INRIA

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Project-Team Alchemy

Architectures, Languages and Compilers to Harness the End of Moore Years

Saclay - Île-de-France

Theme : Architecture and Compiling

Table of contents

1. Team

Research Scientists

Olivier Temam [Research Director (DR) Inria, Team Leader, HdR]
Albert Cohen [Research Director (DR) Inria, HdR]
Christine Eisenbeis [Research Director (DR) Inria]
Grigori Fursin [Research Associate (CR) Inria]

Faculty Members

Sid-Ahmed-Ali Touati [Assistant professor, University of Versailles-Saint-Quentin, delegation INRIA, since September, 2009]
Cédric Bastoul [Assistant Professor]
Frédéric Gruau [Assistant Professor]
Jean-Luc Gaudiot [professeur invité Digiteo, April-May, 2010]

External Collaborators

Pierre Amiranoff [PRAG, IUT d'Orsay, University of Paris-Sud 11]
Nathalie Drach [Professor, University Pierre et Marie Curie]

PhD Students

Frédéric Brault [Engineer at Kalray]
Ramakrishna Upadrasta [MENRT scholarship, University of Paris-Sud 11, and STMicroelectronics contract]
Feng Li [INRIA scholarship, University Pierre et Marie Curie, since September 2010]
Walid Benabderrhamane [MENRT scholarship, University of Paris-Sud 11, until May 2010]
Mouad Bahi [Inria scholarship, University of Paris-Sud 11]
Cupertino Miranda [Portugese grant, University of Paris-Sud 11]
Konrad Trifunovic [Inria scholarship, University of Paris-Sud 11]
Boubacar Diouf [MENRT scholarship then ATER (half), University of Paris-Sud 11]
Mounira Bachir [ATER, University of Versailles-Saint-Quentin, until september, 2010]
Olivier Certner [STMicroelectronics fellowship (CIFRE), University of Paris-Sud 11]
Mohammed Fellahi [ATER (half), University of Paris-Sud 11, until August, 2010]
Anne-Sophie Coquel [Large Scale Initiative ColAge]
Michael Kruse [contrat doctoral de l'University of Paris-Sud 11, from september 2010]
Taj Muhammad Khan [Inria scholarship, University of Paris-Sud 11]
Zheng Li [Inria scholarship, University of Paris-Sud 11]
Luidnel Maignan [MENRT scholarship, University of Paris-Sud 11, until september, 2010, then ATER, University of Paris-Sud 11]
Louis-Noël Pouchet [MENRT scholarship, University of Paris-Sud 11]
Sean Halle [Inria expert engineer, U. of California Santa Cruz]
Marouane Belaoucha [MESR scolarship, co-supervised by S. Touati, University of Versailles-Saint-Quentin]

Post-Doctoral Fellows

Sven Verdoolaege [Expert engineer, Systematic competitivity cluster]
Anna Beletksa [Inria postdoc, Systematic competitivity cluster, until August, 2010]
Philippe Dumont [Inria postdoc, FP7 IST grant, until August 2010]
Armin Größlinger [Inria postdoc, Systematic competitivity cluster, until April 2010]

Visiting Scientist

Joern Rennecke [November to December, 2010]

Administrative Assistant

Valérie Berthou [TR Inria]

Others

Riyad Baghdadi [Internship, until August, 2010]
Soufiane Baghdadi [Internship, until August, 2010]
Nicolas Zermati [Master 2 intern, from March, 2009 until September, 2010]

Howard Wong [Internship, from April, 2010 until September, 2010, collaboration with UCI, Los Angeles]
Abdelfetteh Louati [ADT expert engineer, until April 2010]

2. Overall Objectives

2.1. Overall Objectives

ALCHEMY is a joint Inria/University of Paris Sud research group.

The general research topics of the ALCHEMY group are architectures, languages and compilers for high-performance embedded and general-purpose processors. ALCHEMY investigates *scalable* architecture and compiler/programming solutions for high-performance general-purpose and embedded processors. ALCHEMY stands for Architectures, Languages and Compilers to Harness the End of Moore Years, referring to both the traditional processor architectures implemented using the current photo-lithographic processes, and novel architecture/language paradigms compatible with future and alternative technologies. The current emphasis of ALCHEMY is on the former part, and we are progressively increasing our efforts on the latter part.

The research goals of ALCHEMY span from short term to long term. The short-term goals target existing complex processor architectures, and thus focus on improving program performance on these architectures (software-only techniques). The medium-term goals target the upcoming CMPs (Chip Multi-Processors) with a large number of cores, which will result from the now slower progression of core clock frequency due to technological limitations. The main challenge is to take advantage of the large number of cores for a wide range of applications, considering that automatic parallelization techniques have not yet proved an adequate solution. In ALCHEMY, we explore joint architecture/programming paradigms as a pragmatic alternative solution. Finally, even longer term research is conducted with the goal of harnessing the properties of future and alternative technologies for processing purposes.

Most of the research in ALCHEMY attempts to jointly consider the hardware and software aspects, based on the premise that many of the limitations of existing architecture and compiler techniques stem from the lack of cooperation between architects and compiler designers. However, ALCHEMY addresses the aforementioned research goals through two different, though sometimes complementary, approaches. One approach considers that, in spite of their complexity, architectures and programs can still be accurately and efficiently modeled (and optimized) using *analytical* methods. The second approach considers the architecture/program pair already has or will reach a complexity level that will evade analytical methods, and explores a *complex systems* approach; the principle is to accept that the architecture/program pair is more easily understood (and thus optimized) based on its observed behavior rather than inferred from its known design.

3. Scientific Foundations

3.1. Scientific Foundations

In the sections below, the different research activities of ALCHEMY are described, from short-term to long-term goals. For most of the goals, both analytical and complex systems approaches are conducted.

3.1.1. A practical approach to program optimizations for complex architectures

This part of our research work is more targeted to single-core architectures but also applies to multi-cores. The rationale for this research activity is that compilers rely on architecture models embedded in heuristics to drive compiler optimizations and strategy. As architecture complexity increases, such models tend to be too simplistic, often resulting in inefficient steering of compiler optimizations.

3.1.1.1. Iterative optimization

Our general approach consists in acknowledging that architectures are too complex to embed reliable architecture models in compilers, and to explore the behavior of the architecture/program pair through repeated executions. Then, using machine-learning techniques, a model of this behavior is inferred from the observations. This approach is usually called *iterative optimization*.

In the recent years, iterative optimization has emerged as a major research trend, both in traditional compilation contexts and in application-specific library generators (like ATLAS or SPIRAL). The topic has matured significantly since the pioneering works of Mike O'Boyle [105] at University of Edinburgh, UK or Keith Cooper [60] at Rice University. While these research works successfully demonstrated the performance *potential* of the approach, they also highlighted that iterative optimization cannot become a *practical* technique unless a number of issues are resolved. Some of the key issues are: the size and structure of the search space, the sensitivity to data sets, and the necessity to build long transformation sequences.

Scanning a large search space. Transformation parameters, the order in which transformations are applied, and even which transformations must be applied and how many times, all form a huge transformation space. One of the main challenges of iterative optimization is to rapidly converge towards an efficient, if not optimal, point of the transformation space. Machine-Learning techniques can help build an empirical model of the transformation space in a simple and systematic way, only based on the observation of transformations behavior, and then rapidly deduce the most profitable points of the space. We are investigating how to correlate static and dynamic program features with transformation efficiency. This approach can speed up the convergence of the search process by one or two orders of magnitude compared to random search [32], [50] [71] [29].

We have also shown that by representing the impact of loop transformations using structured encoding derived from polyhedral program representation, it is possible to reduce the complexity of the search by several orders of magnitude [113], [112]. This encoding is further described in Section 3.1.1.

Finally we have found that it is possible to further speed up transformation space exploration by exploring several transformations during a single run [72]. Currently, one program transformation is explored for each loop nest, while performance often reaches a stable state soon after the start of the execution. We have shown that, assuming we properly identify the phase behavior of programs, it is possible to explore multiple transformations in each run.

Data set sensitivity. Iterative optimization is based on the notion that the compiler will discover the best way to optimize a program through repeatedly running the same program on the same data set, trying one or a few different optimizations upon each run. However, in reality, a user rarely needs to execute the same data set twice. Therefore, iterative optimization is based on the implicit assumption that the best optimization configuration found will work *well* for *all data sets* of a program. To the best of our knowledge, this assumption has never been thoroughly investigated. Most studies on iterative optimization repeatedly execute the same program/data set pair [59], [74], [70], [94], [33], only recently, some studies have focused on the impact of data sets on iterative optimizations [87], [46].

In order to explore the issue of data set sensitivity, we have assembled a data set suite, of 20 data sets per benchmark, for most of the MiBench [84] embedded benchmarks. We have found that, though a majority of programs exhibit stable performance across data sets, the variability can significantly increase with many optimizations. However, for the best optimization configurations, we find that this variability is in fact small. Furthermore, we show that it is possible to find a compromise configuration across data sets which is often within 5% of the best possible optimization configuration for most data sets, and that the iterative process can converge in less than 20 iterations (for a population of 200 optimization configurations). Overall, the preliminary conclusion, at least for the MiBench benchmarks, is that iterative optimization is a fairly robust technique across data sets, which brings it one step closer to practical usage.

Compositions of program transformations. Compilers impose a certain set of program transformations, an ordering of application and how many times each transformation is applied. In order to explore what are the possible gains beyond these strict constraints, we have manually optimized kernels and benchmarks, trying

to achieve the best possible performance assuming no constraint on transformation order, count or selection [108], [107]. The study helped us clarify which transformations bring the best performance improvements in general. But the main conclusion of that study is that surprisingly long compositions of transformations are sometimes needed (in one case, up to 26 composed loop transformations) in order to achieve good performance. Either because multiple issues must be tackled simultaneously or because some transformations act as enabling operations for other transformations.

As a result, we have started developing a framework facilitating the composition of long transformations. This framework is based on the polyhedral representation of program transformations [4] [78]. This framework also enables a more analytical approach to program optimization and parallelization, beyond the simple composition of transformations. This latter part is further developed in Section 3.1.1.

Putting it all together: continuous optimization. Increasingly, we are now moving toward automatizing the whole iterative optimization process. Our goal is to bring together, within a single software environment, the different aforementioned observations and techniques (search space techniques, data set sensitivity properties, long compositions of transformations,...). We are currently in the process of plugging these different techniques within GCC in order to create a tool capable of doing continuous, whole-program optimization, and even collaborative optimization across different users.

Hardware-Oriented applications of iterative optimization. Because iterative optimization can successfully capture complex dynamic/run-time phenomena, we have shown that the approach can act as a replacement for costly hardware structures designed to improve the run-time behavior of programs, such as out-of-order execution in superscalar processors. An iterative optimization-like strategy applied to an embedded VLIW processor [63] was shown to achieve almost the same performance as if the processor was fitted with dynamic instruction reordering support. We are also investigating applications of this approach to the specialization/idiomization of general-purpose and embedded processors [132]. Currently, we are exploring similar approaches for providing thread scheduling and placement information for CMPs without requiring costly run-time environment overhead or hardware support. This latter study is related to the work presented in Section 3.1.2.

3.1.1.2. Polyhedral program representation: facilitating the analysis and transformation of programs

As loop transformations are utterly important — performance-wise — and among the hardest to predictably drive through static cost models, their current support in compilers is disappointing. After decades of experience and theoretical advances, the best compilers can miss some of the most important loop transformations in simple numerical codes from linear algebra or signal processing codes. Performance hits of more than an order of magnitude are not uncommon on single-threaded code, and the situation worsens when automatically parallelizing or optimizing parallel code.

Our previous work on sequences of loop transformations [4] has led to the design of a theoretical framework, based on the polyhedral model [67], [68], [69], [114], [102], [130], and a set of tools based on the advanced Open64 compiler. We have shown that this framework does simplify the problem of building complex transformation sequences, but also that it scales to real-world benchmarks [57], [125], [126], [78], and allows to significantly reduce the size of the search space and better understand its structure [113], [112], [111]. The latter work, for example, is the first attempt at directly characterizing all *legal and distinct* ways to reschedule a loop nest.

After two decades of academic research, the polyhedral model is finally evolving into a mature, production-ready approach to solve the challenges of maximizing the scalability and efficiency of statically-controlled, loop-based computations on a variety of high performance and embedded targets. After Open64, we are now porting these techniques to the GCC compiler [110], applying them to several multi-level parallelization and optimization problems, including vectorization, extraction and exploitation of thread-level parallelism on distributed memory CMPs like the Cell broadband engine from IBM, NXP's CAT-DI scalable signal-processing accelerator and novel STMicroelectronics emerging xStream architecture.

3.1.1.3. Project-team positioning

Note: The goal of this section and others alike is to not to act as a traditional and exhaustive "related work" section as found in research articles, but rather to provide references to a few research works which are the closest to our own.

While iterative optimization is based on simple principles which have been proposed a long time ago, this approach has been significantly developed by Mike O'Boyle at University of Edinburgh since 1997 [105], and more recently by Keith Cooper at Rice University [60]. Since then, many research groups have shown example cases where an iterative approach might be profitable (various application targets, various steps of the compilation process, various architecture components) [128], [119], [89], [127]. These researchers have shown that iterative optimization has a significant *potential*. Since then, other research groups (Polaris group at University of Illinois, CAPS at INRIA) have successfully demonstrated that iterative optimization can be used in practice for the design of libraries [98], [104], or even that it can be integrated in production compilers to assist existing optimizations [123]. As mentioned before, ALCHEMY is now focusing on the issues which hinder its *practical application*.

3.1.2. Joint architecture/programming approaches

While Section 3.1.1 is only concerned with transforming programs for a more efficient exploitation of existing architectures, in the longer term, researchers can assume modifications of architectures and/or programs are possible. These relaxed constraints allow to target the root causes of poor architecture/program performance.

The current architecture/program model partly fails because the burden is either excessively on the architecture (superscalar processors), or the compiler (VLIW and now CMPs). And both compiler and architecture optimizations often aim at program *reverse-engineering*: compilers attempt to dig up program properties (locality, parallelism) from the static program, while architectures attempt to retrieve them from program run-time behavior. Now, in many cases, the user is not only aware of these properties but may pass them effortlessly to the architecture and the compiler provided she had the appropriate programming support, provided the compiler would pass this information to the architecture, and the architecture would be fitted with the appropriate support to take advantage of them. For instance, simply knowing that a C structure denotes a tree rather than a graph can provide significant information for parallel execution. Such approaches, while not fully automatic, are practical and would relieve the complexity burden of the architecture and the compiler, while extracting significant amounts of task-level parallelism.

In the paragraphs below we apply this approach of passing more program semantic to the compiler and the architecture, first for domain-specific stream-oriented programs, and then for the parallelization of more general programs.

