolate Functions for Energy-concentrated Approximations of Bandlimited Functions.* In preparation.

[15] Daniele Finocchiaro, Marco Pellegrini and Paolo Bientinesi. *On Numerical Approximation of Electrostatic Energy in 3D.* Journal of Computational Physics 146/2, 707-725, 1998.