



INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team Alchemy

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

Saclay - Île-de-France

THEME COM

Activity
R *eport*

2008

Table of contents

1. Team	1
2. Overall Objectives	2
3. Scientific Foundations	2
3.1.1. A practical approach to program optimizations for complex architectures	2
3.1.1.1. Iterative optimization	2
3.1.1.2. Polyhedral program representation: facilitating the analysis and transformation of programs	4
3.1.1.3. Project-team positioning	4
3.1.2. Joint architecture/programming approaches	5
3.1.2.1. A targeted domain: Passing program semantics using a synchronous language for high-performance video processing	5
3.1.2.2. A more general approach: Passing program semantic using software components	6
3.1.2.3. Personnel	7
3.1.2.4. Project-team positioning	7
3.1.3. Alternative computing models/Spatial computing	7
3.1.4. Transversal research activities: simulation and compilation	9
3.1.4.1. Simulation platform	9
3.1.4.2. Compilation platform	10
3.1.4.3. Project-team positioning	11
4. Software	12
5. New Results	13
5.1. Program optimizations	13
5.1.1. Practical Approach	13
5.1.2. Transitive Closure of Union of Affine Relations	14
5.1.3. Optimizing code through iterative specialization	14
5.1.4. Simulation of the Lattice QCD and Technological Trends in Computation	14
5.1.5. Loop Optimization using Adaptive Compilation and Kernel Decomposition	15
5.1.6. Dataflow Analysis for Irregular Programs and its applications	15
5.2. Joint architecture/programming approaches	15
5.3. Alternative computing models/Spatial computing	15
5.3.1. Blob computing	15
5.3.2. Bio-Inspired Computing	16
5.3.2.1. A mathematical analysis of the effects of Hebbian learning rules on the dynamics and structure of discrete-time random recurrent neural networks	16
5.3.2.2. Supervised and Evolutionary Learning of Echo State Networks: Balancing the Double Pole	16
5.3.2.3. A formalization of the computational impact of intrinsic plasticity	16
5.3.2.4. From molecules to organisms: towards multiscale integrated model of biological systems	17
5.3.2.5. Nonequilibrium phase transition in scattered cell communities coupled by auto/paracrine-like signalling	17
5.3.2.6. Biological neural networks as bio-inspiration sources for future architectures	17
5.3.3. Spatial complexity of reversible computing	18
6. Contracts and Grants with Industry	18
6.1. Collaborations involving industry	18
6.2. National and international collaborative grants	18
7. Other Grants and Activities	20
7.1. Informal collaborations	20
7.2. Seminar and invited scientists	22

8. Dissemination	22
8.1. Leadership within scientific community	22
8.2. Teaching at university	25
8.3. Workshops, seminars, invitations	25
9. Bibliography	27

1. Team

Research Scientist

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

Faculty Member

Denis Barthou [Assistant professor, University of Versailles-Saint-Quentin, delegation Inria]
Cédric Bastoul [Assistant Professor]
Frédéric Gruau [Assistant Professor]

External Collaborator

Pierre Amiranoff [PRAG, IUT d'Orsay]
Nathalie Drach [Professor, Paris-6 University]
Sid-Ahmed-Ali Touati [Assistant professor, University of Versailles-Saint-Quentin]

Technical Staff

Martin Schindewolf [T. U. Karlsruhe, 1 month]
Sylvain Girbal [Expert engineer, FP6 IST grant]

PhD Student

Uday Bondhugula [Ohio State U., 2 months]
Andrea Marongiu [U. Bologna, 3 months]
Walid Ben Abderrhamane [MENRT scholarship, University of Paris-Sud]
Mouad Bahi [Inria scholarship, University of Paris-Sud]
Cupertino Miranda [Portugese grant, University of Paris-Sud]
Konrad Trifunovic [Inria scholarship, University of Paris-Sud]
Boubacar Diouf [MENRT scholarship, University of Paris-Sud]
Mounira Bachir [Inria scholarship, University of Versailles-Saint-Quentin]
Olivier Certner [STMicroElectronics fellowship (Cifre), University of Paris-Sud]
Mohammed Fellahi [Inria scholarship, University of Paris-Sud]
Fei Jiang [Inria scholarship, with the TAO Inria team, University of Paris-Sud]
Taj Muhammad Khan [Inria scholarship, University of Paris-Sud]
Piotr Lesnicki [MENRT scholarship, University of Paris-Sud]
Zheng Li [Inria scholarship, University of Paris-Sud]
Luidnel Maignan [MENRT scholarship, University of Paris-Sud]
Louis-Noël Pouchet [MENRT scholarship, University of Paris-Sud]
Adrien Eliche [MENRT scholarship, University of Versailles-Saint-Quentin]
Benoît Siri [Inria scholarship, University of Paris-Sud]
Marouane Belaoucha [MESR scholarship, co-supervised with S. Touati, University of Versailles-Saint-Quentin]
Andres Charif-Rubial [ANR ProHMPT, co-supervised with W. Jalby, Prism, University of Versailles-Saint-Quentin]
Julien Jaeger [ANR Para, University of Versailles-Saint-Quentin]
Pablo Oliveira [CEA scholarship, co-supervised with S. Louise, University of Versailles-Saint-Quentin]

Administrative Assistant

Valérie Berthou [TR Inria]

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 [104] at University of Edinburgh, UK or Keith Cooper [62] 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 [39], [53] [71] [34].

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 [110], [29]. 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 [61], [75], [70], [94], [40], only recently, some studies have focused on the impact of data sets on iterative optimizations [89], [49].

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 [86] 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 [107], [106]. 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] [79]. 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 [64] 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 [129]. 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], [111], [101], [128], 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 [59], [123], [124], [79], and allows to significantly reduce the size of the search space and better understand its structure [110], [29], [28]. 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 [109], 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 [104], and more recently by Keith Cooper at Rice University [62]. 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) [126], [117], [90], [125]. 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 [97], [102], or even that it can be integrated in production compilers to assist existing optimizations [121]. 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 [54], [58].

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 Sychrone* 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 [56], [57]. 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 [87]. Writing, and especially debugging, large parallel programs is a notoriously difficult task [91], and one may wonder whether the vast majority of programmers will be able to cope with it. Currently, transactional memory is a popular approach [88] 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 [98], 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* [118]. 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], [105]. 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 [41] 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 [112]. As mentioned before, transactional memory has become a popular approach [88] for reducing the burden of parallelizing applications. Other approaches include parallel languages, such as HPF [98] or parallel directives such as OpenMP [63].