3.1.2.1. A targeted domain: Passing program semantics using a synchronous language for high-performance video processing

While we are currently investigating the aforementioned approach for general-purpose applications, we have started with the investigation of the specific domain of high-end video processing. In this domain, assessing that real-time properties will be satisfied is as important as reaching uncommon levels of compute density on a chip. 150 giga-operations per second per Watt (on pixel components) is the norm for current high-definition TVs, and cannot be achieved with programmable cores at present. The future standards will need an 8-fold increase (e.g., for 3D displays or super-high-definition). Predictability and efficiency are the keywords in this domain, in terms of both architecture and compiler behavior.

Our approach combines the aforementioned iterative optimization and polyhedral modeling research with a predictability- and efficiency-oriented parallel programming language. We focus on warrantable (as opposed to best-effort) usage of hardware resources with respect to real-time constraints. Therefore, this parallel programming language must allow overhead-free generation of tightly coupled parallel threads, interacting through dedicated registers rather than caches, streaming data through high-bandwidth, statically managed interconnect structures, with frequent synchronizations (once every few cycles), and very limited memory resources immediately available. This language also needs to support advanced loop transformations, and its representation of concurrency compatible with the expression of multi-level partitioning and mapping

decisions. All these conditions tend to consider a language closer to hardware synthesis languages than general-purpose, von Neumann oriented imperative ones [51], [56].

The synchronous data-flow paradigm is a natural candidate, because of its ability to combine high-productivity in programming complex concurrent applications (due to the determinism and compositionality of the underlying model, a rare feature of a concurrent semantics), direct modeling of computation/communication time, and static checking of non-functional properties (time and resource constraints). Yet generating low-level, tightly fused loops with maximal exposition of fine-grain parallelism from such languages is a difficult problem, as soon as the target processor is not the one being described by the synchronous data-flow program, but a pre-existing target on which we are folding an application program. The two tasks are totally different: although the most difficult decisions are pushed back to the programmer in the hardware synthesis case, application programmers usually rely on the compiler to abstract away the folding of their code in a reasonably portable fashion across a variety of targets. This aspect of synchronous language compilation has largely been overlooked and constitutes the main direction of our work. Another direction lies in the description of hardware resources, at the same level as the application being mapped and scheduled onto them; this unified representation would allow the expression of the search space of program transformations, and would be a necessary step to apply incremental refinement methods (expert-driven, very popular in this domain).

Technically, we extend the classical clock calculus (a type system) of the *Lucid Synchrone* language, expliciting significantly more information about the program behavior, especially when tasks must be started and will be completed, how information flow among tasks, etc. Our main contribution is the integration of relaxed synchronous operators like jittering and bursty streams within synchronous bounds [54], [55]. This research consists in revisiting the semantics of synchronous Kahn networks in the domain of media streaming applications and reconfigurable parallel architectures, in collaboration with Marc Duranton from Philips Research Eindhoven (now NXP Semiconductors) and with Marc Pouzet from LRI and the Proval INRIA project team.

3.1.2.2. A more general approach: Passing program semantic using software components

Beyond domain-specific and regular applications (loops and arrays), automatic compiler-based parallelization has achieved only mixed results on programs with complex control and data structures [85]. Writing, and especially debugging, large parallel programs is a notoriously difficult task [90], and one may wonder whether the vast majority of programmers will be able to cope with it. Currently, transactional memory is a popular approach [86] for reducing the programmer burden using intuitive transaction declarations instead of more complex concurrency control constructs. However, it does not depart from the classic approach of parallelizing standard C/C++/Fortran programs, where parallelism can be difficult to extract or manipulate. Parallel languages, such as HPF [99], require more ambitious evolutions of programming habits, but they also let programmers pass more semantic about the control and data characteristics of programs to the compiler for easier and more efficient parallelization. However, one can only observe that, for the moment, few such languages have become popular in practice.

A solution would have a better chance to be adopted by the community of programmers at large if it integrates well with popular practices in *software engineering*, and this aspect of the parallelization problem may have been overlooked. Interestingly, software engineering has recently evolved towards programming models that can blend well with multi-core architectures and parallelization. Programming has consistently evolved towards more encapsulation: procedures, then objects, then *components* [120]. Essentially for two reasons, because programmers have difficulties grasping large programs and need to think locally, and because encapsulation enables *reuse* of programming efforts. Component-based programming, as proposed in Java Beans, .Net or more ad-hoc component frameworks, is the step beyond C++ or Java objects: programs are decomposed into modules which fully encapsulate code and data (no global variable) and which communicate among themselves through explicit interfaces/links.

Components have many assets for the task of developing parallel programs. (1) Components provide a pragmatic approach for bringing parallelization to the community at large thanks to component reuse. (2) Components provide an implicit and intuitive programming model: the programmer views the program as a "virtual space" (rather than a sequence of tasks) where components reside; two components residing together

in the space and not linked or not communicating through an existing link implicitly operate in parallel; this virtual space can be mapped to the physical space of a multi-threaded/multi-core architecture. (3) Provided the architecture is somehow aware of the program decomposition into components, and can manipulate individual components, the compiler (and the user) would be also relieved of the issue of mapping programs to architectures.

In order to use software components for large-scale and fine-grain parallelization, the key notion is to augment them with the ability to split or replicate. For instance, a component walking a binary tree could spawn two components to scan two child nodes and the corresponding sub-trees in parallel.

We are investigating a low-overhead component-based approach for fine-grain parallelism, called CAPSULE, where components have the ability to replicate [96], [106]. We investigate both a hardware-supported and software-only approach to component division. We show that a low-overhead component framework, possibly paired with component hardware support, can provide both an intuitive programming model for writing fine-grain parallel programs with complex control flow and data structures, and an efficient platform for parallel components execution.

3.1.2.3. Personnel

3.1.2.4. Project-team positioning

As explained before, both approaches pursued rely on the same philosophy, pass more program semantic to the compiler and the architecture, though the techniques differ significantly. Naturally, there is a huge body of literature on parallelization, and here, we can only hint at some of the main research directions. Current approaches either rely on automatic parallelization [34] of standard programs, but the automatic parallelization of "complex" applications (complex control flow and data structures) has registered mixed results. Another approach is software/hardware thread-level speculation, but one may question its cost and scalability [115]. As mentioned before, transactional memory has become a popular approach [86] for reducing the burden of parallelizing applications. Other approaches include parallel languages, such as HPF [99] or parallel directives such as OpenMP [61].

Synchronous data-flow languages. The synchronous data-flow approach to the design and optimization of massively parallel, highly compute-efficient and predictable systems is quite unique. It is a long-term, largely fundamental effort motivated by well-established practices in the industry, mostly in the domain of high-definition language programming for hardware synthesis, and combines these practices with the best semantic properties of high-level programming languages. It is a holistic approach to combining productivity *and* scalability *and* compute-efficiency in a unified design, targeting the domain of real-time, predictable, stream-oriented parallel systems.

The closest work is the StreamIt language and compiler from MIT [122], and to a lesser extent, the Sequoia project from Stanford [65]; these two mature projects achieved important contributions in the exposition and exploitation of thread-level parallelism on a coarse grain distributed-memory, stream-oriented architecture. StreamIt is also much more limited in expressiveness, and Sequoia is more an incremental progress on how to compile and optimize a parallel program than a productivity-oriented design of a new concurrent programming paradigm. We are currently working on a shorter term, intermediate milestone much closer to these two projects, but allowing to expose and exploit multi-level parallelism, at all stages of the design-space exploration and in all passes of the compiler.

Software components. Software components, as provided in the .Net or Java Beans frameworks, have little support for parallelism. Several years ago, a few frameworks proposed a component-like approach for parallelizing complex applications on large-scale multiprocessors, especially the Cilk [48] and Charm++ [92] frameworks. However Cilk does not promote encapsulation, essentially a mechanism for spawning C functions. Charm++ provides both encapsulation and spawning, but it targets large-scale multiprocessors, even grid computing [93], and its overhead is rather large for fine-grain parallelism as required by multi-threaded/multi-core architectures.

Probably the closest work to our hardware support for components is the Network-Driven Processor proposed by Chen et al. [53] which aims at implementing CMP hardware support for Cilk programs. Thread creation decisions are not taken directly by the architecture, they enact any thread spawning decision taken by the Cilk environment, but they provide a sophisticated support for communications and work stealing between processors.

3.1.3. Alternative computing models/Spatial computing

The last research direction stems from possible evolutions of technology. While this research direction may seem very long term, processor manufacturers cannot always afford to investigate many risky alternatives way ahead in time. At the same time, for them to accept and adopt radical changes, they have to be anticipated long in advance. Thus, we believe prospective research is a core role for academic researchers, which may be less immediately useful to companies, but which can bring a real addition to their internal research activities, and which also carries the potential of bringing disruptive technology.

Prospective information on the future of CMOS technology suggests that, though the density of transistors will keep increasing, the commuting speed of transistors will not increase as fast, and transistors may be more faulty (either fabrication defects or execution faults). Possible replacement/alternative technologies, such as nanotubes [79] which have received a lot of attention lately, share many of these properties: high density, but slow components (possibly even slower than current components), a large rate of defects/faults, and more difficulty to place them except than in fairly regular structures.

In short, several potential upcoming technologies seem to bring a very large number of possibly faulty and not so fast components with layout issues. For architectures to take advantage of such technology, they would have to rely on *space* much more than *time/speed* to achieve high performance. Large spatial architectures bring a set of new architecture issues, such as controlling the execution of a program in a totally decentralized way, efficiently managing the placement of program tasks on the space, and managing the relative movement of these different tasks so as to minimize communications. Furthermore, beyond a certain number of processing elements, it is not even clear whether many applications will embed enough traditional task-level parallelism to take advantage of such large spaces, so applications may have to be expressed (programmed) differently in order to leverage that space. These two research issues are addressed in the two research activities described below.

Blob computing. Blob computing [83] is both a spatial programming and architecture model which aims at investigating the utilization of a vast amount of processing elements. The key originality of the model is to acknowledge that the chip space becomes too large for anything else than purely *local* actions. As a result, all architecture control becomes local. Similarly, the program itself is decomposed into a set of purely local actions/tasks, called Blobs, connected together through links; the program can create/destroy these links during its lifetime.

With respect to architecture control, for instance, the local method for expressing that two tasks frequently communicating through a link must get close together in space so that their communication latency is low is expressed through a simply physical law, emulating spring tension; the more communications, the higher the tension. Similarly, expressing that tasks should move away because too many tasks are grouped in the same physical spot is achieved through a law similar to pressure: as the number of tasks increases, the local pressure on neighbor tasks increases, inducing them to move away. Overall many of these local control rules derive from physical or biological laws which achieve the same goals: controlling a large space through simple local interactions.

With respect to programming, the user essentially has to decompose the program into a set of nodes and links. The program can create a static node/link topology that is later used for computations, or it can dynamically change that topology during execution. But the key concept is that the user is not in charge of placing tasks on the physical space, only to express the *potential* parallelism through task division. As can be observed, several of the intuitions of the CAPSULE environment of Section 3.1.2.2 stems from this Blob model.

Bio-Inspired computing. As mentioned above, beyond a certain number of individual components, it is not even clear whether it will be possible to decompose tasks in such a way they can take advantage of a large

space. Searching for pieces of solution to this problem has progressively lead us to biological neural networks. Indeed, biological neural networks (as opposed to artificial neural networks, ANNs) are well-known examples of systems capable of complex information processing tasks using a large number of self-organized, but slow and unreliable components. And the complexity of the tasks typically processed by biological neurons is well beyond what is classically implemented with ANNs.

Emulating the workings of biological neural networks may at first seem far-fetched. However, the SIA (Semiconductor Industry Association) in its 2005 roadmap addresses for the first time "biologically inspired architecture implementations" [116] as emerging research architectures, and focuses on biological neural networks as interesting scalable designs for information processing. More importantly, the computer science community is beginning to realize that biologists have made tremendous progress in the understanding of how certain complex information processing tasks are implemented using biological neural networks.

One of the key emerging features of biological neural networks is that they process information by *abstracting* it, and then only manipulate such higher abstractions. As a result, each new input (e.g., for image processing) can be analyzed using these learned abstractions directly, thus avoiding to rerun a lengthy set of elementary computations. More precisely, Poggio et al. [109] at MIT have shown how combinations of neurons implementing simple operations such as MAX or SUM, can automatically create such abstractions for image processing, and some computer science researchers in the image processing domain have started to take advantage of these findings.
We are starting to investigate the information processing capabilities of this abstraction programming method [118], [117], [44] [45]. While image processing is also our first application, we plan to later look at a more diverse set of example applications.

A complex systems approach to computing systems. More generally, the increased complexity of computing systems at stake, whether due to a large number of individual components, a large number of cores or simply complex architecture program/pairs, suggest that novel design and evaluation methodologies should be investigated that rely less on known design information than on observed behavior of the global resulting system. The main problem here is to be able to extract general characteristics of the architecture on the basis of measurements of its global behavior. For that purpose, we are using tools provided by the physics of complex systems (nonlinear time series analysis, phase transitions, multi-fractal analysis...).
We have already applied such tools to better understand the performance behavior of complex but traditional computing systems such as superscalar processors [42], [43]. And we are starting to apply them to sampling techniques for performance evaluation [80], [81]. We will be progressively expanding the reach of these techniques in our research studies in the future.

3.1.3.1. Project-team positioning

While spatial computing is an expression used for many purposes [79], the Blob computing work in our research group refers more to unconventional spatial programming paradigms such as MGS [76] and Gamma [38].

There has recently been a surge of research works targeting novel technologies in computer architecture,but they have mostly focused on quantum computing, and, to our knowledge, few have focused on bio-inspired computing.
Furthermore, several researchers in the computer science community have recently started applying ideas from complex systems approaches. But their focus are usually on the software or algorithm part. Our utilization of complex systems approaches in the field of architecture is thus less investigated, although other groups have very recently expressed similar interests [95], [121].

3.1.4. Transversal research activities: simulation and compilation

Since our research group has been involved in both compiler and architecture research for several years, we have progressively given increased attention to tools, partly because we found a lot of productivity was lost in inefficient or hard to reuse tools. Since then, both simulation and compilation platforms have morphed into research activities of their own. Our group is now coordinating the development of the simulation platform of the European HiPEAC network, and it is co-coordinating the development of the compiler research platform of HiPEAC together with University of Edinburgh.

3.1.4.1. Simulation platform

As processor architecture and program complexity increase, so does the development and execution time of simulators. Therefore, we have investigated simulation methodologies capable of increasing our research productivity. The key point is to improve the reuse, sharing, comparison and speed capabilities of simulators. For the first three properties, we are investigating the development of a *modular* simulation platform, and for the latter fourth property, we are investigating sampling techniques and more abstract modeling techniques. Our simulation platform is called UNISIM [31].

