INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

Team Alchemy

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

Futurs

THEME COM

Activity Report

2005

Table of contents

1. Team

Head of project team
Olivier Temam [Research Director (DR) Inria]

Administrative assistants
Stéphanie Meunier [TR Inria, with Gemo]

Staff members, Inria
Hugues Berry [Research Associate (CR) Inria, on secondment from the Cergy-Pontoise University]
Albert Cohen [Research Associate (CR) Inria]
Christine Eisenbeis [Research Director (DR) Inria]
Grigori Fursin [Postdoctoral Fellow]
Claire Pagetti [Postdoctoral Fellow, until June 30th, 2005]

Staff members, Paris-11 University
Cédric Bastoul [Assistant Professor, since October 1st, 2005]
Julien Cohen [1/2 ATER, until August 31st, 2005]
Frédéric Gruau [Assistant Professor]
David Parello [1/2 ATER, University of Paris-Sud, until August 31th, 2005]

Technical staff
Sylvain Girbal [Expert engineer, since September 1st, 2005]
Marc Gonzalez-Sigler [Associate engineer, until August 31st, 2005]

Ph. D. students
Patrick Carribault [Bull fellowship (Cifre), University of Versailles-Saint-Quentin]
Sébastien Donadio [University of Versailles-Saint-Quentin]
Sébastien Favre [Inria scholarship]
Sylvain Girbal [CEA scholarship, until Janvier 31st, 2005, then Inria scholarship, until August 31st, 2005, University of Paris-Sud,]
Daniel Gracia Pérez [Grant of the University of Paris-Sud, until November 30th, 2005]
Yves Lhuillier [MENRT scholarship, until September 30th, 2005, University of Paris-Sud]
Pierre Palatin [CNRS BDI scholarship]
Sebastian Pop [École Nationale Supérieure des Mines de Paris]
Anil Thimma Reddy [MENRT scholarship, University of Paris-Sud]
Nicolas Vasilache [MENRT scholarship, University of Paris-Sud]

Student interns
Jean-Charles Baritaux [Summer internship, École Polytechnique, April to July, 2005]
Gérard Emond [Summer internship, École Polytechnique, April to July, 2005]
Marek Doniec [Summer internship, École Polytechnique, April to August, 2005]
Noureddine Oulagha [Master of computer science, University of Paris-Sud, March to September, 2005]

External collaborators
Cédric Bastoul [Full time ATER, Université de Clermont-Ferrand, until August 31st, 2005]
Benjamin Dauvergne [PhD student, Tropics project-team, Inria Sophia-Antipolis]
Nathalie Drach [Professor, Paris-6 University]

2. Overall Objectives

2.1. Overall Objectives

ALCHEMY is an joint Inria-LRI (CNRS and University of Paris-Sud) team created in fall 2003 as a result of the merger of the former Inria group A3 on compilation, and the Paris South University/CNRS group Architecture on processor architecture. It is located in Orsay.

The general research topics of the ALCHEMY group are architectures, languages and compilers for high-performance embedded and general-purpose processors. ALCHEMY investigates alternative solutions to incremental architecture and compiler optimizations for high-performance general-purpose and embedded processors. The increasing complexity of high-performance processor architectures has two main consequences. (1) In the short term, the inability to embed a sufficiently accurate architecture model in compilers makes it increasingly hard to generate efficient program optimizations, and thus to achieve high sustained performance. (2) In the long term, the architecture complexity makes it increasingly hard to scale processor architectures, more exactly to translate technology improvements into higher sustained performance. We are developing two approaches respectively corresponding to the short-term and long-term issues outlined above.

ALCHEMY stands for Architectures, Languages and Compilers to Harness the End of Moore Years, meaning both the complex but traditional processor architectures implemented using the current photolithographic processes, and novel architecture/language paradigms compatible with future and alternative technologies.