Synchronous 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 [120], and to a lesser extent, the Sequoia project from Stanford [66]; 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 [51] 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. [55] 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 [80] 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 [85] 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” [113] 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. [108] 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 [116], [115], [47] [48]. 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 [45], [46]. And we are starting to apply them to sampling techniques for performance evaluation [81], [82]. 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 [80], the Blob computing work in our research group refers more to unconventional spatial programming paradigms such as MGS [77] and Gamma [42].

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], [119].

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 [38].

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 [84], 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 [37], or the M5 simulator [50]. 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 [99] 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;
- 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 [52]) 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 [65], TSS at Philips, MaxSim at ARM,...), they are not standard nor disseminated. Only SystemC [37] 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 [122]. 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 [50], Flexus [33], GEMS [100] or Vasa [127].

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

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 [44] and of the CLoG code generator [43]. 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 [60]. 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 — performance-wise and infrastructure-wise — 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 Participant: Cédric Bastoul.

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.

Clan Participant: Cédric Bastoul.

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. 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 CLoG, LeTSeE, Candl etc.). Using Clan, the user has to deal with source codes based on C grammar only (as C, C++, C# or Java).

FADALib (<http://www.prism.uvsq.fr/users/bem/fadalib/home.html>). Dataflow dependence for irregular programs (not static control programs). The library is developed 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 developers to focus on code fragments that require performance tuning, analyzes compiler optimizations and proposes tuning hints. MAQAO works on Itanium, Pentium architectures.

PROCESSOR SIMULATION:

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.

BeeRS & IDDCA BeeRS [83] is a sampling technique that focuses on practicality by jointly considering warm-up and sampling. Most sampling techniques treat the problem separately which complicates their practical usage. BeeRS also includes the IDDCA clustering technique which has been shown to outperform traditional k-means techniques by an order of magnitude [81].

MicroLib MicroLib [84] is our former version of a modular simulation platform. It includes a library of modular simulator components freely distributed on a web site (<http://www.microlib.org>). As of now, it contains generic modules for each of the main components of a superscalar processor, a full superscalar processor model, an embedded processor model (PowerPC 750).

FastSysC FastSysC [103] is an enhanced SystemC engine. SystemC is itself a modular simulation environment which is becoming a de facto standard supported by more than 50 companies in the embedded domain. However, the SystemC engine development is geared toward adding functionalities rather than improving performance. Because performance is critical in processor simulation, due to excessively long traces, we have developed from scratch a new SystemC engine geared toward performance.

DiST As part of our efforts on speeding up simulation execution, we have developed a tool for parallelizing simulators, called DiST [78], requiring little simulator modifications and incurring only a small loss of accuracy. The main asset of the tool is that it can take advantage of multiple computing resources.

5. New Results

5.1. Program optimizations

5.1.1. Practical Approach

Participants: Albert Cohen, Louis-Noël Pouchet.

Here are the most recent key scientific achievements.

- Empirically demonstrating that significant performance gains can be achieved with program optimizations, provided architecture phenomena are better factored in during the optimization process. Observing though that long compositions of program transformations are required.
- Showing that it is possible to capture the complex interplays between architecture and program behavior using machine-learning techniques, using that knowledge to drive program optimizations.

Publications of 2008: [71], [74] [34][22].

- Developing the first version of the machine-learning based research compiler (MILEPOST GCCFootciteFMTP2008) that combines Interactive Compilation Interface [34] and static program feature extractor to predict good program optimizations to reduce execution time and code size for a given program on a given architecture automatically using predictive modeling and statistical techniques. This compiler opens many research opportunities and is used in the EU HiPEAC network of excellence [35] as a default compilation platform. IBM made a press-release about this work in June, 2008 [36].
- Developing a polyhedral program representation that facilitates the composition of complex transformation sequences.
- Addressing the code generation performance issues associated with polyhedral program representation.
- Further leveraging polyhedral program representation to propose novel methods for scanning the space of program transformations.

Publications of 2008: [29], [28].

5.1.2. *Transitive Closure of Union of Affine Relations*

Participants: Denis Barthou, Anna Beletska, Albert Cohen, Konrad Trifunovic.

We studied a method to compute the transitive closure of a union of affine relations on integer tuples. Within Presburger arithmetics, complete algorithms to compute the transitive closure exist for convex polyhedra only. In presence of non-convex relations, there exists little but special cases and incomplete heuristics. We introduce novel sufficient and necessary conditions defining a class of relations for which an exact computation is possible. These conditions can be relaxed to define larger classes where conservative approximations and/or more complex closed forms can be obtained. Our method is immediately applicable to a wide area of symbolic computation problems. It is illustrated on representative examples and compared with state of the art approaches.

5.1.3. *Optimizing code through iterative specialization*

Participants: Minhaj Khan, Henri-Pierre Charles, Denis Barthou.

Code specialization is a way to obtain significant improvement in the performance of an application. It works by exposing values of different parameters in source code. The availability of these specialized values enables the compilers to generate better optimized code. Although most of the efficient source code implementations contain specialized code to benefit from these optimizations, the real impact of specialization may however vary depending upon the value of the specializing parameter.

We have studied in [26] an iterative approach for code specialization. From some specialized code, we search for a better version of code by re-specializing the code, followed by a low-level code analysis. The specialized versions fulfilling the required criteria are then transformed to generate another equivalent version of the original specialized code. The approach, tested on Itanium2 architecture using gcc/icc compilers show significant improvement in the performance of different benchmarks.

5.1.4. *Simulation of the Lattice QCD and Technological Trends in Computation*

Participants: 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 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 [32], 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 (2008-2010).

5.1.5. Loop Optimization using Adaptive Compilation and Kernel Decomposition

Participants: J. Jaeger, P. Oliveira, S. Louise, D. Barthou.

We study a new hierarchical compilation approach for the generation of high performance applications, relying on the use of state of the art compilers. This approach is not application dependent and do not require any assembly hand-coding. It relies on the decomposition of the loop nests of the hottest functions in the application into simpler kernels, typically 1D to 2D loops, much simpler to optimize. We successfully applied this approach for dense linear algebra in 2005, reaching performance of constructor libraries. The advantage of the generated kernels is that their performance no longer depend on data input, but only on its location in memory hierarchy. Using a performance model for the memory hierarchy, it is possible to find out the best composition of kernels to use.

For larger applications, the code is no longer regular and data accesses are in particular irregular (use of indirections). Working with applications of project ANR PARA (MPEG4, QCD, oil simulation and BLAST), we study how to adapt the previous approach to these cases. When control is irregular (involving different execution path), we study the the WCET, in particular in the context of embedded applications for MPSOC architectures. This is the subject of an on-going collaboration with CEA/Lastre.

5.1.6. Dataflow Analysis for Irregular Programs and its applications

Participants: M. Belaoucha, S. Touati, D. Barthou.