What is UNISIM? UNISIM is a structural simulation environment which provides an intuitive mapping from the hardware block diagram to the simulator; each hardware block corresponds to a simulation module. UNISIM is also a library of modules where researchers will be able to download and upload (contribute) modules.

What are the assets of UNISIM over other simulation platforms? UNISIM allows to reuse, exchange and compare simulator parts (and architecture ideas), something that is badly needed in academic research, and between academia and industry. Recently, we did a comparison of 10 different cache mechanisms proposed over the course of 15 years [82], and suggested the progress of research has been all but regular because of the lack of a common ground for comparison, and because simulation results are easily skewed by small differences in the simulator setup.

Also, other simulation environments or simulators advocate modular simulation for sharing and comparison, such as the SystemC environment [30], or the M5 simulator [47]. While they do improve the modularity of simulators, in practice, reuse is still quite difficult because most simulation environments overlook the difficulty and importance of reusing *control*. For instance, SystemC focuses on reusing hardware blocks such as ALUs, caches, and so on. However, while hardware blocks correspond to the greatest share of transistors in the actual design, they often correspond to the least share of simulator lines. For instance, the cache data and instruction banks often correspond to a sizable amount of transistors, but they merely correspond to array declarations in the simulator; conversely, cache control corresponds to few transistors but most of the source lines of any cache simulator function/module. As a result, it is difficult to achieve reuse in practice, because control code is often not implemented in such a way that it can lend well to reuse.

On the contrary, UNISIM is focused on reuse of control code, provides a standardized module communication protocol and a control abstraction for that purpose. Moreover, UNISIM will later on come with an open library in order to better structure the set of available simulators and simulator components.

Taking a realistic approach at simulator usage. Obviously, many research groups will not accept easily to drop years of investment in their simulation platforms and to switch to a new environment. We take a pragmatic approach and UNISIM is designed from the ground up to be interoperable with existing simulators, from industry and academia. We achieve interoperability by wrapping full simulators or simulator parts within UNISIM modules. We have an example full SimpleScalar simulator stripped of its memory, wrapped into a UNISIM module, and plugged into a UNISIM SDRAM module.

Moreover, we are in the process of developing a number of APIs (for power, GUI, functional simulators, sampling,...) which will allow third-party tools to be plugged into the UNISIM engine. We call these APIs simulator capabilities or services.

With CMPs, communications become more important than cores cycle-level behavior. While the current version of UNISIM is focused on cycle-level simulators, we are developing a more abstract view of simulators

called Transaction-Level Models (TLM). Later on, we will also allow hybrid simulators, using TLM for prototyping, and then zooming on some components of a complex system.

Because CMPs also require operating system support for a large part, and because existing alternatives such as SIMICS [100] are not open enough, we are also developing full-system support in our new simulators jointly with CEA. Currently, UNISIM has a functional simulator of a PowerPC750 capable of booting Linux.

3.1.4.2. Compilation platform

The free *GNU Compiler Collection* (GCC) is the leading tool suite for portable developments on open platforms. It supports more than 6 input languages and 30 target processor architectures and instruction sets, with state-of-the-art support for debugging, profiling and cross-compilation. It has long been supported by the general-purpose and high-performance hardware vendors. The last couple of years have seen GCC taking momentum in the embedded system industry, and also as a platform for advanced research in program analysis, transformation and optimization.

GCC 4.4 features about 200 compilation passes, two thirds of them playing a direct role in program optimization. These passes are selected, scheduled, and parametrized through a versatile pass manager. The main families of passes can be classified as:

- inter-procedural analyzes and optimizations;
- profiling, coverage analysis and instrumentation;
- induction variable analysis, canonicalization and strength-reduction;
- loop optimization, automatic vectorization and parallelization;
- data layout optimization.

More advanced developments involving GCC are in progress in the ALCHEMY group:

- global, whole program optimization (towards link-time and just-in-time compilation), with emphasis on scalability;
- transactional memory extensions independent from yet compatible with OpenMP, and a recent intrusion into data-flow synchronous programming;
- polyhedral loop nest optimization, with support for automatic vectorization in the Graphite branch of GCC; this branch has merged with GCC 4.4; it was initiated by the ALCHEMY group and a former student now at AMD (Sebastian Pop);
- automatic parallelization, including the extraction and adaptation of loop and pipeline parallelism, with extensions towards speculative forms of parallelism.

The HiPEAC network supports GCC as a platform for research and development in compilation for high-performance and embedded systems. The network's activities on the compiler platform are coordinated by Albert Cohen.

3.1.4.3. Project-team positioning

Simulation (UNISIM). The rationale for the simulation effort, and the current situation in the community (dominance of monolithic simulators like SimpleScalar [49]) has been described as part of the presentation of this research activity in Section 3.1.4. While several companies have internal modular simulation environments (ASIM at Intel [64], TSS at Philips, MaxSim at ARM,...), they are not standard nor disseminated. Only SystemC [30] is gaining wide acceptance as a modular simulation environment with companies, less so with high-performance academic research groups. The academic research group which has the most similar approach is the Liberty group at Princeton University. They have been similarly advocating modular simulation in the past few years [124]. Due to the growing importance of CMP architectures, several research groups have since then proposed CMP simulation platforms, some of them with modularity properties, such as M5 [47], Flexus [28], GEMS [101] or Vasa [129].

Finally, UNISIM is also participating to a French simulation platform called SoCLib through a recent contract (SoCLib). The technical goals of UNISIM are rather different as we initially targeted processor decomposition into modules while SoCLib targeted systems-on-chip. As architectures are moving to multi-cores, the collaboration could become fruitful. UNISIM is also more focused on trying to gather, from the start, groups from different countries in order to increase the chances of adoption.

Compilation (GCC). We are also deeply committed to the enhancement and popularization of GCC as a common compilation research platform. The details of this investment are listed in Section 3.1.4. GCC is of course an interesting option for the industry, as development costs surge and returns in performance gains quickly diminish with the complexity of the modern architectures. But GCC is also, and for the first time, a serious candidate to help researchers mutualize development efforts, to experiment their contributions in a complete tool chain with production codes, to enable the sharing and comparison of these contributions in an open licensing model (a necessary condition for assessing the quality of experimental results), and to facilitate the transfer of these contributions to production environments (with an immediate impact on billions of embedded devices, general-purpose computers and servers). Learning from the failures of a well known attempt at building a common compiler infrastructure (SUIF-NCI in the late 90s), we follow a pragmatic approach based on joint industry-academia research projects 6.1), training (tutorials, courses, see Section 3.1.4), and direct contributions to the enhancement of the platform (e.g., for iterative optimization research and automatic parallelization).

4. Software

4.1. Main software developments

4.1.1. Main software developments

Participants: Veerle Desmet, Sylvain Girbal, Zheng Li, Olivier Temam.

COMPILERS & PROGRAM OPTIMIZATION:

Polyhedral transformations in Open64 The WRaP-IT tool (WHIRL Represented as Polyhedra – Interface Tool) is a program analysis and transformation tool implemented on top of the Open64 compiler [40] and of the CLooG code generator [39]. The formal basis of this tool is the polyhedral model for reasoning about loop nests. We introduced a specific polyhedral representation that guarantees strong transformation compositionality properties [58]. This new representation is used to generalize classical loop transformations, to lift the constraints of classical compiler frameworks and enable more advanced iterative optimization and machine learning schemes. WRaP-IT — and its loop nest transformation kernel called URUK (Unified Representation Universal Kernel) — is designed to support a wide range of transformations on industrial codes, starting from the SPEC CPU2000 benchmarks, and recently considering a variety of media and signal processing codes (vision, radar, software radio, video encoding, and DNA-mining in particular, as part of the IST STREP ACOTES, ANR CIGC PARA, and a collaboration with Thales).

Based on this framework, we are also planning an extension of the polyhedral model to handle speculative code generation and transformation of programs with data-dependent control, and a direct search and transformation algorithm based on the Farkas lemma. These developments will take place in the GRAPHITE project: a migration/rewrite of our Open64-based software to the GCC suite. This project is motivated by the maturity — performancewise and infrastructurewise — of GCC 4.x, and on the massive industrial investment taking off on GCC in the recent years, especially in the embedded world. We are heavily involved in fostering research projects around GCC as a common compilation platform, and GRAPHITE is one of those projects.

Grigori Fursin developed the first prototype of an iterative optimization API for GCC, and started using this infrastructure for continuous and adaptive optimization research, in collaboration with the University of Edinburgh.

Candl **Participants:** Cédric Bastoul, Louis-Noël Pouchet.

Candl is a free software and a library devoted to data dependences computation. It has been developed to be a basic bloc of our optimizing compilation tool chain in the polyhedral model. From a polyhedral representation of a static control part of a program, it is able to compute exactly the set of statement instances in dependence relation. Hence, its output is useful to build program transformations respecting the original program semantics. This tool has been designed to be robust and precise. It implements some usual techniques for data dependence removal, as array privatization or array expansion.

Clan **Participants:** Cédric Bastoul, Louis-Noël Pouchet, Walid Benabderrahmane.

Clan is a free software and library that translates some particular parts of high level programs written in C, C++, C# or Java into a polyhedral representation (strict or extended to irregular control flow). This representation may be manipulated by other tools to, e.g., achieve complex program restructurations (for optimization, parallelization or any other kind of manipulation). It has been created to avoid tedious and error-prone input file writing for polyhedral tools (such as CLooG, LeTSeE, Candl etc.). Using Clan, the user has to deal with source codes based on C grammar only (as C, C++, C# or Java).

CLooG **Participants:** Cédric Bastoul, Walid Benabderrahmane, Louis-Noël Pouchet, Sven Verdoolaege.

CLooG is a free software and library to generate code for scanning Z-polyhedra. That is, it finds a code (e.g. in C, FORTRAN...) that reaches each integral point of one or more parameterized polyhedra. CLooG has been originally written to solve the code generation problem for optimizing compilers based on the polytope model. Nevertheless it is used now in various area e.g. to build control automata for high-level synthesis or to find the best polynomial approximation of a function. CLooG may help in any situation where scanning polyhedra matters. While the user has full control on generated code quality, CLooG is designed to avoid control overhead and to produce a very effective code. Irregular extentions have been integrated during 2009 in the irCLooG prototype.

OpenScop **Participants:** Cédric Bastoul, Louis-Noël Pouchet.

OpenScop is an open specification that defines a file format and a set of data structures to represent a static control part (SCoP for short), i.e., a program part that can be represented in the polyhedral model. The goal of OpenScop is to provide a common interface to the different polyhedral compilation tools in order to simplify their interaction. To help the tool developpers to adopt this specification, OpenScop comes with an example library (under 3-clause BSD license) that provides an implementation of the most important functionalities necessary to work with OpenScop.

FADAlib (http://www.prism.uvsq.fr/users/bem/fadalib/home.html). Dataflow dependence for irregular programs (not static control programs). The library is developped by M. Belaoucha, funded by projects Teraops (pole de competitivite systematic) and PARMA (ITEA2).

MAQAO (modular assembly quality analyzer and optimizer, http://maqao.prism.uvsq.fr/). MAQAO analyzes static assembly codes and dynamic application performance. The objective of MAQAO is to help developpers to focus on code fragments that require performance tuning, analyzes compiler optimizations and proposes tuning hints. MAQAO works on Itanium, Pentium architectures.

CAPSULE. **Participants:** Olivier Certner, Yves Lhuillier, Zheng Li, Pierre Palatin, Olivier Temam.

CAPSULE is our component-like parallelization environment. It consists of a runtime system which enacts tasks divisions. The environment is publicly disseminated at `alchemy.futurs.inria.fr/capsule`, along with several CAPSULE-parallelized benchmarks. CAPSULE was developed through several

PROCESSOR SIMULATION:

archexplorer.org The project can be summarized as an open and continuous exploration of the architecture design space, and takes the form of a service and web site we have just opened, `www.archexplorer.org`, hosting the software at the server side.

The goal of this project is twofold: to enable a more rigorous methodology approach in our domain by enabling the comparison of architecture ideas, and to propose a novel architecture design approach which relies on automatic design-space exploration, as an alternative, or at least a complement, to the current design process essentially driven by intuition and experience.

The server-side software is mostly based on UNISIM (`www.unisim.org`), one of our large developments in software simulation: it corresponds to an environment on top of SystemC for truly enabling sharing, reuse and comparison, by offering a rigorous communication protocol between modules, architecture interfaces, and a set of simulators.

The archexplorer.org project is a joint project with Ghent University, Belgium (Veerle Desmet), and Thales TRT (Sylvain Girbal). I have started the project and I am coordinating the research, though Veerle and Sylvain are doing most of the implementation work; Veerle also has taken an active role in the project and can be considered as co-leading it.

UNISIM The UNISIM platform has been described in Section 3.1.4. As of now, besides the simulation engine, the developments include a shared-memory CMP based on the PowerPC 405, functional simulators for the PowerPC 405 (and cycle-level), PowerPC 750, a functional system simulator of the PowerPC 750 capable of booting Linux, 10 different cache modules corresponding to various research works. The following simulators or tools are currently under development: a functional and cycle-level version of the ARM 9 with full-system capability, a distributed-memory CMP based on the Power 405 core, an ST231 VLIW functional and later on cycle-level simulator. During his internship, Taj Khan integrated the CACTI (http://www.hpl.hp.com/personal/Norman_Jouppi/cacti4.html) Power Estimation Model developed at HP Labs in UniSim.

5. New Results

5.1. Program optimizations

5.1.1. Practical Approach

Participants: Cédric Bastoul, Walid Benabderrahmane, Albert Cohen, Grigori Fursin, Louis-Noël Pouchet, Olivier Temam.

Here are the most recent key scientific achievements.

- It was empirically demonstrated that iterative optimization is not data set sensitive, and as a result, the best compiler optimizations can be learned across data sets [52].
- The notion of automatically exploring compiler optimizations has been extended to the joint exploration of both hardware and software optimizations, with the set up of an automatic architecture/software exploration facility (archexplorer.org) [62].

5.1.2. Collective Tuning Center

Participants: Grigori Fursin, Olivier Temam.

We created an open community-driven collaborative wiki-based portal http://cTuning.org that brings together academia, industry and end-users to develop intelligent collective tuning technology that automates and simplifies compiler, program and architecture design and optimization. This technology minimizes repetitive time consuming tasks and human intervention using collective optimization, run-time adaptation, statistical and machine learning techniques. It can already help end users and researchers to improve execution time, code size, power consumption, reliability and other important characteristics of the available computing systems automatically (ranging from supercomputers to embedded systems) and should eventually enable development of the emerging intelligent self-tuning adaptive computing systems. Collective Optimization Database is intended to improve the quality of academic research by avoiding costly duplicate experiments and providing reproducible results.