Instance-wise dataflow analysis provides the exact execution of a statement defining a value that is read at some other point during a program execution. This analysis generates more precise information than traditional dependence analyses and can therefore validate more optimizing transformations. An implementation of this analysis, as a standalone library, has be performed by M. Belaoucha (and funded by contract Teraops and PARMA) and its integration in gcc/Graphite is in progress.

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.
- Showing that a combination of hardware support and component-like programming can relieve the user of some of the burdens of parallel programming (especially task granularity and load balancing). The approach may also suggest a research direction for satisfying real-time constraints in the context of complex micro-architectures, without resorting to WCET analysis [7].

5.3. Alternative computing models/Spatial computing

5.3.1. Blob computing

Participants: Frédéric Gruau, Christine Eisenbeis, Luidnel Maignan.

This year, the blob project has developed in two directions:

1. We have developed syntactic constructs in a Caml-like functional style that allows to describe concisely blob programs doing non trivial algorithm;
2. We have started implementing the backbone algorithm of a simulation platform for blob computing: it is an iterated construction of Voronoi diagrams on cellular automata that allow to move particles in order to homogeneize their density. We are studying the convergence properties and convergence and time in the 1D, and 2D cases. This algorithm can be generalized also to the homogeneization of blob membranes and will thus be at the heart of the blob simulator.

A paper about the foundations of the blob computing system has appeared in [16]

5.3.2. Bio-Inspired Computing

5.3.2.1. A mathematical analysis of the effects of Hebbian learning rules on the dynamics and structure of discrete-time random recurrent neural networks

Participants: Hugues Berry, Bruno Cessac, Bruno Delord, Mathias Quoy, Benoît Siri.

We present in [18] a mathematical analysis of the effects of Hebbian learning in random recurrent neural networks, with a generic Hebbian learning rule, including passive forgetting and different timescales, for neuronal activity and learning dynamics. Previous numerical work has reported that Hebbian learning drives the system from chaos to a steady state through a sequence of bifurcations. Here, we interpret these results mathematically and show that these effects, involving a complex coupling between neuronal dynamics and synaptic graph structure, can be analyzed using Jacobian matrices, which introduce both a structural and a dynamical point of view on neural network evolution. Furthermore, we show that sensitivity to a learned pattern is maximal when the largest Lyapunov exponent is close to 0. We discuss how neural networks may take advantage of this regime of high functional interest.

5.3.2.2. Supervised and Evolutionary Learning of Echo State Networks: Balancing the Double Pole

Participants: Hugues Berry, Fei Jiang, Marc Schoenauer.

A possible alternative to topology fine-tuning for Neural Network (NN) optimization is to use Echo State Networks (ESNs), recurrent NNs built upon a large reservoir of sparsely randomly connected neurons. The promises of ESNs have been fulfilled for supervised learning tasks, but unsupervised ones, e.g. control problems, require more flexible optimization methods - such as Evolutionary Algorithms. The papers [24], [25] propose to apply CMA-ES, the state-of-the-art method in evolutionary continuous parameter optimization, to the evolutionary learning of ESN parameters. First, a standard supervised learning problem is used to validate the approach and compare it to the standard one. But the flexibility of Evolutionary optimization allows us to optimize not only the outgoing weights but also, or alternatively, other ESN parameters, sometimes leading to improved results. The classical double pole balancing control problem is then used to demonstrate the feasibility of evolutionary (i.e. reinforcement) learning of ESNs. We show that the evolutionary ESN obtain results that are comparable with those of the best topology-learning methods.

5.3.2.3. A formalization of the computational impact of intrinsic plasticity

Participants: Jérémie Naudé, Stephane Genet, Hugues Berry, Jeanne Paz, Bruno Delord.

La plasticité synaptique dépendante de l'activité est classiquement considérée comme le substrat cellulaire de l'apprentissage et de la mémorisation. Cependant, il est expérimentalement avéré que les propriétés membranaires des neurones sont également modifiées par l'activité. L'impact fonctionnel de cette plasticité intrinsèque reste largement inconnu. Récemment, Paz et al. (soumis) ont montré in vivo que le seuil et le gain de la relation intensité-fréquence (I-f) des neurones pyramidaux du cortex de rat sont modifiés par l'activité, probablement via la régulation plastique de conductances ioniques non synaptiques. Cependant, l'effet de la plasticité intrinsèque sur la fonction de transfert des neurones reste encore inexpliqué en terme de mécanismes. Nous montrons dans [27] au travers de simulations de type Hodgkin-Huxley que l'impact computationnel de la plasticité de conductances infra- ou supraliminaire (s'activant en dessous ou au-dessus du seuil du potentiel d'action) diffère radicalement. En effet, ces formes de plasticité affectent respectivement le seuil et le gain de

la relation I-f des neurones et s'appliquent à la vaste majorité des conductances sodium, calcium et potassium. Nous décrivons analytiquement comment la relation I-f dépend explicitement des paramètres biophysiques de la conductance plastique, et démontrons que les modifications plastiques dues à différentes conductances se somment linéairement. Ensemble, ces résultats montrent que la plasticité intrinsèque permet au neurone de sélectionner indépendamment la distribution de ses entrées et la réponse à celles-ci. Nos résultats analytiques permettent de tester directement l'impact fonctionnel probablement crucial de ces effets à l'échelle des réseaux de neurones.

5.3.2.4. *From molecules to organisms: towards multiscale integrated model of biological systems*

Participants: Hugues Berry, Christophe Lavelle, Guillaume Beslon.

A consensus has recently emerged that further progress in understanding human physiopathology will demand integrative views of biological systems. In this context, complex systems and related interdisciplinary approaches of biology are expected to help. The aim of the collective paper [17] is basically to provide a starting point for further discussions and interactions within the community of complex systems biologists. After briefly introducing some general concepts, we present four major challenges that should be tackled in the next years. These represent future directions that we isolated as priority concerns for modern biology. Suggestions of how to reach these destinations are provided, with the hope that they will soon lead to concrete advances towards fully consistent multiscale models of biological systems and a better understanding of physiopathology.

5.3.2.5. *Nonequilibrium phase transition in scattered cell communities coupled by auto/paracrine-like signalling*

Participant: Hugues Berry.

Auto/paracrine cell-to-cell communications via diffusive messengers can be coupled to a positive feedback loop in which cell stimulation by a messenger results in the production of new messengers. This yields a potential mechanism for relay transmission of the emitted message. The paper [31] investigates the influence of noise on this mutual coupling of the cells with their environment, using numerical simulations of a stochastic minimal model. The results demonstrate that the deterministic (mean-field) approximation of this stochastic process fails short of predicting its behaviour because of the presence of strong noise-induced fluctuations. Instead, the behaviour of the model can be explained by the occurrence of a nonequilibrium phase transition, which is found to be in the universality class of directed percolation. This provides a theoretical framework to understand signal transmission in these stochastic systems.