5.1.2.1. Simulation of the Lattice QCD and Technological Trends in Computation

Participants: Mouad Bahi, Cédric Bastoul, Walid Benabderrhamane, Christine Eisenbeis, Jean-Luc Gaudiot, Julien Jaeger, Michael Kruse, Louis-Noël Pouchet, Howard Wong.

This is a joint ANR project "PetaQCD" with Lal (Orsay), Irisa Rennes (Caps/Alf), IRFU (CEA Saclay), LPT (Orsay), Caps Entreprise (Rennes), Kerlabs (Rennes), LPSC (Grenoble).

Simulation of the Lattice QCD is a challenging computational problem. Currently, technological trends in computation show multiple divergent models of computation. We are witnessing homogeneous multicore architectures, the use of accelerator on-chip or off-chip, in addition to the traditional architectural models.

On the verge of this technological abundance, assessing the performance tradeoffs of computing nodes based on these technologies is of crucial importance to many scientific computing applications.

In this study [91], we focus on assessing the efficiency and the performance expected for the Lattice QCD problem on representative architectures and we project the expected improvement on these architectures and their impact on performance for the Lattice QCD. We additionally try to pinpoint the limiting factors for performance on these architectures. This work takes place in ANR PARA and ANR QCDNEXT (both 2005-2008) and has led to the project ANR PetaQCD (2009-2011)[6].

5.2. Joint architecture/programming approaches

Here are the most recent key scientific achievements.

- A joint programming/architecture approach for streaming applications which is successfully used at NXP (formerly Philips Semiconductors). An extension of the synchronous Kahn process networks using a relaxed notion of synchrony, called N-synchrony, applied to the efficient and scalable parallelization of media streaming applications.

5.2.1. CAPSULE: division-based parallelization

Participants: Olivier Certner, Zheng Li, Olivier Temam.

We have decided to ride a popular trend in software engineering, *software components*, which blends well with multi-cores: it proposes to decompose a large program into smaller fully independent parts, just like multi-cores consist in decomposing large monolithic architectures into a set of smaller cores. In itself, componentization does not yield much parallelism, our contribution is to augment components with the ability to *divide*, yielding as much parallelism as resources allow. The programmer is only exposed to this very simple notion of parallelization, and the role of the architecture and/or the run-time system is to manage parallel tasks. We have shown that this approach performs well on programs with irregular control flow behavior and complex data structures, which are typically difficult to efficiently parallelize. We have developed a hardware support for making this parallelization approach efficient in distributed-memory multicores [97].

5.3. Alternative computing models

5.3.1. Compound circuits

Participants: Hugues Berry, Sylvain Girbal, Olivier Temam, Sami Yehia.

Besides parallelization, the other "spatial" scalability path is customization. Customization, which is very popular in embedded systems, has many assets: custom circuits are cheaper, faster and more power efficient than processors. They can also speed up tasks which are by nature sequential (not parallel), so that they are complementary, not an alternative, to parallelism. Their main limitation is *flexibility*. As a result, we have investigated techniques which can improve the flexibility of custom circuits while achieving the best possible performance, area and power properties. We have developed a novel bottom-up approach where we show how to efficiently combine any number of custom circuits to create a far more flexible *compound* circuit [131], without sacrificing the performance, area and power benefits of custom circuits. That approach was recently patented jointly with Thales. More recently, we have developed a memory interface for efficiently feeding data to these compound circuits [77], [35].

5.3.2. ANNs as accelerators

Participant: Olivier Temam.

We make the case for considering a hardware ANN as a flexible yet energy efficient, high-performance and defect-resilient accelerator, ideally positioned to tackle upcoming technology, applications and programming challenges. For now, we focus this study on one type of algorithms, classifiers, but which are commonly used in many RM applications. We present a hardware accelerator design for ANNs, geared towards robustness and high-performance. We show that transistor density has reached a level where it is now possible to spatially expand in hardware an ANN capable of handling medium-sized applications. Spatial expansion has multiple benefits in terms of robustness, energy efficiency, performance and scalability, over previous time-multiplexed designs.

We synthesized our design at 90nm and showed that such a spatially expanded ANN accelerator achieves orders of magnitude reductions in energy, and similar improvements in performance with respect to the same task executed on a modern processor at the same technology node, at a fraction of the on-chip area, justifying scaling down just one core in order to rip the energy and performance benefits.

5.3.3. Bio-Inspired Computing

5.3.3.1. AMYBIA : Aggregating MYriads of Bio-Inspired Agents

Participants: Hugues Berry, Nazim Fates, Bernard Girau.

In the framework of the ARC Amybia, we are searching for innovative schemes of decentralised and massively distributed computing. We mainly aim at contributing to this at three levels. At the modelling level, we think that biology provides us with complex and efficient models of such massively distributed behaviours. We start our study by addressing the decentralised gathering problem with the help of an original model of aggregation based on the behaviour of social amoebae. At the simulation level, our research mainly relies on achieving large scale simulations and on obtaining large statistical samples. Mastering these simulations is a major scientific issue, especially considering the imposed constraints: distributed computations, parsimonious computing time and memory requirements. Furthermore its raises further problems, such as: how to handle asynchronism, randomness and statistical analysis? At the hardware level, the challenge is to constantly confront our models with the actual constraints of a true practise of distributed computing. The main idea is to consider the hardware as a kind of sanity check. Hence, we intend to implement and validate our distributed models on massively parallel computing devices. In return, we expect that the analysis of the scientific issues raised by these implementations will influence the definition of the models themselves.// As a first step, we have recently proposed a bio-inspired system based on the so-called Greenberg-Hastings cellular automaton (GHCA), to achieve decentralized and robust gathering of mobile agents scattered on a surface or computing tasks scattered on a massively-distributed computing medium. As usual with such models, GHCA has mainly been studied using an homogeneous and regular lattice. However, in the context of massively distributed computing, one also needs to consider unreliable elements and defect-based noise. A first analysis showed that in this case, phase transitions could govern the behaviour of the system. Our next goal was to broaden the knowledge on stochastic reaction-diffusion media by investigating how such systems behave when various types of noise are introduced. Hence, in [66], we study GHCA where noise and topological irregularities of the grid are taken into account. The decrease of the probability of excitation changes qualitatively the behaviour of the system from an active to an extinct steady state. Simulations show that this change occurs near a critical threshold; it is identified as a nonequilibrium phase transition which belongs to the directed percolation universality class. We test the robustness of the phenomenon by introducing persistent defects in the topology : directed percolation behaviour is conserved. Using experimental and analytical tools, we suggest that the critical threshold varies as the inverse of the average number of neighbours per cell. The inverse proportionality law we presented paves the way for obtaining generic laws (even approximate ones) to predict the position of the critical threshold in various simulation conditions.

5.3.3.2. Cortical Microarchitecture: Computing by Abstractions

Participants: Hugues Berry, Olivier Temam.

Recent advances in the neuroscientific understanding of the brain are bringing about a tantalizing opportunity for building synthetic machines that perform computation in ways that differ radically from traditional Von Neumann machines. These brain-like architectures, which are premised on our understanding of how the human neocortex computes, are highly fault-tolerant, averaging results over large numbers of potentially faulty components, yet manage to solve very difficult problems more reliably than traditional algorithms. A key principle of operation for these architectures is that of automatic abstraction: independent features are extracted from highly disordered inputs and are used to create abstract invariant representations for external entities expressed in the inputs. This feature extraction is applied hierarchically, leading to increasing levels of abstraction at higher layers in the hierarchy.// In collaboration with Mikko Lipasti, University of Wisconsin at Madison, WI, USA, we introduce in [88] a behavioral model for this process, using biologically-plausible neuron-level behavior and structure, and illustrates it with an image recognition task. We also introduce a computationally-effective higher-order modelÅ one that representsthe behavior of hundreds of neurons in a cortical column using just two perceptronsÅ is shown to be capable of this same task. These models are a first step towards developing a comprehensive and biologically-plausible understanding of the computational algorithms and microarchitecture of computing systems that mimic the human neocortex.

5.3.4. Spatial complexity of reversible computing

Participants: Mouad Bahi, Christine Eisenbeis.

Especially since the work of Bennett about reversibility of computation and how to make a computation reversible, the relationship between reversibility, energy, computation and space complexity has gained interest in a lot of domains in computer science. This direction could help us understanding physical limitations of processors performance. We have chosen to start by studying the space complexity of a DAG computation, defined as the maximum number of registers needed for performing the computation in both directions. This criteria is closely related to our more classical criterion of "register saturation". We have defined heuristics for computing this number and have performed systematic experiments on all possible graphs of given size. The first experiments tend to show that for a graph of size n, no more that $n/2$ registers are needed to perform the computations in both directions compared to the forward direction. This latter number can be considered as the "garbage" of the computation. More work is needed to prove/disprove this result more formally and understand the hypothesis in which it is valid [37]. In this work, all operations in the DAG are assumed to be reversible. See also [36].

5.3.5. Rematerialization-based register allocation through reverse computing

Participants: Mouad Bahi, Christine Eisenbeis.

Reversible computing aims at keeping all information on input and intermediate values available at any step of the computation. Rematerialization in register allocation is an alternate solution to spilling where values are recomputed from available data instead of held in registers. In this paper we analyze how register pressure is impacted by rematerialization through reverse operations. We propose an algorithm for rematerialization with reverse computing and we use the memory demanding LQCD (Lattice Quantum ChromoDynamics) application to demonstrate that important gains of up to 33% on register pressure can be obtained. This in turn enables an increase in Instruction-Level Parallelism or in Thread-Level Parallelism. We demonstrate a 30% (statically timed) gain on a basic LQCD computation [22].

5.3.6. Self Developing System for Programming Computing Media

Participants: Frédéric Gruau, Luidnel Maignan, Christine Eisenbeis.

We are targeting high performance, general purpose programming of uniform spatial computing media. Such a medium is a homogeneous assembly of computing units with local neighbour-to-neighbour communication. On one hand, spatial locality enables to consider very large media, on the other hand, it turns the programming into a more difficult task. We are proposing Self Developing Systems (SDS) as new solution for this programming problem: An SDS is programmed by a Finite State Automaton (FSA) with specific output actions acting on the automata itself, that let an initial automaton duplicate and create links, thus developing a network of automata. When implemented in a distributed manner, self development behaves as a distributed

operating system, managing the placement of light threads (FSA) and communication links, as they duplicate or as they are deleted. Thus, the SDS transforms the spatial computing medium into a higher level, virtual machine, which is easier to program because placement is handled at the hardware level. The difficult task in this scenario, is the distributed implementation of self-development, which is the focus or our research. We consider cellular automata as a first step, while keeping in mind other less regular spatial media, such as amorphous computers. The implementation involves several new distributed algorithms. In particular, we developed a distributed algorithm to homogeneize the placement of a cloud of particles, which is both a new problem for cellular automata, and a key ingredient for self-placement. In the context of his Phd Thesis [12], Luidnel Maignan has worked out a distributed algorithms constructing a minimal connected proximity graph establishing connection between nearest particles. His work approach is to consider the metric space underlying the cellular automata topology and construct generic mathematical object based solely on this metric. As a result, the algorithm derived from the properties of those objects, generalize over arbitrary regular grid, including hexagonal, 4 neighbors, and 8 neighbor square grid.

6. Contracts and Grants with Industry

6.1. Collaborations involving industry

Thales TRT Collaboration with Thales TRT, and the CNRS-Thales lab on several topics: customization, simulation, design-space exploration, heterogeneous systems programming, memristors. As mentioned before, the research work on customization recently led to a joint patent application. Main contact: Sami Yehia.

STMicroelectronics Collaboration with STMicroelectronics on program parallelization and architecture support for parallelization.

Philips Semiconductors, now NXP We have had regular collaborations with Philips for almost 10 years now, including direct contracts. Currently, we are involved in several grants with Philips (IP SARC, Marie-Curie fellowships, ACOTES). Philips Semiconductors has recently become NXP.

6.2. National and international collaborative grants

- "PAGDEG" (Causes and consequences of protein aggregation in cellular degeneration): an ANR-funded project (call Piribio) on modeling and simulation of cellular degeneration in bacteria (2010-2012). Supervisor: A. Lindner. Total amount funded: 450 keuros.
- Large-Scale initiative "ColAge" (Natural and engineering solutions to the control of bacterial growth and aging: A systems and synthetic biology approach): an INRIA-INSERM joint grant on modeling and simulation of systems biology (2008-2011). Supervisor: H. Berry. Total amount funded: 430 keuros.

Arch^{2}Neu (150kEuros) This project aims at designing a novel type of hardware for digital signal processing (sounds, images,...) based on analog neural networks. This design shall be significantly more defect and fault tolerant than previous designs, while achieving very low power. This project is a joint INRIA ALCHEMY/CEA LETI ANR project as part of the "Return of PostDoc": we have attracted a young French postdoc at University of California, originally from Supelec, to come back to France and set up this new project (2009-2012).

ARC MACACC (20 keuros): "Modeling Cortical Activity and Analysing the Brain Neural Code", Supervision: B. Cessac (Institut Non Linéaire de Nice). Other partners: Cortex (INRIA Nancy), Institut des Neurosciences cognitives de la Méditerannée (Marseille), Lab. Jean-Alexandre Dieudonné (Nice), Odyssee (INRIA Sophia).

ARC AMYBIA (20 keuros): "Aggregating MYriads of Biologically-Inspired Agents", Supervision: N. Fates (Maia, INRIA NAncy). Other Participants: B. Girau (Cortex, INRIA Nancy).

PEPS-STI CNRS MARTINE (5 Keuros): "Multifractal Analysis to Resolve information Transfer In NEural networks", Supervision: M. Quoy (ETIS, ENSEA, U. Cergy-Pontoise). Other Participants: F. Germinet (AGP, U. Cergy-Pontoise).

Appel à Idées 2008 de l'ISC-PIF (4 keuros): "Organization of a conference on spatial/amorphous computing", Supervision: H. Berry, Other Participants: F. Gruau (ALCHEMY), O. Michel, J.L. Giavitto (Ibisc, U. Evry).

"Action d'Envergure" ColAge : an INRIA-INSERM joint grant (3 years) on modeling and simulation of systems biology (official start Feb. 2009). Supervisor: H. Berry. Total amount funded (for 2008): 41 keuros.

GGCC: EU, MEDEA+ program ITEA Call 8 project on global analysis and optimization in GCC. Our involvment lie in the compiler infrastructure, static analysis in the polyhedral model, and feature extraction for global and contiunous optimization. With CEA (dpt. of energy), UPM (Spain), SICS (Sweden), major industrial partners (Airbus, Telefonica, Bertin) and SMEs (Mandriva, MySQL, and others). 04/2006–04/2009.

PetaQCD: ANR project on the design of architecture, software tools and algorithms for Lattice Quantum Chromodynamics. With Lal (Orsay), Irisa Rennes (Caps/Alf), IRFU (CEA Saclay), LPT (Orsay), Caps Entreprise (Rennes), Kerlabs (Rennes), LPSC (Grenoble).

PARA: French Ministry of Research ANR CIGC project on multi-level parallel programming and automatic parallelization. We are involved in automatic code generation approaches for domain-specific and target-specific optimizations; iterative and polyhedral compilation methods are explored in an application-specific context. With Bull, University of Versailles, LaBRI (University of Bordeaux), INT (Evry), CAPS Entreprise (Rennes). 01/2006–01/2009.

SARC: EU, IST program FP6 FET Proactive IP on advanced computer architecture. The goal is to address all the aspects of a scalable processor architecture based on multi-cores. It includes programming paradigms, compiler optimization, hardware support and simulation issues. CAPSULE is being used as component-based programming approach, and UNISIM for the simulation platform. 01/2006–01/2010.

Embedded TeraOps A SYSTEMATIC "Pôle de Competitivité" regional funding for the development of a large-scale embedded multi-core architectures, coordinated by Thales. It will initially focus on streaming applications but it will later target programs with more complex control flow. Thales, Dassault, Thomson, CEA, INRIA. 01/2006–01/2010.

NoE HiPEAC and HiPEAC2 HiPEAC is a network of excellence on High-Performance Embedded Architectures and Compilers. It involves more than 70 European researchers from 10 countries and 6 companies, including ST, Infineon and ARM. The goal of HiPEAC is to steer European research on future processor architectures and compilers to key issues, relevant to the European embedded industry.

The HiPEAC consortium has submitted a second edition of the network, which has started officially since November 2007 and for four years again. Olivier Temam is a member of the steering committee. 09/2004–11/2011.

Arch2Neu The goal of this project is the design of a hybrid analog/digital chip for the energy efficient and defect-tolerant implementation of signal processing tasks, using analog spiking neurons. The chip will be effectively implemented by the end of the project. 2010–2012.

7. Other Grants and Activities

7.1. Informal collaborations

Cédric Bastoul collaborates with Sébastien Salva from Clermont 1 University and Clément Delamare from Direction Générale des Impôts on web service client parallelization. He collaborates with various people at Reservoir Labs Inc. (New York) on high-level compilation for multicore architectures [41], [103].

Denis Barthou collaborates with these people.

- W. Jalby, Univ. of Versailles St Quentin, PRISM lab.
- S. Louise, CEA/Lastre.
- S. Rajopadhye, U. of Colorado, Etat-Unis.

Grigori Fursin collaborates with the following reseachers:

- Michael O'Boyle, University of Edinburgh, UK
- Chengyong Wu, ICT, China
- Nacho Navarro and Marisa Gil, UPC, Spain
- Mircea Namolaru, Ayal Zaks, Bilha Mendelson, IBM Haifa, Israel
- Francois Bodin, CAPS Entreprise/IRISA, France

Olivier Temam collaborates with these people.

- Mikko Lipasti (University of Wisconsin).
- Kathryn McKinley (University of Texas).
- Veerle Desmet, Lieven Eeckhout (Ghent University).
- Chengyong Wu (ICT, Beijing, China)
- Daniel Gracia-Perez, Gilles Mouchard (CEA LIST).
- Sylvain Girbal, Sami Yehia (Thales TRT).
- Bruno Jego (ST).

ICT Collaboration with Prof. Chengyong Wu at ICT, China, on machine-learning techniques for compilers and data centers.

University of Wisconsin Collaboration with Mikko Lipasti, University of Wisconsin, on bio-inspired architectures.

University of Texas Collaboration with Kathryn McKinley at University of Texas, Austin, on a novel component-based programming approaches for heterogeneous and homogeneous computing systems.

Ghent University and Thales Collaboration with Veerle Desmet at Ghent University, Belgium, on design-space exploration. As part of this collaboration, we recently set up the `www.archexplorer.org` web site and related project.

University of California Santa Cruz Thanks to a France-Berkeley travel grant, We are starting a collaboration with the group of Jose Renau, thanks to a 2006-2007 France-Berkeley grant. The topics are close to the infrastructure work of ALCHEMY: fast and accurate simulation of multi-core processors, and support for a modern parallelisation infrastructure in GCC. Jose Renau is a member of the OpenSparc consortium and contributed to major advances in architecture and compiler support for thread-level speculation.

University of Edinburgh For the past 3 years, we had a very active cooperation with University of Edinburgh on iterative optimization; Grigori Fursin, got his PhD from University Edinburgh. This collaboration has resulted in a series of joint articles [72], [50], [73].

University of Illinois We have a regular collaboration with the group of David Padua, Urbana-Champaign, Illinois, which started 6 years ago, with multiple joint publications and travel grants (CNRS-UIUC). Research focused on high-performance Java, dependence and alias analysis, processors in memory, and currently on adaptive program generation and machine learning compilers.

Texas A&M University We started a regular exchange of ideas and personnel with the Parasol laboratory, led by Lawrence Rauchwerger, a reference in parallel language compilation and architecture support. ProfṘauchwerger visited ALCHEMY for a total of 5 months in the last 3 years, and many of us visited TAMU for shorter periods. The collaboration led to numerous advances in the understanding of the main challenges and pitfals in scalable parallel processing, and also facilites the organization of multiple academic events (e.g., the upcoming PACT'07)

Ohio State University We have a regular collaboration with the group of Prof. Sadayappan, Columbus, Ohio. Recently, we also started to publish together. We invited Uday Bondhugula, PhD student from Ohio for two months, and a Louis-Noël Pouchet will start a postdoc in Ohio in January 2010. The collaboration focuses on polyhedral compilation and new approaches to loop tiling for automatic parallelization.

Louisiana State University We have a regular collaboration with the group of Prof. Ramanujam, Baton Rouge, Louisiana. Recently, we also started to publish together. Mohammed Fellahi was scheduled to spend a 3 month internship in Baton Rouge in 2009, but our plans were cancelled because of difficulties to get a US visa. The collaboration focuses on code generation for polyhedral transformations, and automatic parallelization for GPUs.

UPC We have a regular collaboration with UPC, Barcelona, which started 7 years ago, with several groups on topics ranging from program optimization to micro-architecture, resulting in several publications, joint contracts.

University of Passau We have a regular collaboration with the group of Christian Lengauer and Martin Griebl, Passau, Germany, which started 10 years ago, with multiple joint publications and travel grants (Procope, Ministry of Foreign Affairs). Our collaboration focused on polyhedral compilation techniques and recently headed towards domain-specific program generation and metaprogramming.

Lal-LPT, University of Paris Sud We have started a collaboration with physicists working on LQCD (Lattice Quantic Chromo Dynamics). We focus on the next generation of computer that would gain an order of magnitude speedup over their current APE-next processor (sustained 300 GFlops).

Paris 6 University The properties of biological neural networks that are of direct interest to architecture research are in part due to the intrinsic properties of the individual neurons. We are collaborating with the neuroscience research lab ANIM (INSERM U742) to develop simulation and modelling studies of specific properties of individual biological neurons such as time handling or plasticity and memory properties [75].

CEA List For the past 6 years, we had a regular collaboration with the *Laboratoire SÃ»reté du Logiciel* (Software Safety Lab) at CEA LIST on two topics: processor simulation and program optimization. Simulation of complex processor architectures is necessary for the development of software test of complex systems investigated at CEA. Program optimization is more a way to factor in the CEA expertise in static analysis and develop new applications. CEA has funded two scholarships in our group until 2004 and 2005 respectively.

Others We also have regular contacts with several foreign research groups: the CAPSL group at University of Delaware; and the PASCAL group at University of California Irvine (NSF-INRIA grant).

8. Dissemination

8.1. Leadership within scientific community

Cédric Bastoul

- Member of the *Conseil Consultatif des Spécialistes de l'université Paris-Sud 11* since 2010.

- Member of the *Commission Mixte des Spécialistes* pour l'IUT d'Orsay from 2006 to 2009.
- Member of the *Conseil de laboratoire LRI* since 2006.

Albert Cohen

Christine Eisenbeis

- Member of IFIP WG 10.3.
- Member of the "comité de programmes" of Digiteo.
- Member of the "conseil de gouvernance" of Polytech Paris Sud (école d'ingénieurs de l'University of Paris-Sud 11).
- Elected member of the "conseil d'administration"' of Inria [2006-2010].
- Elected member of the "conseil scientifique" of University of Paris-Sud 11 [2008-].
- Chair [2008-] of the "commission des utilisateurs des moyens informatiques - recherche" of the Saclay Inria Research Center.

Olivier Temam

- HiPEAC2 Steering Committee, Research workpackage leader, leader of the Research Cluster on simulation.
- Program Co-Chair of the 2011 International Conference on High-Performance and Embedded Systems (HiPEAC).
- General Chair of the 2011 ACM/IEEE International Symposium on Code Generation and Optimization (CGO), to be organized in Chamonix, France. It is the first time that CGO will be held outside the US.
- Leader of the INRIA ALCHEMY group.

PROGRAM COMMITTEES:

Denis Barthou

Cédric Bastoul

2011: General co-chair of the International Workshop on Polyhedral Compilation Techniques (IMPACT).

2011: Program Committee member of the International Conference on Computing Frontiers (CF).

2010: Program Committee member of the International Symposium on Stabilization, Safety and Security (SSS).

2009: Program Committee member of the International Conference on Design Automation and Test (DATE).

Albert Cohen

Grigori Fursin

Olivier Temam

2011: Program Committee member of the International Symposium on Computer Architecture (ISCA)

2011: General Chair of the International Symposium on Code Generation and Optimization (CGO)

2011: Program Chair of the International Conference on High Performance Embedded Architectures and Compilers (HiPEAC)

2010-: Member of the "Bureau du Comite des Projets", INRIA Saclay

2010-: Member of the Scientific Advisory Board of Thales

2010: Program Committee member of the International Symposium on Microarchitecture (MICRO)

2010: Program Committee member of the International Conference on Architecture of Computing Systems (ARCS)

2010: Member of the ACM SIGARCH Nominating Committee

2010: Program Committee member of the International Symposium on Computer Architecture (ISCA)

2009-2013: Member of the advisory panel of UK project "Biologically-Inspired Massively Parallel Architectures - computing beyond a million processors", coordinated by Steve Furber, University of Manchester, 5-million Euros grant

2009-: Organizer of the HiPEAC cluster on New Technologies and New Architecture Paradigms

2009, 2010: Program Committee member of the International Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools (RAPIDO)

8.2. Teaching at university

Denis Barthou gave these courses:

- 15h in Master2, UVSQ on vectorization/parallelization,
- Summer School INRIA/CEA/EDF on High Performance Computing (june 2008).

Cédric Bastoul gives Java, System, Network and Security lectures and labs at the Orsay Institute of Technology to first, second and third year students (L1 to L3). He also teaches a Object Oriented Programming course at Paris-Sud University to second year students (L2). Lastly, he is teaching computer architecture at École Polytechnique, for third year students (M1).

Anna Beletska gave 9 hours of lectures in the Master 2 “Recherche” of Computer Science of University of Paris-Sud 11.

Mohamed-Walid Benabderrahmane: Monitorat at IFIPS - University Paris-sud 11, Courses: C/C++/C# , Web Services, Security, Level: 5 year engineer.

Philippe Dumont: Components of a Computing System, Introduction to Computer Architecture and Operating Systems, École Polytechnique - Licence 3 - 36h

Christine Eisenbeis gave a 3 hours lecture about “Reversible computing” in the Master 2 “Recherche” of Computer Science of University of Paris-Sud 11.

Olivier Temam teaches a computer architecture course at École Polytechnique to 3rd-year students on computer architectures (appr. 35 hours). He also co-teaches a course on novel processor architectures at University of Paris Sud to Master’s students.

Albert Cohen teaches an introductory computing systems (computer architecture, operating systems, distributed systems) at École Polytechnique to 2nd-year students (appr 35 hours, 120 students); it was the first course using the Google Android development kit as a virtual platform for lab sessions; an e-book published with Eyrolles came out of this first experiment in 2009. He also teaches an advanced operating systems course to 3rd-year students at École Polytechnique. He also co-chairs the Electrical Engineering curriculum at École Polytechnique.

8.3. Workshops, seminars, invitations

The project-team members have given the following talks and attended the following conferences:

Mounira Bachir

- LCPC 09, University of Delaware, USA, October 8-10, 2009, "Using The Meeting Graph Framework to Minimise Kernel Loop Unrolling for Scheduled Loops"

Cédric Bastoul

- Participation to the International Conference on Supercomputing (SC'10), New Orleans, USA, November 2010.
- Invited presentation at the Computer Science Colloquium of the University of Innsbruck, Austria, September 2010.
- Paper presentation at PMEA 2009 (September, Raleigh, North Carolina) Workshop on Programming Models for Emerging Architectures in conjunction with PACT 2009.
- Poster presentation at PACT 2009 (September, Raleigh, North Carolina) Intl. Conf. on Parallel Architectures and Compilation Techniques.
- Participation to SPC 2009 Fault-Tolerant Spaceborne Computing Employing New Technologies Workshop (May 26-29, Albuquerque, New Mexico).

Anna Beletska

- Cocoa' 2009, talk "Computing the transitive closure of a union of affine integer tuple relations"
- ISPDC 2009, talk "Coarse-Grained Loop Parallelization: Iteration Space Slicing vs Affine Transformations"

Mohamed-Walid Benabderrahmane

- Poster Pact 2009, "A Conservative Approach to Manipulate Data-Dependent Control Flow in the Polyhedral Model", with Louis-Noël Pouchet
- Summer school : Acaces 2009, Fifth International Summer School on Advanced Computer Architecture and Compilation for Embedded Systems July 12 to July 18, 2009 Terrassa (near Barcelona), Spain

Hugues Berry

- "The Effects of Hebbian Learning on the Structure and Dynamics of Chaotic Neural Networks", given at the Dept. Electrical and Computer Enginerring, Univ. Wisconsin at Madison, WI, USA, Jan. 13, 2009 (invited by M. Lipasti).
- "Estimating the effects of intrinsic plasticity on neural network dynamics using a realistic model", at the "Journees Mathematiques du Vivant", Laboratory J.A. Dieudonnee, Nice, France, March 25, 2009 (invited by B. Cessac)
- "ColAge: A systems and synthetic biology approach to the control of bacterial growth and aging", the 2nd NIH-INRIA workshop on Biomedical Computing, INRIA Rocquencourt, France, June 3, 2009.
- "Cell biochemistry in cytoplasms with large molecular crowding : anomalous diffusion and bacterial aging", at the 2nd Paris Workshop on Multi-Agent Systems in Biology at the Meso or Macroscopic Scales, Univ. Pierre et Marie Curie, Paris, France, June 23, 2009 (invited by M. Beurton-Aimar)

Christine Eisenbeis

- Meeting QPACE - AURORA - PetaQCD, Regensburg, April 14-15, 2010, talk on "PetaQCD Introduction and Summary".
- PetaQCD meeting, "Status of the PetaQCD project", May 10-11, 2010.
- International Conference on Parallel Processing (ICPP 2010), San Diego, September 13-16, 2010, talk on "Speculative Execution on GPU: An Exploratory Study" and talk on "A Theoretical Framework for Value Prediction in Parallel Systems".