5.3.2.6. *Biological neural networks as bio-inspiration sources for future architectures*

Participants: Hugues Berry, Olivier Temam.

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 such a large number of computing resources. Searching for 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" [113] 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. [108] 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 [48]. While image processing is also our first application, we plan to later look at a more diverse set of example applications.

5.3.3. Spatial complexity of reversible computing

Participants: Mouad Bahi, Benjamin Dauvergne, 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 than $n/2$ 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 [30]. In this work, all operations in the DAG are assumed to be reversible.

6. Contracts and Grants with Industry

6.1. Collaborations involving industry

STMicroelectronics Besides the HiPEAC network of excellence, and IP SARC, we have a regular and informal collaboration on iterative compilation and novel processor architecture with the AST (Advanced Systems and Technologies) research group of STMicroelectronics based in Lugano, Switzerland and Grenoble, France.

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.

ARM R&D, Cambridge In the context of the SARC FP6 FET Proactive IP project, Pierre Palatin spent a 3 months summer internship at ARM R&D, Cambridge. The goal was the application of Capsule on specific ARM architectures.

6.2. National and international collaborative grants

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 Méditerranée (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 involvement lie in the compiler infrastructure, static analysis in the polyhedral model, and feature extraction for global and continuous 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.
- ACOTES: EU, IST program FP6 STREP on language and compiler support for high-performance streaming applications. We are one of the largest contractors in the project, with major involvement in interprocedural optimization and loop transformations for concurrent distributed streaming applications; it is both a programming model and compiler project. With Philips Research (Eindhoven), IBM Research (Haifa), STMicroelectronics (AST Lugano), Nokia (Helsinki), and UPC (Barcelona). 05/2006–05/2009.
- MilePost: EU, IST program FP6 STREP on machine-learning compilation. This project matches one of the core directions of the project: iterative optimization research, with an emphasis on making iterative compilation methods practical in real development environments. With IBM Research (Haifa), ARC (London), CAPS Entreprise (Rennes), IRISA (Rennes), and University of Edinburgh. 05/2006–05/2009.
- 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.
- APE: French Ministry of Research ANR RNTL project on parallel real-time applications for embedded systems. We are developing a component-based environment called CAPSULE for distributed-memory processors. It will be applied to a novel processor of STMicroelectronics and tested on applications from Thales. With STMicroelectronics, Thales, University of Paris 6, CEA. 01/2006–01/2009.
- PSYCHES: EU, IST program Marie Curie ToK IAP (Transfer of Knowledge, Industry-Academia Partnership); long-term exchange of personnel and 2 years of post-doc; with Philips Research (Eindhoven) and UPC (Barcelona). 03/2006–03/2009.
- 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 Compétitivité” 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.

MODSIM MODSIM is an INRIA grant for a joint international team between INRIA and Princeton University. The goal is the development of the UNISIM simulation platform. With Princeton University. 01/2006–12/2009.

ACI ASTICO Grant French Minister of Research grant to explore biological neuron networks as possible sources of inspiration for future computing systems, with a focus on the complex structure of these networks. Our aim is at the same time to investigate bio-inspired computing systems, and original approaches for the modeling and understanding of biological neural networks. With University of Cergy-Pontoise, University of Nice-Sophia-Antipolis and University of Paris 6. 01/2005–01/2008.

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.

item[FET OMP] OpenMediaPlatform (OMP) aims at overcoming the cost and time-to-market risks that affect the development of media-rich evolving services for the growing range of networked consumer devices. It will provide an open architecture, combining two main streams of modern software engineering: (1) open application programmers interfaces (API) for media components, to be enhanced over standards like Khronos OpenMAX, and (2) new resource-aware system design tools and standards-complying static/dynamic compilers that ease the design, implementation and efficient execution of media services on a range of consumer platforms. 01/2008–12/2009.

ACI Nanosys French Minister of Research grant to study the impact of alternative technologies, particularly nanotubes, on future computing circuits and architectures. With a large array of French laboratories in VLSI and architecture design.

- Hugues Berry is a member of GdR Dycoec: “Dynamique et contrôle des ensembles complexes” (<http://www.coria.fr/dycoec/>)

7. Other Grants and Activities

7.1. Informal collaborations

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.

Hugues Berry collaborates with these people.

- Bruno Cessac (Institut Non Linéaire de Nice, UMR 6618 CNRS / Université Nice-Sophia Antipolis; Team-Project Odyssee, INRIA Sophia)
- Bruno Delord, Stéphane Genet, E. Guigon (ANIM, UMR 742 Inserm / Université Pierre et Marie Curie, Paris)
- Mathias Quoy (ETIS, UMR 8051 CNRS / Université de Cergy-Pontoise / ENSEA)
- Olivier Michel (LACL, U. Paris 12 Creteil)
- Marc Schoenauer (TAO, INRIA Saclay)
- Nazim Fates (MAIA, INRIA Loraine, Nancy), Bernard Girau (Cortex, INRIA Loraine, Nancy)
- Francois Taddei, Ariel Lindner (INSERM U571, Faculté de Médecine Necker-Enfants Malades, Paris)
- Eshel Ben-Jacob (School of Physics and Astronomy, Tel Aviv University, Israel)

Olivier Temam collaborates with Veerle Desmet at Ghent University, Belgium, on design-space exploration. He collaborates with Thales TRT on several topics (main contact: Sami Yehia).

University of Princeton We have an active collaboration with the Liberty group (David August) at University of Princeton in the past year. The goal is to unify our approach in modular simulation within the UNISIM framework and thus increase the likelihood that a joint environment be adopted by the wider community. This interaction is further synchronized with the Common Simulation Platform activity of the HiPEAC network. Starting January 2005, we obtained an “Joint Team” grant called MODSIM, together with the Liberty group at University of Princeton.

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 2 years, we had a very active cooperation with University of Edinburgh on iterative optimization; Grigori Fursin, postdoc in our group, got his PhD from University Edinburgh. This collaboration has resulted in a series of joint articles [72], [53], [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. Prof.Rauchwerger 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 pitfalls in scalable parallel processing, and also facilitates the organization of multiple academic events (e.g., the upcoming PACT’07)

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 [76].

Project-Team TAO, INRIA Futurs We started a collaboration with Marc Schoenauer on evolutionary algorithms for optimization of complex systems. More precisely, we study evolutionary methods to optimize the complex structure of large size neural networks. The aim is to find whether there exists optimal organizations for the interconnect network of such large systems. This collaboration grounds F. Jiang’s Ph.D. work, which is co-supervised and co-founded by the two groups.

CEA List For the past 6 years, we had a regular collaboration with the *Laboratoire Sûreté 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).