Grigori Fursin

- Participation to MICRO'09 (42nd IEEE/ACM International Symposium on Microarchitecture), New York, USA, December 2009
- invited talk, "Collective Tuning Initiative", presented at the University of Versailles, France, May 2009; presented at the HiPEAC industrial workshop and HiPEAC clusters, Infineon, Munich, Germany, June 2009;
- paper presentation "Collective Tuning" at the GCC Summit'09, Montreal, Canada, June 2009;
- invited talk, "Collective Tuning Initiative: collective optimization, run-time adaptation and machine learning", presented at University of Illinois at Urbana Champaign, USA, April 2009
- paper presentation "Collective Optimization" at HiPEAC'09, Cyprus, January 2009
- paper presentation "Finding representative sets of optimizations for adaptive multiversioning applications" at SMART'09, Cyprus, January 2009
- invited talk (by EU FP7 commision), "MILEPOST project - using machine learning to automate and speed up program optimization for reconfigurable processors", presented at the Information and Brokerage Conference on Information and Communication Technologies in the EU's 7th Framework, Moscow, Russia, October 2008
- invited talk, "Enabling Dynamic Optimization and Adaptation for Statically Compiled Programs Using Function Multi-Versioning", presented at ScalPerf'08 (Scalable Approaches to High Performance and High Productivity Computing), Bertinoro, Italy, September 2008
- invited talk, "Continuous adaptive program optimizations", presented at Reservoir Labs and IBM TJ Watson Research Center, New York, USA, August 2008;
  presented at Imperial College (Software Performance Engineering Laboratory), London, UK, February 2008;
  presented at the Institute of Computing Technology (Chinese Academy of Sciences), Beijing, China, January 2008;
- invited talk, "Program iterative continuous optimizations, run-time adaptation and machine learning", presented at IBM Toronto Lab (compiler group), Canada, July 2007;
- invited talk, "Machine learning techniques for iterative program optimizations and run-time adaptation", presented for the TAO group (machine learning group), LRI, Paris-Sud XI University, INRIA and CNRS, France, June 2007;
- invited talk, "Overview of current activities: Interactive Compilation Interface for fine-grain program optimizations, dataset sensitivity, machine learning to speed up optimizations and DSE, run-time program adaptation, optimizations for heterogeneous computing systems, continuous collective optimizations, HiPEAC activities", presented at Intel (compiler group), Moscow, Russia, February 2007 and at the ISP RAS (Institute for System Programming, Russian Academy of Sciences), Moscow, Russia, February 2007
- "Continuous run-time adaptation and optimization of statically compiled programs", presented at the UPC, Barcelona, Spain, January 2007.

Albert Cohen

- Seminar at the U. of Delaware, February 2009, Newark DE: “state of the art in polyhedral compilation for production compilers”.
- Visit of Reservoir Labs, February 2009, New York.
- Visit of the group of Markus Püschel and Franz Franchetti, Carnegie Mellon University, February 2009, Pittsburgh, Pennsylvania.
- Visit of the group of Kathryn O’Brien, of Kenneth Zadeck and David Edelsohn at IBM Research Watson, June 2009, Yorkton Heights, New York.
- Invited presentation and contribution to a planning meeting for a future European call for research proposals on system- and process-level virtualization, September 2009, Bruxelles.
- Presentation at the second STMicroelectronics-INRIA Plateform2012 meeting, October 2009, Grenoble.
- Invited panelist at the LCPC’09 Panel on the future of compilation research and technology, October 2009, Newark, Delaware.
- Co-organizer (with Joseph Sifakis, Ahmed Jerraya and BenoÃ®t Dupont de Dinechin) of the ESWeek’09 Industrial Panel on compilers for embedded multicore architectures, October 2009, Grenoble.
- Presentation at Dagstuhl Seminar 09481 (SYNCHRON’09), December 2009: “A data-flow synchronous perspective to performance portability”.
- Seminar at U. Saarbrücken, December 2009: “Languages and compilers for Volkscomputing”.
- Seminar at U. Passau, December 2009: “Language and compilers for Volkscomputing”.
- Presentation at the second Bull-INRIA-CEA partnership meeting, December 2009, Rocquencourt.

Philippe Dumont

- Workshop on PetaScale Computing, First workshop of INRIA-Illinois Petascale Computing Joint Lab, June 10 to June 12, 2009, Paris, France
- Acaces Summer School, July 12 to July 18, 2009, Terrassa, Spain
- “ERBIUM: A Deterministic, Low-Level Concurrent Representation for Portability and Scalable Performance”, Synchronics day, December 17 2009, Paris, France

Sean Halle

- Poster at ACACES 2009 International Conference “Bidirectional Libraries for Portable High Performance Parallelism

9. Bibliography

Major publications by the team in recent years

[1] F. AGAKOV, E. BONILLA, J. CAVAZOS, B. FRANKE, G. FURSIN, M. O’BOYLE, J. THOMSON, M. TOUSSAINT, C. WILLIAMS. *Using Machine Learning to Focus Iterative Optimization*, in "Proceedings of the 4th Annual International Symposium on Code Generation and Optimization (CGO)", 2006.

[2] H. BERRY, D. GRACIA PÉREZ, O. TEMAM. *Chaos in computer performance*, in "Chaos", 2006, vol. 16, 013110, http://hal.inria.fr/inria-00000109/en/.

[3] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *N-Sychronous Kahn Networks*, in "33th ACM Symp. on Principles of Programming Languages (PoPL'06)", Charleston, South Carolina, January 2006, p. 180–193, http://www-rocq.inria.fr/~acohen/publications/CDEPPP06.ps.gz.

[4] A. COHEN, S. GIRBAL, O. TEMAM. *A Polyhedral Approach to Ease the Composition of Program Transformations*, in "Euro-Par'04", Pisa, Italy, LNCS, Springer-Verlag, August 2004, n° 3149, p. 292–303, http://www-rocq.inria.fr/~acohen/publications/CGT04.ps.gz.

[5] D. GRACIA PÉREZ, G. MOUCHARD, O. TEMAM. *MicroLib: A Case for the Quantitative Comparison of Micro-Architecture Mechanisms*, in "MICRO-37: Proceedings of the 37th International Symposium on Microarchitecture", IEEE Computer Society, Dec 2004, p. 43–54 [*DOI :* 10.1109/MICRO.2004.25].

[6] G. GROSDIDIER, C. EISENBEIS, F. BODIN, A. SEZNEC, R. BILHAUT, G. LE MEUR, P. ROUDEAU, F. TOUZE, J.-C. ANGLÈS D'AURIAC, J. CARBONELL, D. BECIREVIC, P. BOUCAUD, O. BRAND-FOISSAC, O. PENE, D. BARTHOU, P. GUICHON, P. HONORE, P. GALLARD, L. RILLING. *The PetaQCD project*, in "17th International Conference on Computing in High Energy and Nuclear Physics (CHEP09)", Prague Tchèque, République, 03 2009, The proceedings of the International Conference on Computing in High Energy and Nuclear Physics (CHEP 2009) will be published in the open access Journal of Physics: Conference Series (JPCS), published by IOP Publishing. All papers will be free to read and download immediately upon publication. LAL 09-58, http://hal.in2p3.fr/in2p3-00380246/en/.

[7] F. GRUAU, Y. LHUILLIER, P. REITZ, O. TEMAM. *Blob Computing*, in "Computing Frontiers 2004 ACM SIGMicro.", 2004, http://blob.lri.fr/publication/2004-model-blob-machine.pdf.

[8] P. PALATIN, Y. LHUILLIER, O. TEMAM. *Capsule : Hardware-Assisted Parallel Execution of Component-Based Programs*, in "The 39th Annual IEEE/ACM International Symposium on Microarchitecture, 2006", Orlando, Florida, december 2006.

[9] D. PARELLO, O. TEMAM, J.-M. VERDUN. *On increasing architecture awareness in program optimizations to bridge the gap between peak and sustained processor performance : Matrix-Multiply revisited*, in "Supercomputing", IEEE, Nov 2002.

[10] S. POP, A. COHEN, G.-A. SILBER. *Induction Variable Analysis with Delayed Abstractions*, in "Intl. Conf. on High Performance Embedded Architectures and Compilers (HiPEAC'05)", Barcelona, Spain, LNCS, Springer-Verlag, November 2005, n° 3793, p. 218–232, http://www-rocq.inria.fr/~acohen/publications/PCS05.ps.gz.

[11] N. VASILACHE, C. BASTOUL, S. GIRBAL, A. COHEN. *Violated dependence analysis*, in "Proceedings of the ACM International Conference on Supercomputing (ICS'06)", Cairns, Australia, ACM, June 2006.

Publications of the year

Doctoral Dissertations and Habilitation Theses

[12] L. MAIGNAN. *Points, Distances, and Cellular Automata: Geometric and Spatial Algorithmics*, Université Paris-Sud 11, France, 8 décembre 2010.

Articles in International Peer-Reviewed Journal

[13] J.-C. ANGLÈS D'AURIAC, D. BARTHOU, D. BECIREVIC, R. BILHAUT, F. BODIN, P. BOUCAUD, O. BRAND-FOISSAC, J. CARBONELL, C. EISENBEIS, P. GALLARD, G. GROSDIDIER, P. GUICHON, P.-F. HONORÉ, G. L. MEUR, O. PÈNE, L. RILLING, P. ROUDEAU, A. SEZNEC, A. STOCCHI, F. TOUZE. *Towards the Petaflop for Lattice QCD simulations the PetaQCD project*, in "Journal of Physics: Conference Series", 2010, vol. 219, n^{O} 5, 052021, This work was presented at the 17th International Conference on Computing in High Energy and Nuclear Physics (CHEP09), http://stacks.iop.org/1742-6596/219/i=5/a=052021.

[14] S. LIU, C. EISENBEIS, J.-L. GAUDIOT. *Value Prediction and Speculative Execution on GPU*, in "International Journal of Parallel Programming", 2010, p. 1-20, 10.1007/s10766-010-0155-0, http://dx.doi.org/10.1007/s10766-010-0155-0.

[15] H. MUNK, E. AYGUADÉ, C. BASTOUL, P. CARPENTER, Z. CHAMSKI, A. COHEN, M. CORNERO, P. DUMONT, M. DURANTON, M. FELLAHI, R. FERRER, R. LADELSKY, M. LINDWER, X. MARTORELL, C. MIRANDA, D. NUZMAN, A. ORNSTEIN, A. POP, S. POP, L.-N. POUCHET, A. RAMÍREZ, D. RÓDENAS, E. ROHOU, I. ROSEN, U. SHVADRON, K. TRIFUNOVIC, A. ZAKS. *ACOTES Project: Advanced Compiler Technologies for Embedded Streaming*, in "International Journal of Parallel Programming", Apr 2010, p. 1-54 [*DOI :* 10.1007/s10766-010-0132-7], http://hal.archives-ouvertes.fr/inria-00551083/en/.

International Peer-Reviewed Conference/Proceedings

[16] A. COHEN, E. ROHOU. *Processor Virtualization and Split Compilation for Heterogeneous Multicore Embedded Systems*, in "47th Annual Design Automation Conference", États-Unis Anaheim, CA, 2010, http://hal.inria.fr/inria-00472274/en.

[17] N. FATÈS, H. BERRY. *Robustness of the critical behaviour in a discrete stochastic reaction-diffusion medium*, in "IWNC'09", Japon Himeji, Y. SUZUKI, M. HAGIYA, H. UMEO, A. ADAMATZKY (editors), Proceedings in Information and Communications Technology, Springer, 2010, http://hal.inria.fr/inria-00396473/en.

[18] Y. HUANG, L. PENG, C. WU, Y. KASHNIKOV, J. RENNECKE, G. FURSIN. *Transforming GCC into a research-friendly environment: plugins for optimization tuning and reordering, function cloning and program instrumentation*, in "2nd International Workshop on GCC Research Opportunities (GROW'10)", Italie Pisa, Jan 2010, http://hal.inria.fr/inria-00451106/en.

[19] S. LIU, C. EISENBEIS, J.-L. GAUDIOT. *A Theoretical Framework for Value Prediction in Parallel Systems*, in "Parallel Processing, International Conference on", Los Alamitos, CA, USA, IEEE Computer Society, 2010, vol. 0, p. 11-20 [*DOI :* 10.1109/ICPP.2010.10].

[20] S. LIU, C. EISENBEIS, J.-L. GAUDIOT. *Speculative Execution on GPU: An Exploratory Study*, in "Parallel Processing, International Conference on", Los Alamitos, CA, USA, IEEE Computer Society, 2010, vol. 0, p. 453-461 [*DOI :* 10.1109/ICPP.2010.53].

Workshops without Proceedings

[21] R. BAGHDADI, A. COHEN, C. BASTOUL, L.-N. POUCHET, L. RAUCHWERGER. *The Potential of Synergistic Static, Dynamic and Speculative Loop Nest Optimizations for Automatic Parallelization*, in "Pespma 2010 - Workshop on Parallel Execution of Sequential Programs on Multi-core Architecture", France Saint Malo, W. LIU, S. MAHLKE, TIN-FOOK. NGAI (editors), 2010, http://hal.inria.fr/inria-00494305/en.

Research Reports

[22] M. BAHI, C. EISENBEIS. *Rematerialization-based register allocation through reverse computing*, 2011, Soumis.

[23] S. BRIAIS, S.-A.-A. TOUATI, K. DESCHINKEL. *Ensuring Lexicographic-Positive Data Dependence Graphs in the SIRA Framework*, Mar 2010, http://hal.inria.fr/inria-00452695/en.

[24] A. MAZOUZ, S.-A.-A. TOUATI, D. BARTHOU. *Measuring and Analysing the Variations of Program Execution Times on Multicore Platforms: Case Study*, Sep 2010, http://hal.inria.fr/inria-00514548/en.

[25] S.-A.-A. TOUATI, J. WORMS, S. BRIAIS. *The Speedup Test*, 2010, http://hal.inria.fr/inria-00443839/en.

Other Publications

[26] A. COHEN. *Automatic Parallelization in GCC: for Research and for Real (Keynote Talk)*, Jun 2010, Type : Short paper, http://hal.inria.fr/inria-00494301/en.

[27] O. TEMAM. *The Rebirth of Neural Networks*, Jun 2010, Type : Presentation, http://hal.inria.fr/inria-00535554/en.

References in notes

[28] *FLEXUS*, http://www.ece.cmu.edu/~simflex/flexus.html.

[29] *GCC ICI: Interactive Compilation Interface*, http://gcc-ici.sourceforge.net.

[30] *SystemC v2.0.1 Language Reference Manual*, 2003, http://www.systemc.org/.

[31] *UNISIM: UNIted SIMulation environment*, http://unisim.org.

[32] F. AGAKOV, E. BONILLA, J. CAVAZOS, B. FRANKE, G. FURSIN, M. F. P. O'BOYLE, J. THOMSON, M. TOUSSAINT, C. WILLIAMS. *Using Machine Learning to Focus Iterative Optimization*, in "Proceedings of the 4th Annual International Symposium on Code Generation and Optimization (CGO)", 2006.

[33] F. AGAKOV, E. BONILLA, J. CAVAZOS, B. FRANKE, G. FURSIN, M. F. P. O'BOYLE, J. THOMSON, M. TOUSSAINT, C. WILLIAMS. *Using Machine Learning to Focus Iterative Optimization*, in "CGO-4: The Fourth Annual International Symposium on Code Generation and Optimization", 2006.

[34] R. ALLEN, D. CALLAHAN, K. KENNEDY. *Automatic decomposition of scientific programs for parallel execution*, in "Proceedings of the 14th ACM SIGACT-SIGPLAN symposium on Principles of programming languages", ACM Press, 1987, p. 63–76, http://doi.acm.org/10.1145/41625.41631.

[35] D. AURAS, S. GIRBAL, H. BERRY, O. TEMAM, S. YEHIA. *CMA: Chip Multi-Accelerator*, in "IEEE, International Symposium on Application Specific Processors (SASP)", Anaheim, California, US, June 2010.

[36] M. BAHI, C. EISENBEIS. *Spatial complexity of reversibly computable DAG*, in "CASES, International Conference on Compilers, Architecture, and Synthesis for embedded systems", 2009, p. 47-56.

[37] M. BAHI, C. EISENBEIS, B. DAUVERGNE, A. COHEN. *Spatial complexity of reversible computing*, in "Third International Summer School on Advanced Computer Architecture and Compilation for Embedded Systems (ACACES'08)", L'Aquila, Italy, July 2008.

[38] J.-P. BANÂTRE, D. L. MÉTAYER. *Gamma and the Chemical Reaction Model : Ten Years After*, in "Coordination Programming: Mechanisms, Models and Semantics", J.-M. ANDREOLI, H. GALLAIRE, D. L. MÉTAYER (editors), 1996, p. 1–39.

[39] C. BASTOUL. *Code Generation in the Polyhedral Model Is Easier Than You Think*, in "PACT'13 IEEE International Conference on Parallel Architecture and Compilation Techniques", Juan-les-Pins, september 2004, p. 7–16, http://hal.ccsd.cnrs.fr/ccsd-00017260.

[40] C. BASTOUL, A. COHEN, S. GIRBAL, S. SHARMA, O. TEMAM. *Putting Polyhedral Loop Transformations to Work*, in "Workshop on Languages and Compilers for Parallel Computing (LCPC'03)", College Station, Texas, LNCS, Springer-Verlag, October 2003, p. 23–30.

[41] C. BASTOUL, N. VASILACHE, A. LEUNG, B. MEISTER, D. WOHLFORD, R. LETHIN. *Extended Static Control Programs as a Programming Model for Accelerators, A Case Study: Targetting ClearSpeed CSX700 With the R-Stream Compiler*, in "PMEA'09 Workshop on Programming Models for Emerging Architectures", Raleigh, North Carolina, September 2009, p. 45-52.

[42] H. BERRY, D. GRACIA PÉREZ, O. TEMAM. *Chaos in computer performance*, in "Chaos", 2006, vol. 16, 013110, http://hal.inria.fr/inria-00000109/en/.

[43] H. BERRY, D. GRACIA PÉREZ, O. TEMAM. *Complex dynamics of microprocessor performances during program execution: Regularity, chaos, and others*, in "NKS2006 Wolfram Science Conference", Washington D.C., USA, June 2006.

[44] H. BERRY, M. QUOY. *Structure and dynamics of random recurrent neural networks*, in "Adaptive Behavior", 2006, vol. 14, p. 129-137.

[45] H. BERRY, O. TEMAM. *Modeling Self-Developping Biological Neural Network*, in "Neurocomputing", 2007, vol. 70, n^{o} 16-18, p. 2723–2734.

[46] P. BERUBE, J. AMARAL. *Aestimo: a feedback-directed optimization evaluation tool*, in "Proceedings of the International Symposium on Performance Analysis of Systems and Software (ISPASS)", 2006.

[47] N. L. BINKERT, R. G. DRESLINSKI, L. R. HSU, K. T. LIM, A. G. SAIDI, S. K. REINHARDT. *The M5 Simulator: Modeling Networked Systems*, in "IEEE Micro", 2006, vol. 26, n^{o} 4, p. 52–60, http://dx.doi.org/10.1109/MM.2006.82.

[48] R. BLUMOFE, C. JOERG, B. KUSZMAUL, C. LEISERSON, K. RANDALL, Y. ZHOU. *Cilk: An Efficient Multithreaded Runtime System*, in "Proceedings of the 5th Symposium on Principles and Practice of Parallel Programming", 1995, http://supertech.csail.mit.edu/papers/PPoPP95.pdf.

[49] D. BURGER, T. M. AUSTIN. *The SimpleScalar tool set, version 2.0*, in "SIGARCH Comput. Archit. News", 1997, vol. 25, n^{o} 3, p. 13–25, http://doi.acm.org/10.1145/268806.268810.

[50] J. CAVAZOS, C. DUBACH, F. AGAKOV, E. BONILLA, M. F. P. O'BOYLE, G. FURSIN, O. TEMAM. *Automatic Performance Model Construction for the Fast Software Exploration of New Hardware Designs*, in "International Conference on Compilers, Architecture, And Synthesis For Embedded Systems (CASES 2006)", October 2006, To appear.

[51] Z. CHAMSKI, M. DURANTON, A. COHEN, C. EISENBEIS, P. FEAUTRIER, D. GENIUS. *Application Domain-Driven System Design for Pervasive Video Processing*, in "Ambient Intelligence: Impact on Embedded-System Design", Kluwer Academic Press, 2003.

[52] Y. CHEN, Y. HUANG, L. EECKHOUT, L. PENG, G. FURSIN, O. TEMAM, C. WU. *Evaluating Iterative Optimization across 1000 Data Sets*, in "International Conference on Programming Language Design and Implementation (PLDI)", Toronto, Canada, June 2010.

[53] J. CHEN, P. JUANG, K. KO, G. CONTRERAS, D. PENRY, R. RANGAN, A. STOLER, L.-S. PEH, M. MARTONOSI. *Hardware-modulated parallelism in chip multiprocessors*, in "SIGARCH Comput. Archit. News, Special Issue: Proc. of the dasCMP'05 Workshop", 2005, vol. 33, n° 4, p. 54–63, http://doi.acm.org/10.1145/1105734.1105742.

[54] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *Synchronization of Periodic Clocks*, in "ACM Conf. on Embedded Software (EMSOFT'05)", Jersey City, New York, September 2005, http://www-rocq.inria.fr/~acohen/publications/CDEPPP05.ps.gz.

[55] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *N-Sychronous Kahn Networks*, in "33th ACM Symp. on Principles of Programming Languages (PoPL'06)", Charleston, South Carolina, January 2006, p. 180–193, http://www-rocq.inria.fr/~acohen/publications/CDEPPP06.ps.gz.

[56] A. COHEN, D. GENIUS, A. KORTEBI, Z. CHAMSKI, M. DURANTON, P. FEAUTRIER. *Multi-Periodic Process Networks: Prototyping and Verifying Stream-Processing Systems*, in "Euro-Par'02", Paderborn, Germany, LNCS, Springer-Verlag, August 2002, vol. 2400, http://www-rocq.inria.fr/~acohen/publications/CGKCDF02.ps.gz.

[57] A. COHEN, S. GIRBAL, D. PARELLO, M. SIGLER, O. TEMAM, N. VASILACHE. *Facilitating the Search for Compositions of Program Transformations*, in "ACM Intl. Conf. on Supercomputing (ICS'05)", Boston, Massachusetts, June 2005, p. 151–160, http://www-rocq.inria.fr/~acohen/publications/CGPSTV05.ps.gz.

[58] A. COHEN, S. GIRBAL, O. TEMAM. *A Polyhedral Approach to Ease the Composition of Program Transformations*, in "Euro-Par'04", Pisa, Italy, LNCS, Springer-Verlag, August 2004, n° 3149, p. 292–303, http://www-rocq.inria.fr/~acohen/publications/CGT04.ps.gz.

[59] K. D. COOPER, A. GROSUL, T. J. HARVEY, S. REEVES, D. SUBRAMANIAN, L. TORCZON, T. WATERMAN. *ACME: adaptive compilation made efficient*, in "Proceedings of the Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)", 2005, p. 69–77.

[60] K. D. COOPER, D. SUBRAMANIAN, L. TORCZON. *Adaptive Optimizing Compilers for the 21st Century*, in "J. Supercomput.", 2002, vol. 23, n° 1, p. 7–22, http://dx.doi.org/10.1023/A:1015729001611.

[61] L. DAGUM, R. MENON. *OpenMP: An Industry- Standard API for Shared- Memory Programming*, in "IEEE COMPUTATIONAL SCIENCE & ENGINEERING", 1998, p. 46-55.

[62] V. DESMET, S. GIRBAL, O. TEMAM. *A Methodology for Facilitating a Fair Comparison of Research Ideas*, in "IEEE, International Symposium on Performance Analysis of Systems and Software (ISPASS)", White Plains, NY, IEEE Computer Society Press, March 2010.

[63] M. DUPRÉ, N. DRACH, O. TEMAM. *Quickly building an optimizer for complex embedded architectures*, in "International Symposium on Code Generation and Optimization", ACM/IEEE, Mar 2004.

[64] J. S. EMER, P. AHUJA, E. BORCH, A. KLAUSER, C.-K. LUK, S. MANNE, S. S. MUKHERJEE, H. PATIL, S. WALLACE, N. L. BINKERT, R. ESPASA, T. JUAN. *Asim: A Performance Model Framework.*, in "IEEE Computer", 2002, vol. 35, n^{o} 2, p. 68-76.

[65] K. FATAHLIAN, T. J. KNIGHT, M. HOUSTON, M. EREZ, D. R. HORN, L. LEEM, J. Y. PARK, M. REN, A. AIKEN, W. J. DALLY, P. HANRAHAN. *Sequoia: Programming the Memory Hierarchy*, in "Supercomputing 2006", Tampa, Florida, November 2006.

[66] N. FATES, H. BERRY. *Critical phenomena in a discrete stochastic reaction-diffusion medium*, in "Fourth International Workshop on Natural Computing, IWNC 2009", September 2009.

[67] P. FEAUTRIER. *Dataflow Analysis of Array and scalar references*, in "Int. J. of Parallel Programming", 1991, vol. 20, n^{o} 1, p. 23-53.

[68] P. FEAUTRIER. *Some efficient solutions to the affine scheduling problem I. One-dimensional time*, in "Int. J. of Parallel Programming", 1992, vol. 21, n^{o} 5, p. 313-347.

[69] P. FEAUTRIER. *Some efficient solutions to the affine scheduling problem II. Multi-dimensional time*, in "Int. J. of Parallel Programming", 1992, vol. 21, n^{o} 6, p. 389-420.

[70] B. FRANKE, M. F. P. O'BOYLE, J. THOMSON, G. FURSIN. *Probabilistic Source-Level Optimisation of Embedded Programs*, in "Proceedings of the Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES)", 2005.

[71] G. FURSIN, A. COHEN. *Building a Practical Iterative Interactive Compiler*, in "1st Workshop on Statistical and Machine Learning Approaches Applied to Architectures and Compilation (SMART'07), colocated with HiPEAC 2007 conference", Ghent, Belgium, January 2007.

[72] G. FURSIN, A. COHEN, M. O'BOYLE, O. TEMAM. *A Practical Method For Quickly Evaluating Program Optimizations*, in "Intl. Conf. on High Performance Embedded Architectures and Compilers (HiPEAC'05)", Barcelona, Spain, LNCS, Springer-Verlag, November 2005, n^{o} 3793, p. 29–46, http://hal.inria.fr/inria-00001054/en/.

[73] G. FURSIN, A. COHEN, M. O'BOYLE, O. TEMAM. *Quick and practical run-time evaluation of multiple program optimizations*, in "Trans. on High Performance Embedded Architectures and Compilers", 2006, vol. 1, n^{o} 1, p. 13-31.

[74] G. FURSIN, M. F. P. O'BOYLE, P. KNIJNENBURG. *Evaluating Iterative Compilation*, in "Proc. Languages and Compilers for Parallel Computers (LCPC)", 2002, p. 305-315.

[75] S. GENET, B. DELORD, L. SABARLY, E. GUIGON, H. BERRY. *On the propagation of Ca-dependent plateau and valley potentials in cerebellar Purkinje cells and how they drive the cell output*, in "Proceedings of NeuroComp'06", Pont-à-Mousson, France, 23-24 October 2006, p. 167–170.

[76] J.-L. GIAVITTO, O. MICHEL. *MGS: a Rule-Based Programming Language for Complex Objects and Collections*, in "Electronic Notes in Theoretical Computer Science", 2001, vol. 59, n° 4.

[77] S. GIRBAL, O. TEMAM, S. YEHIA, H. BERRY, Z. LI. *A Memory Interface for Multi-Purpose Multi-Stream Accelerators*, in "IEEE, International Conference on Compilers, Architecture, And Synthesis For Embedded Systems (CASES)", Scottsdale, Arizona, October 2010.

[78] S. GIRBAL, N. VASILACHE, C. BASTOUL, A. COHEN, D. PARELLO, M. SIGLER, O. TEMAM. *Semi-Automatic Composition of Loop Transformations for Deep Parallelism and Memory Hierarchies*, in "Intl. J. of Parallel Programming", 2006, Accepted with minor revisions.

[79] S. C. GOLDSTEIN, M. BUDIU. *NanoFabrics: spatial computing using molecular electronics*, in "Proceedings of the 28th annual international symposium on Computer architecture", Göteborg, Sweden, ACM Press, 2001, p. 178–191.

[80] D. GRACIA PÉREZ, H. BERRY, O. TEMAM. *IDDCA: A New Clustering Approach For Sampling*, in "MoBS: Workshop on Modeling, Benchmarking, and Simulation MoBS: Workshop on Modeling, Benchmarking, and Simulation", Madison, Wisconsin, 2005, http://hal.inria.fr/inria-00001062/en/.