Cédric Bastoul collaborates with Sébastien Salva from Clermont 1 University and Clément Delamare from Direction Générale des Impôts on web service client parallelization [114].

Hugues Berry collaborates with Bruno Cessac (Institut Non Linéaire de Nice, UMR 6618 CNRS / Université Nice-Sophia Antipolis), Bruno Delord (ANIM, UMR 742 Inserm / Université Pierre et Marie Curie, Paris), Stéphane Genet (ANIM, UMR 742 Inserm / Université Pierre et Marie Curie, Paris), Mathias Quoy (ETIS, UMR 8051 CNRS / Université de Cergy-Pontoise / ENSEA), Olivier Michel (Ibisc, Université d'Evry), Marc Schoenauer (TAO, INRIA Futurs, Orsay), Nazim Fates (MAIA, INRIA Lorraine, Nancy).

7.2. Seminar and invited scientists

Marc Duranton (Philips NXP, Eindhoven, Netherlands) visits ALCHEMY regularly.

8. Dissemination

8.1. Leadership within scientific community

Cédric Bastoul

- Member of the LRI department committee at the University of Paris-Sud of Paris-Sud since 2006.
- Member of the Orsay Technology Institute (IUT D'Orsay) Computer Science department committee since 2006.
- Director of the *Licence Professionnelle Sécurité des Systèmes et Réseaux Informatiques* (third year diploma on System and Network Security) at Orsay Institute of Technology since 2007.

Hugues Berry

- Hugues Berry is a member of the “Scientific Commission” (commission scientifique) of the INRIA Saclay-Ile-de-France research centre.

Albert Cohen

- HiPEAC'06 Summer School course on GCC (55-65 attendees). The support material for the courses and tutorials is freely available (public domain or GPL license) and has been contributed to the main GCC site (<http://gcc.gnu.org>, Wiki section; see also <http://www.hipeac.net/gcc-tutorial>).
- Founding member of IFIP WG 2.11.
- President of the 1st recruiting committee (admissibilité) for INRIA Saclay research scientists, 2007 and 2008.

Christine Eisenbeis

- member of IFIP WG 10.3.

- member of the “comité de programmes” of Digiteo.
- elected member of the “conseil d’administration” of Inria [2006-].
- elected member of the “comité technique paritaire” of Inria [2006-].
- elected member of the “conseil scientifique” of University of Paris-Sud [2008-].
- chair [2008-] of the “commission des utilisateurs des moyens informatiques - recherche” of the Saclay Inria Research Center.

Olivier Temam

- ANR Future Processor Architectures grants evaluation committee, 2008.
- HiPEAC2 Steering Committee, Research workpackage leader, leader of the Research Cluster on simulation.

PROGRAM COMMITTEES:

Denis Barthou

- Program committee member of IEEE HPCC 2009, The 11th IEEE International Conference on High Performance Computing and Communications (HPCC-09), June 2009 Korea University, Seoul, Korea

Cédric Bastoul

- Program committee member of DATES 2009 The 12th International Conference on Design, Automation & Test in Europe.
- Program committee member of SSS 2009 The 11th International Symposium on Stabilization, Safety, and Security of Distributed Systems.

Hugues Berry

- Co-organizer of the Workshop “From Amorphous to Spatial Computing”, July 7-8, 2008, Paris.
- Member of the organization committee of the CNRS-INRA thematic school “**EIEFB**: Ecole interdisciplinaire d’échanges et de formation en biologie”, Berder Island (Morbihan, France), March 20 - April 5, 2008 (<http://www.lptmc.jussieu.fr/users/lesne/berder2008.html>). Main Organizer: Annick Lesne (LPTMC, Univ. P.M. Curie, Paris)
- Scientific vulgarization:
 - * “Alchimistes de l’informatique”: article in *La Recherche* on bio-inspired computing, may 2008.;
 - * “Quand les bactéries se mêlent du calcul des ordinateurs...”: a short vulgarization text plus a longer podcast (in French) on the *interstices* web journal (http://interstices.info/jcms/c_40644/quand-les-bacteries-se-melent-du-calcul-des-ordinateurs)
- Member of the Program Committee for The 2008 International Conference on Reconfigurable Computing and FGPAs, Reconfig 2008, Dec. 5-8, Cancun, Mexico (special track "Bioinspired reconfigurable computing systems")
- Review editor for the journal “Frontiers in Neurorobotics” (<http://frontiersin.org/neuroscience/user.do?actionType=JournalIssues&displayJournalPage=13&journalId=13>)
- Committee member for J.-B. Rouquier’s PhD thesis: “Robustesse et emergence dans les systemes complexes: le cas de automates cellulaires” (Dir. M. Morvan), Computer Science, ENS Lyon, 08 Dec. 2008, Lyon.

Albert Cohen

- Co-organization (with Marc Shapiro from INRIA Rocquencourt and Jean Roman from INRIA Bordeaux) of the INRIA Massively Multicore and Manycore (IMMM) Days, February 4 and 5 2008. 100-150 attendees expected, covering core research and technology as well as application domains impacted by manycore processors.
- Editor of the special issue of the Transactions on High Performance and Embedded Architectures and Compilers (HiPEAC Journal) for the best papers of the SHCMP workshop.
- Program committee member of IEEE conf. on Parallel Architectures and Compilation Techniques (PACT'08 and PACT'09).
- Program committee member of the HiPEAC'09 conference
- Program committee member of a CGO'08 workshop on the Open64 compiler.
- Program committee member of an ISCA'08 workshop (SHCMP'08) on Software Tools for Multicores.
- Thesis proposal committee of Frédéric De Mesmay, Carnegie Mellon University, USA, February 2009.
- Habilitation thesis committee of Henri-Pierre Charles, University of Versailles, December 2008.
- PhD thesis committee of Peter Faber, University of Passau, DE, February 2008.
- PhD committee of Harald Devos, Ghent University, BE, January 2008.
- PhD committee of Anna Beletka, Polytecnico di Milano, IT, March 2008.
- PhD thesis committee (reviewer) of Minhaj Khan, University of Versailles, June 2008.
- PhD thesis committee (reviewer) of Jamel Tayeb, University of Valenciennes (and Intel Portland), December 2009.
- PhD thesis committee (examiner) of Lamia Djoudi, University of Versailles, January 2009.

Christine Eisenbeis

- Software and Compilers for Embedded Systems, SCOPES' 2008, March 2008, Munich, Germany.
- Software and Compilers for Embedded Systems, SCOPES' 2009, April, 2009, Nice, France.
- IFIP International Conference on Network and Parallel Computing (NPC 2008), September 2008, Shanghai, China.

Grigori Fursin

- Program Committee Member of the IEEE International Parallel and Distributed Processing Symposium (IPDPS'08, Miami, Florida, USA);
- Program Committee Member and organizer of the Workshop on Statistical and Machine learning approaches applied to ARchitectures and compilaTion (SMART'08, Goteborg, Sweden, <http://www.hipeac.net/smart-workshop-08.html>);
- Program Committee Member of the 2007 International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES'07, Salzburg, Austria);
- Chair of the 1st Workshop on Statistical and Machine learning approaches applied to ARchitectures and compilaTion (SMART'07, Ghent, Belgium, <http://www.hipeac.net/smart-workshop-07.html>);

Olivier Temam

- IEEE/EDA Design and Automation in Europe (DATE), 2008.

- ACM/IEEE International Symposium on Micro-Architecture (MICRO), 2008.
- HiPEAC'08, International Conference on High-Performance Embedded Architectures and Compilers, 2008.
- Workshop on Modeling, Benchmarking and Simulation (MoBS), 2008.
- ACM/IEEE International Symposium on Computer Architecture (ISCA), 2009.
- HPCA, High-Performance Computer Architecture, 2009.
- Steering committee member and co-organizer of the Rapido workshop at the 2009 HiPEAC Conference.
- Associate editor of the HiPEAC Transactions.

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).

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

Grigori Fursin was an organizer of the Master 2 course "Future Computing Systems" http://fursin.net/research_teaching_m2r_course_2008_2009.html in the Fall 2008 semester.

Grigori Fursin gave lectures on “Continuous adaptive iterative compilation and machine learning techniques” in the Master 2 “Recherche” of Computer Science of University of Paris-Sud. in the Fall 2007 semester.

Olivier Temam teaches a computer architecture course at Ecole 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.

8.3. Workshops, seminars, invitations

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

Cédric Bastoul

- Participation to LCTES 2008 (June 12-13, Tucson, Arizona), ACM SIGPLAN/SIGBED 2008 Conference on Languages, Compilers, and Tools for Embedded Systems;
- Participation to PLDI 2008 (June 7-11, Tucson, Arizona), ACM SIGPLAN International Conference on Programming Language Design and Implementation;
- Presentations to the GRAPHITE Workshop (AMD Campus November 16-17, Austin, Texas), Research and Development on the Polyhedral Framework of GCC;

Hugues Berry

- Entretiens de Cargese du Reseau National des Systèmes Complexes, Cargese, Corsica, France, Feb. 25-29, 2008.
- EIEFB: Ecole CNRS-INRA interdisciplinaire d'échanges et de formation en biologie, Berder Island (Morbihan, France), March 20 - April 5, 2008
- 5th European Conference on Complex Systems, Jerusalem, Israel, Sept. 14-19, 2008

- 1st American - French kavli Frontiers of Science symposium, Roscoff, France, Nov. 20-22, 2008
- "Two examples of multi-agent simulation for cell biochemistry in crowded environments", Sèminaires SMABio, Programme d'Epigènomique, Gènopole, Evry, January 25th, 2008.
- "Influence of Hebbian learning on the dynamics, topology and function of chaotic recurrent neural networks", Journèes du GdR Dycoec ("Dynamique et controle des ensembles complexes"), Nice, February 05-06, 2008
- "Individual-based simulations for cell biochemistry in crowded environments", Sèminaire matinal de l'Institut des Systèmes Complexes de Paris (ISC-PIF), February 22nd, 2008
- "Individual-based simulations for cell biochemistry in crowded environments", Sèminaire de l'Equipe Contraintes, May 13, 2008

Olivier Certner

- Poster presentation at the ACACES 2007 Summer School (July, 9th–13th, 2007, L'Aquila, Italy). Title: "CAPSULE: An Environment to Ease Parallel Programming".
- CAPSULE presentation at a SARC European Project meeting (September, 25th–28th 2007, Delft, Nederland)

Sylvain Girbal

- UNISIM tutorial at Philips NXP. 3 days tutorial in Eindhoven, Netherland on December 18-20 2006.
- A full day tutorial on UNISIM presented at HiPEAC 2007 conference in January 2007, Ghent Belgium.
- A full day tutorial on UNISIM at INRIA Futurs, France in April 2007.
- A full week tutorial on UNISIM presented in Delft, Netherland for the SARC consortium.

Grigori Fursin

- 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.

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", vol. 16, 2006, 013110, <http://hal.inria.fr/inria-00000109/en/>.
- [3] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *N-Synchronous 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, n° 3149, Springer-Verlag, August 2004, 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.
- [6] 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>.
- [7] 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.
- [8] 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.
- [9] 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, n° 3793, Springer-Verlag, November 2005, p. 218–232, <http://www-rocq.inria.fr/~acohen/publications/PCS05.ps.gz>.
- [10] 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.

Year Publications

Doctoral Dissertations and Habilitation Theses

- [11] D. BARTHO. *Contributions to code optimization and high performance library generation*, Habilitation Thesis, Univ. Versailles St Quentin, January 2008.

- [12] H. BERRY. *Modelling Complex Biological Systems: Examples in Computational Cell Biology, Computational Neuroscience and applications to Computer Science*, Habilitation Thesis, University Paris-Sud 11, Orsay, France, June 2008.
- [13] P. PALATIN. *Exploitation pratique et efficace du parallélisme sur processeurs multi-cœurs*, PhD Thesis, University Paris-Sud 11, Orsay, France, September 2008.
- [14] B. SIRI. *Interactions dynamique-topologie-apprentissage dans les réseaux de neurones complexes bioinspirés*, PhD Thesis, University Paris-Sud 11, Orsay, France, December 2008.

Articles in International Peer-Reviewed Journal

- [15] P. CARRIBAULT, S. ZUCKERMAN, A. COHEN, W. JALBY. *Deep jam: Conversion of coarse-grain parallelism to fine-grain and vector parallelism*, in "Journal of Instruction-Level Parallelism", 2008.
- [16] F. GRUAU, C. EISENBEIS, L. MAIGNAN. *The foundation of self-developing blob machines for spatial computing*, in "Physica D: Nonlinear Phenomena", Special issue: "Novel Computing Paradigms: Quo Vadis?", vol. 237, n^o 1, July 2008, p. 1282-1301.
- [17] C. LAVELLE, H. BERRY, G. BESLON, F. GINELLI, J.-L. GIAVITTO, Z. KAPOULA, A. LE BIVIC, N. PEYRIERAS, O. RADULESCU, A. SIX, V. THOMAS-VASLIN, P. BOURGINE. *From molecules to organisms: towards multiscale integrated models of biological systems*, in "Theoretical Biology Insights", vol. 1, 2008.
- [18] B. SIRI, H. BERRY, B. CESSAC, B. DELORD, M. QUOY. *A mathematical analysis of the effects of Hebbian learning rules on the dynamics and structure of discrete-time random recurrent neural networks*, in "Neural Computation", vol. 20, 2008, p. 2937-2966.