[81] D. GRACIA PÉREZ, H. BERRY, O. TEMAM. *Budgeted Region Sampling (BeeRS): Do Not Separate Sampling From Warm-Up, And Then Spend Wisely Your Simulation Budget*, in "5th IEEE International Symposium on Signal Processing and Information Technology 5th IEEE International Symposium on Signal Processing and Information Technology", Athens, Greece, 2006, http://hal.inria.fr/inria-00001061/en/.

[82] D. GRACIA PÉREZ, G. MOUCHARD, O. TEMAM. *MicroLib: A Case for the Quantitative Comparison of Micro-Architecture Mechanisms*, in "MICRO-37: Proceedings of the 37th International Symposium on Microarchitecture", IEEE Computer Society, Dec 2004, p. 43–54, http://dx.doi.org/10.1109/MICRO.2004.25.

[83] F. GRUAU, Y. LHUILLIER, P. REITZ, O. TEMAM. *Blob Computing*, in "Computing Frontiers 2004 ACM SIGMicro.", 2004, http://blob.lri.fr/publication/2004-model-blob-machine.pdf.

[84] M. R. GUTHAUS, J. S. RINGENBERG, D. ERNST, T. M. AUSTIN, T. MUDGE, R. B. BROWN. *MiBench: A free, commercially representative embedded benchmark suite.*, in "IEEE 4th Annual Workshop on Workload Characterization", Austin, TX, December 2001.

[85] MARY H. HALL, SAMAN P. AMARASINGHE, BRIAN R. MURPHY, S.-W. LIAO, MONICA S. LAM. *Detecting coarse-grain parallelism using an interprocedural parallelizing compiler*, in "Supercomputing '95: Proceedings of the 1995 ACM/IEEE conference on Supercomputing (CDROM)", New York, NY, USA, ACM Press, 1995, 49, http://doi.acm.org/10.1145/224170.224337.

[86] L. HAMMOND, V. WONG, M. CHEN, B. D. CARLSTROM, J. D. DAVIS, B. HERTZBERG, M. K. PRABHU, H. WIJAYA, C. KOZYRAKIS, K. OLUKOTUN. *Transactional Memory Coherence and Consistency*, in "Proceedings of the 31st Annual International Symposium on Computer Architecture", IEEE Computer Society, June 2004, 102, http://tcc.stanford.edu/publications/tcc_isca2004.pdf.

[87] M. HANEDA, P. KNIJNENBURG, H. WIJSHOFF. *On the Impact of Data Input Sets on Statistical Compiler Tuning*, in "Workshop on Performance Optimization for High-Level Languages and Libraries (POHLL)", 2006.

[88] A. HASHMI, H. BERRY, O. TEMAM, M. LIPASTI. *Leveraging progress in neurobiology for computing systems*, in "1st Workshop on New Directions in Computer Architecture (NDCA-1)", December 2009.

[89] S. HU, M. VALLURI, L. K. JOHN. *Effective Adaptive Computing Environment Management via Dynamic Optimization*, in "IEEE / ACM International Symposium on Code Generation and Optimization (CGO 2005)", 2005.

[90] J. HUSELIUS. *Debugging Parallel Systems: A State of the Art Report*, Mälardalen University, Department of Computer Science and Engineering, September 2002, n^{o} 63, http://www.mrtc.mdh.se/index.php?choice=publications&id=0434.

[91] K. IBRAHIM, J. JAEGER, Z. LIU, L.-N. POUCHET, P. LESNICKI, L. DJOUDI, D. BARTHOU, F. BODIN, C. EISENBEIS, G. GROSDIDIER, O. PÈNE, P. ROUDEAU. *Simulation of the Lattice QCD and Technological Trends in Computation*, Aug 2008, n^{o} arXiv:0808.0391, submitted to the to the 14th International Workshop on Compilers for Parallel Computers.

[92] L. V. KALE, S. KRISHNAN. *CHARM++ : A Portable Concurrent Object-Oriented System Based on C++*, in "Proceedings of the Conference on Object Oriented Programming Systems, Languages and Applications (OOPSLA)", A. PAEPCKE (editor), ACM Press, September 1993, p. 91-108, http://citeseer.ist.psu.edu/viewdoc/download;jsessionid=B59CA14AAAA059061638FCDC61AD6E57?doi=10.1.1.55.1941&rep=rep1&type=ps.

[93] G. A. KOENIG, L. V. KALE. *Using Message-Driven Objects to Mask Latency in Grid Computing Applications*, in "19th IEEE International Parallel and Distributed Processing Symposium", April 2005.

[94] P. KULKARNI, S. HINES, J. HISER, D. WHALLEY, J. DAVIDSON, D. JONES. *Fast searches for effective optimization phase sequence*, in "Proc. ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI)", 2004.

[95] J. W. LAWSON, D. H. WOLPERT. *Adaptive Programming of Unconventional Nano-Architectures*, in "J. Comput. Theor. Nanosci.", 1986, vol. 3, p. 272-279.

[96] Y. LHUILLIER, O. TEMAM. *AP+SOMT: AgentProgramming SelfOrganized*, in "International Workshop on Complexity-Effective Design", Munich, Germany, ISCA, May 2004.

[97] Z. LI, O. CERTNER, J. DUATO, O. TEMAM. *Scalable Hardware Support for Conditional Parallelization*, in "IEEE/ACM, International Conference on Parallel Architecture and Compilation Techniques (PACT)", Vienna, Austria, September 2010.

[98] X. LI, M. GARZARAN, D. PADUA. *A dynamically tuned sorting library*, in "In ACM Conference on Code Generation and Optimization (CGO'04)", Palo Alto, California, March 2004.

[99] D. B. LOVEMAN. *High Performance Fortran*, in "IEEE Parallel Distrib. Technol.", 1993, vol. 1, n^{o} 1, p. 25–42, http://dx.doi.org/10.1109/88.219857.

[100] P. S. MAGNUSSON, M. CHRISTENSSON, J. ESKILSON, D. FORSGREN, G. HALLBERG, J. HOGBERG, F. LARSSON, A. MOESTEDT, B. WERNER. *Simics: A Full System Simulation Platform*, in "Computer", 2002, vol. 35, n^{o} 2, p. 50-58, http://doi.ieeecomputersociety.org/10.1109/2.982916.

[101] M. M. K. MARTIN, D. J. SORIN, B. M. BECKMANN, M. R. MARTY, M. XU, A. R. ALAMELDEEN, K. E. MOORE, M. D. HILL, D. A. WOOD. *Multifacet's general execution-driven multiprocessor simulator (GEMS) toolset*, in "SIGARCH Comput. Archit. News", 2005, vol. 33, n^{o} 4, p. 92–99, http://doi.acm.org/10.1145/1105734.1105747.

[102] D. E. MAYDAN, J. L. HENNESSY, M. S. LAM. *Efficient and Exact Data Dependency Analysis*, in "Proceedings of the SIGPLAN '91 Conference on Programming Language Design and Implementation", June 1991, p. 1-14.

[103] B. MEISTER, A. LEUNG, N. VASILACHE, D. WOHLFORD, C. BASTOUL, R. LETHIN. *Productivity via Automatic Code Generation for PGAS Platforms with the R-Stream Compiler*, in "APGAS'09 Workshop on Asynchrony in the PGAS Programming Model", Yorktown Heights, New York, June 2009.

[104] A. MONSIFROT, F. BODIN, R. QUINIOU. *A machine learning approach to automatic production of compiler heuristics*, in "Proc. AIMSA", LNCS 2443, 2002, p. 41-50.

[105] M. O'BOYLE, P. KNIJNENBURG, G. FURSIN. *Feedback Assisted Iterative Compiplation*, in "Parallel Architectures and Compilation Techniques (PACT'01)", IEEE Computer Society Pres, October 2001.

[106] P. PALATIN, Y. LHUILLIER, O. TEMAM. *Capsule : Hardware-Assisted Parallel Execution of Component-Based Programs*, in "The 39th Annual IEEE/ACM International Symposium on Microarchitecture, 2006", Orlando, Florida, december 2006.

[107] D. PARELLO, O. TEMAM, A. COHEN, J.-M. VERDUN. *Towards a Systematic, Pragmatic and Architecture-Aware Program Optimization Process for Complex Processors*, in "ACM Supercomputing'04", Pittsburgh, Pennsylvania, November 2004, 15, http://www-rocq.inria.fr/~acohen/publications/PTCV04.ps.gz.

[108] D. PARELLO, O. TEMAM, J.-M. VERDUN. *On increasing architecture awareness in program optimizations to bridge the gap between peak and sustained processor performance: matrix-multiply revisited.*, in "SC", 2002, p. 1-11, http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.132.3339.

[109] T. POGGIO, C. R. SHELTON. *Machine Learning, Machine Vision, and the Brain*, in "The AI Magazine", 1999, vol. 20, n^{o} 3, p. 37–55, http://www.aaai.org/ojs/index.php/aimagazine/article/viewArticle/1465.

[110] S. POP, A. COHEN, C. BASTOUL, S. GIRBAL, P. JOUVELOT, G.-A. SILBER, N. VASILACHE. *GRAPHITE: Loop optimizations based on the polyhedral model for GCC*, in "Proc. of the 4th GCC Developper's Summit", Ottawa, Canada, June 2006.

[111] L.-N. POUCHET, C. BASTOUL, J. CAVAZOS, A. COHEN. *A Note on the Performance Distribution of Affine Schedules*, in "2nd Workshop on Statistical and Machine learning approaches to ARchitectures and compilaTion (SMART'08)", Göteborg, Sweden, January 2008.

[112] L.-N. POUCHET, C. BASTOUL, A. COHEN, J. CAVAZOS. *Iterative optimization in the polyhedral model: Part II, multidimensional time*, in "ACM SIGPLAN Conference on Programming Language Design and Implementation (PLDI'08)", Tucson, Arizona, June 2008.

[113] L.-N. POUCHET, C. BASTOUL, A. COHEN, N. VASILACHE. *Iterative optimization in the polyhedral model: Part I, one-dimensional time*, in "ACM International Conference on Code Generation and Optimization (CGO'07)", San Jose, California, March 2007, p. 144–156.

[114] W. PUGH. *The Omega test: A fast and practical integer programming algorithm for dependence analysis*, in "Comm. of the ACM", 1992, vol. 8, p. 102-114.

[115] C. G. QUIÑONES, C. MADRILES, J. SÁNCHEZ, P. MARCUELLO, A. GONZÁLEZ, D. M. TULLSEN. *Mitosis Compiler: An Infrastructure for Speculative Threading Based on Pre-Computation Slices*, in "PLDI '05: Proceedings of the ACM SIGPLAN 2004 conference on Programming language design and implementation", ACM Press, 2005.

[116] SIA. *Semiconductor Industry Association 2005 roadmap, section on Emerging Research Devices*, 2005, http://www.sia-online.org/.

[117] B. SIRI, H. BERRY, B. CESSAC, B. DELORD, M. QUOY. *Topological and dynamical structures induced by Hebbian learning in random neural networks*, in "International Conference on Complex Systems, ICCS 2006", Boston, MA, USA, June 2006.

[118] B. SIRI, H. BERRY, B. CESSAC, B. DELORD, M. QUOY, O. TEMAM. *Learning-induced topological effects on dynamics in neural networks*, in "Proceedings of NeuroComp'06", Pont-à-Mousson, France, 23-24 October 2006, p. 206–209.

[119] M. SMITH. *Overcoming the challenges to feedback-directed optimization*, in "Proc. ACM SIGPLAN Workshop on Dynamic and Adaptive Compilation and Optimization (Dynamo'00)", 2000.

[120] C. SZYPERSKI. *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[121] C. TEUSCHER. *Small-World Power-Law Interconnects for Nanoscale Computing Architectures*, in "Proceedings of the 6th IEEE Conference on Nanotechnology, IEEE Nano 2006", July 2006.

[122] W. THIES, M. KARCZMAREK, M. GORDON, D. MAZE, J. WONG, H. HO, M. BROWN, S. AMARASINGHE. *StreamIt: A Compiler for Streaming Applications*, December 2001, MIT-LCS Technical Memo TM-622, Cambridge, MA, http://publications.csail.mit.edu/lcs/pubs/pdf/MIT-LCS-TM-622.pdf.

[123] S. TRIANTAFYLLIS, M. VACHHARAJANI, N. VACHHARAJANI, D. I. AUGUST. *Compiler optimization-space exploration*, in "Proc. International Symposium on Code Generation and Optimization", 2003, p. 204–215.

[124] M. VACHHARAJANI, N. VACHHARAJANI, D. A. PENRY, J. A. BLOME, D. I. AUGUST. *Microarchitectural Exploration with Liberty*, in "the 34th Annual International Symposium on Microarchitecture", Austin, Texas, USA., December 2001.

[125] N. VASILACHE, C. BASTOUL, A. COHEN. *Polyhedral Code Generation in the Real World*, in "Proceedings of the International Conference on Compiler Construction (ETAPS CC'06)", Vienna, Austria, LNCS, Springer-Verlag, March 2006, p. 185–201, http://www-rocq.inria.fr/~acohen/publications/VBC06.ps.gz.

[126] N. VASILACHE, C. BASTOUL, S. GIRBAL, A. COHEN. *Violated dependence analysis*, in "Proceedings of the ACM International Conference on Supercomputing (ICS'06)", Cairns, Australia, ACM, June 2006.

[127] M. VOSS, R. EIGENMANN. *ADAPT: Automated de-coupled adaptive program transformation*, in "Proc. ICPP", 2000.

[128] R. VUDUC, J. BILMES, J. DEMMEL. *Statistical Modeling of Feedback Data in an Automatic Tuning System*, in "Proc. 3rd ACM Workshop on Feedback-Directed and Dynamic Optimization", 2000, p. 41-50.

[129] D. WALLIN, H. ZEFFER, M. KARLSSON, E. HAGERSTEN. *Vasa: A Simulator Infrastructure with Adjustable Fidelity*, in "Proceedings of the 17th IASTED International Conference on Parallel and Distributed Computing and Systems (PDCS 2005)", Phoenix, Arizona, USA, November 2005.

[130] M. WOLF, M. LAM. *A loop transformation theory and an algorithm to maximize parallelism*, in "IEEE Transactions on Parallel and Distributed Systems", 1991, vol. 2, n^{o} 4, p. 430-439.

[131] S. YEHIA, S. GIRBAL, H. BERRY, O. TEMAM. *Reconciling Specialization and Flexibility Through Compound Circuits*, in "15th International Symposium on High-Performance Computer Architecture, HPCA", Raleigh, North Carolina, February 2009.

[132] S. YEHIA, O. TEMAM. *From Sequences of Dependent Instructions to Functions: An Approach for Improving Performance without ILP or Speculation*, in "International Symposium on Computer Architecture", May 2004.