International Peer-Reviewed Conference/Proceedings

- [19] M. BACHIR, S. TOUATI, A. COHEN. *Post-pass Periodic Register Allocation To Minimise Loop Unrolling*, in "ACM proceedings of the Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES 2008), Tucson, Arizona", June 2008.
- [20] O. CERTNER, Z. LI, P. PALATIN, O. TEMAM, F. ARZEL, N. DRACH. *A Practical Approach for Reconciling High and Predictable Performance in Non-Regular Parallel Programs*, in "First Workshop on Programmability Issues for Multi-Core Computers (MULTIPROG), within HiPEAC International Conference, Göteborg, Sweden", Jan 2008.
- [21] O. CERTNER, Z. LI, P. PALATIN, O. TEMAM, F. ARZEL, N. DRACH. *A Practical Approach for Reconciling High and Predictable Performance in Non-Regular Parallel Programs*, in "DATE 2008, Munich, Germany", march 2008, p. 740–745.
- [22] G. FURSIN, C. MIRANDA, O. TEMAM, M. NAMOLARU, E. YOM-TOV, A. ZAKS, B. MENDELSON, P. BARNARD, E. ASHTON, E. COURTOIS, F. BODIN, E. BONILLA, J. THOMSON, H. LEATHER, C. WILLIAMS, M. O'BOYLE. *MILEPOST GCC: machine learning based research compiler*, in "Proceedings of the GCC Developers' Summit", June 2008.
- [23] N. INGLART, S. NIAR, A. COHEN. *Hybrid performance analysis to accelerate compiler optimization space exploration for in-order processors*, in "2nd Workshop on Statistical and Machine learning approaches to ARchitectures and compilaTion (SMART'08), Göteborg, Sweden", January 2008.

- [24] F. JIANG, H. BERRY, M. SCHOENAUER. *Supervised and Evolutionary Learning of Echo State Networks*, in "10th International Conference on Parallel Problem Solving From Nature, PPSN-2008, Dortmund, Germany", September 2008.
- [25] F. JIANG, H. BERRY, M. SCHOENAUER. *Unsupervised Learning of Echo State Networks: Balancing the Double Pole*, in "Genetic and Evolutionary Computation Conference, GECCO-2008, Atlanta, GA, USA", July 2008.
- [26] M. A. KHAN, H.-P. CHARLES, D. BARTHOU. *Optimizing Code through Iterative Specialization*, in "ACM Symposium on Applied Computing, New York", 2008, p. 206–210.
- [27] J. NAUDÉ, S. GENET, H. BERRY, J. PAZ, B. DELORD. *A formalization of the computational impact of intrinsic plasticity*, in "NeuroComp 2008", 2008, p. 19–26.
- [28] 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.
- [29] 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.

Workshops without Proceedings

- [30] M. BAHİ, 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.

Scientific Books (or Scientific Book chapters)

- [31] H. BERRY. *Nonequilibrium phase transition in scattered cell communities coupled by auto/paracrine-like signalling*, in "Phase transitions in cell biology", G. POLLACK, W.-C. CHIN (editors), Springer Verlag, 2008, p. 23–41.

Research Reports

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

References in notes

- [33] *FLEXUS*, <http://www.ece.cmu.edu/~simflex/flexus.html>.
- [34] *GCC ICI: Interactive Compilation Interface*, <http://gcc-ici.sourceforge.net>.
- [35] *European Network of Excellence on High-Performance Embedded Architecture and Compilation (HiPEAC)*, <http://www.hipeac.net>.

- [36] *EU Milepost project (Machine Learning for Embedded ProgramS opTimization)*.
- [37] *SystemC v2.0.1 Language Reference Manual*, 2003, <http://www.systemc.org/>.
- [38] *UNISIM: UNIted SIMulation environment*, <http://unisim.org>.
- [39] 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.
- [40] 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.
- [41] 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>.
- [42] 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.
- [43] 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>.
- [44] 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.
- [45] H. BERRY, D. GRACIA PÉREZ, O. TEMAM. *Chaos in computer performance*, in "Chaos", vol. 16, 2006, 013110, <http://hal.inria.fr/inria-00000109/en/>.
- [46] 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.
- [47] H. BERRY, M. QUOY. *Structure and dynamics of random recurrent neural networks*, in "Adaptive Behavior", vol. 14, 2006, p. 129-137.
- [48] H. BERRY, O. TEMAM. *Modeling Self-Developping Biological Neural Network*, in "Neurocomputing", vol. 70, n^o 16-18, 2007, p. 2723–2734.
- [49] 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.

- [50] 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", vol. 26, n^o 4, 2006, p. 52–60, <http://dx.doi.org/10.1109/MM.2006.82>.
- [51] R. BLUMOFFE, 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://citeseer.ist.psu.edu/blumoffe95cilk.html>.
- [52] D. BURGER, T. M. AUSTIN. *The SimpleScalar tool set, version 2.0*, in "SIGARCH Comput. Archit. News", vol. 25, n^o 3, 1997, p. 13–25, <http://doi.acm.org/10.1145/268806.268810>.
- [53] 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)", To appear, October 2006.
- [54] 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.
- [55] 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", vol. 33, n^o 4, 2005, p. 54–63, <http://doi.acm.org/10.1145/1105734.1105742>.
- [56] 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>.
- [57] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *N-Synchronous 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>.
- [58] 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, vol. 2400, Springer-Verlag, August 2002, <http://www-rocq.inria.fr/~acohen/publications/CGKCDF02.ps.gz>.
- [59] 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>.
- [60] A. COHEN, S. GIRBAL, O. TEMAM. *A Polyhedral Approach to Ease the Composition of Program Transformations*, in "Euro-Par'04, Pisa, Italy", LNCS, n^o 3149, Springer-Verlag, August 2004, p. 292–303, <http://www-rocq.inria.fr/~acohen/publications/CGT04.ps.gz>.
- [61] 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.

- [62] K. D. COOPER, D. SUBRAMANIAN, L. TORCZON. *Adaptive Optimizing Compilers for the 21st Century*, in "J. Supercomput.", vol. 23, n^o 1, 2002, p. 7–22, <http://dx.doi.org/10.1023/A:1015729001611>.
- [63] L. DAGUM, R. MENON. *OpenMP: An Industry- Standard API for Shared- Memory Programming*, in "IEEE COMPUTATIONAL SCIENCE & ENGINEERING", 1998, p. 46-55.
- [64] 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.
- [65] 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", vol. 35, n^o 2, 2002, p. 68-76.
- [66] 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.
- [67] P. FEAUTRIER. *Dataflow Analysis of Array and scalar references*, in "Int. J. of Parallel Programming", vol. 20, n^o 1, 1991, p. 23-53.
- [68] P. FEAUTRIER. *Some efficient solutions to the affine scheduling problem I. One-dimensional time*, in "Int. J. of Parallel Programming", vol. 21, n^o 5, 1992, p. 313-347.
- [69] P. FEAUTRIER. *Some efficient solutions to the affine scheduling problem II. Multi-dimensional time*, in "Int. J. of Parallel Programming", vol. 21, n^o 6, 1992, 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, n^o 3793, Springer-Verlag, November 2005, p. 29–46, <http://hal.inria.fr/inria-00001054/en/>.
- [73] G. FURSIN, A. COHEN, M. F. P. O'BOYLE, O. TEMAM. *Quick and practical run-time evaluation of multiple program optimizations*, in "Trans. on High Performance Embedded Architectures and Compilers", vol. 1, n^o 1, 2006, p. 13-31.
- [74] G. FURSIN, A. COHEN, M. F. P. O'BOYLE, O. TEMAM. *Quick and practical run-time evaluation of multiple program optimizations*, in "Trans. on High Performance Embedded Architectures and Compilers", vol. 1, n^o 1, January 2007, p. 13-31.

- [75] 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.
- [76] 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.
- [77] J.-L. GIAVITTO, O. MICHEL. *MGS: a Rule-Based Programming Language for Complex Objects and Collections*, in "Electronic Notes in Theoretical Computer Science", vol. 59, n^o 4, 2001.
- [78] S. GIRBAL, G. MOUCHARD, A. COHEN, O. TEMAM. *DiST: A Simple, Reliable and Scalable Method to Significantly Reduce Processor Architecture Simulation Time*, in "Intl. Conf. on Measurement and Modeling of Computer Systems, ACM SIGMETRICS'03, San Diego, California", June 2003, <http://doi.acm.org/10.1145/781027.781029>.
- [79] 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", Accepted with minor revisions, 2006.
- [80] 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.
- [81] 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/>.
- [82] 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/>.
- [83] D. GRACIA PÉREZ, H. BERRY, O. TEMAM. *The Practicality Dimension of Sampling*, in "IEEE micro", (in press), 2006.
- [84] 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>.
- [85] 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>.
- [86] 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.
- [87] M. H. HALL, S. P. AMARASINGHE, B. R. MURPHY, S.-W. LIAO, M. 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>.
- [88] 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.
- [89] 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.
- [90] 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.
- [91] J. HUSELIUS. *Debugging Parallel Systems: A State of the Art Report*, Technical report, n^o 63, Mälardalen University, Department of Computer Science and Engineering, September 2002, <http://citeseer.ist.psu.edu/huselius02debugging.html>.
- [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/95307.html>.
- [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.", vol. 3, 1986, 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] 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.
- [98] D. B. LOVEMAN. *High Performance Fortran*, in "IEEE Parallel Distrib. Technol.", vol. 1, n^o 1, 1993, p. 25-42, <http://dx.doi.org/10.1109/88.219857>.
- [99] 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", vol. 35, n^o 2, 2002, p. 50-58, <http://doi.ieeecomputersociety.org/10.1109/2.982916>.

- [100] 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", vol. 33, n^o 4, 2005, p. 92–99, <http://doi.acm.org/10.1145/1105734.1105747>.
- [101] 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.
- [102] A. MONSIFROT, F. BODIN, R. QUINIOU. *A machine learning approach to automatic production of compiler heuristics*, in "Proc. AIMSAS", LNCS 2443, 2002, p. 41-50.
- [103] G. MOUCHARD, D. GRACIA PÉREZ, O. TEMAM. *FastSysC: A Fast Simulation Engine*, in "Design, Automation and Test in Europe (DATE), Paris, France", 2004, <http://hal.inria.fr/inria-00001108>.
- [104] M. O'BOYLE, P. KNIJNENBURG, G. FURSIN. *Feedback Assisted Iterative Compiilation*, in "Parallel Architectures and Compilation Techniques (PACT'01)", IEEE Computer Society Pres, October 2001.
- [105] 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.
- [106] 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>.
- [107] 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://gala.univ-perp.fr/~dparello/publis/on_increasing_architecture_awareness.pdf.
- [108] T. POGGIO, C. R. SHELTON. *Machine Learning, Machine Vision, and the Brain*, in "The AI Magazine", vol. 20, n^o 3, 1999, p. 37–55, <http://citeseer.ist.psu.edu/poggio99machine.html>.
- [109] 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 Developer's Summit, Ottawa, Canada", June 2006.
- [110] 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.
- [111] W. PUGH. *The Omega test: A fast and practical integer programming algorithm for dependence analysis*, in "Comm. of the ACM", vol. 8, 1992, p. 102-114.
- [112] 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.

- [113] SIA. *Semiconductor Industry Association 2005 roadmap, section on Emerging Research Devices*, 2005, <http://www.sia-online.org/>.
- [114] S. SALVA, C. DELAMARE, C. BASTOUL. *Web Service Call Parallelization Using OpenMP*, in "3rd International Workshop on OpenMP, Beijing, China", LNCS, June 2007.
- [115] 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.
- [116] 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.
- [117] M. SMITH. *Overcoming the challenges to feedback-directed optimization*, in "Proc. ACM SIGPLAN Workshop on Dynamic and Adaptive Compilation and Optimization (Dynamo'00)", 2000.
- [118] C. SZYPERSKI. *Component Software: Beyond Object-Oriented Programming*, Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.
- [119] 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.
- [120] W. THIES, M. KARZMAREK, M. GORDON, D. MAZE, J. WONG, H. HO, M. BROWN, S. AMARASINGHE. *StreamIt: A Compiler for Streaming Applications*, MIT-LCS Technical Memo TM-622, Cambridge, MA, December 2001, <http://citeseer.ist.psu.edu/article/thies01streamit.html>.
- [121] S. TRIANTAFYLIS, M. VACHHARAJANI, N. VACHHARAJANI, D. I. AUGUST. *Compiler optimization-space exploration*, in "Proc. International Symposium on Code Generation and Optimization", 2003, p. 204–215.
- [122] 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.
- [123] 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>.
- [124] 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.
- [125] M. VOSS, R. EIGENMANN. *ADAPT: Automated de-coupled adaptive program transformation*, in "Proc. ICPP", 2000.
- [126] 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.

-
- [127] 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.
- [128] M. WOLF, M. LAM. *A loop transformation theory and an algorithm to maximize parallelism*, in "IEEE Transactions on Parallel and Distributed Systems", vol. 2, n^o 4, 1991, p. 430-439.
- [129] S. YEHAIA, 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.