ALCHEMY's research themes are:

- **Iterative compilation:** For the short term, we are investigating program optimization techniques relying on dynamical analysis, i.e., the detailed analysis of the program behavior on the architecture during execution. Such techniques are usually called dynamic or iterative compilation. We are about to disseminate this research effort through an iterative compilation environment as part of the Center for Program Tuning that we are currently setting up.
- **Combined language/architecture approach:** For the long term, we consider that both excessive processor architecture complexity and low sustained performance are rooted in the current architecture/programming model itself. More precisely, the current model fails in two ways: passing enough program semantics to the compiler and the architecture, and efficiently managing the increasing chip space brought by technology. For that purpose, we are investigating combined architecture/language approaches that can meet the two abovementioned properties. We are investigating languages that can pass the necessary semantic to the architecture and the compiler without sacrificing the ease of programming. As a result of the richer semantic, both the architecture and the compiler are simpler and potentially more effective. Moreover, we are investigating simple and regular architectures with self-organizing properties that can thus scale easily with technology. This long-term research work is strongly tied to technology issues, and for that reason, we are studying in parallel alternatives to current photolithographic silicon-based processes and their potential impact on architecture and programming paradigms.
- **Transversal methodology activity:** For both approaches, we are also conducting a transversal methodology activity to develop processor simulators used for both program optimization and architecture purposes. This activity focuses on fast development and execution methods for processor simulators.

3. Scientific Foundations

3.1. Scientific Foundations

The goal of Alchemy is to tackle both of the abovementioned short-term and long-term issues which respectively threaten the efficient exploitation of current architectures, and the evolution of future architectures. The first approach (short-term research on program optimizations) aims at coming up with a software environment that can be used both in research and the industry for simplifying the task of optimizing programs on complex processor architectures. The role of the second approach is to investigate possibly radical modifications (if necessary) of the current architecture and programming models in order to come up with architectures which can scale up more easily and are compatible with upcoming (new) technologies. We are less concerned with medium-term architecture research for practical reasons: while incremental architecture modifications can certainly improve performance, we believe they will not be sufficient to let architectures scale up smoothly and regularly again; on the practical side, processor manufacturers already have very skilled architecture research and development groups for coming up with medium-term innovations for their next-generation products; on the other hand, processor manufacturers cannot always afford to investigate very long-term and risky alternatives, and for them to accept such radical changes, they have to be anticipated long in advance. We believe that is a core role for academic researchers in this domain.

At the moment, the research activity on the short-term program optimization issues has been dominant in Alchemy; the longer-term architecture/language research activity as well as our activities on methodology are progressively accounting for a larger share of Alchemy's effort.

3.1.1. A practical approach to program optimizations for complex architectures

The principles of our approach is to heavily rely on dynamic (run-time) analysis as a way to overcome the architecture complexity bottleneck. Using an iterative compilation approach, we want to learn over executions the appropriate method for optimizing a program based on detailed low-level information on the behavior of the program on the architecture. In the recent years, iterative optimization has received increased attention thanks to the OCEANS LTR Esprit project [1] and researchers like Mike O'Boyle at University of Edinburgh, UK or Keith Cooper at Rice University. In these research works, iterative optimization is essentially used to fine-tune the parameters of program optimizations applied to restricted code constructs; moreover, the goal of most research works is to show that the approach may work by exhaustively searching the parameter space, not a practical approach for using iterative optimization. We want to address the issues pertaining practical applications of iterative optimization, and to extend it to whole-program optimization by empowering iterative optimization with the ability not only to select program optimization parameters but program optimizations themselves and their location of application in the program. We want to show that we can both achieved significant performance improvements and significantly reduce the program optimization effort by making it more systematic.

Toward practical applications of iterative optimization. This research work is divided in several steps and projects. (1) First, to some extent, we have started "from scratch". Instead of the top-down approach of compilers which are progressively augmented with information from the architecture as in current profile-based, iterative or dynamic compilation techniques, we have adopted a bottom-up approach to the architecture complexity issue: assuming we know everything about the behavior of the program on the architecture (using cycle-level processor simulators), what can we do to improve its performance? Based on extensive analysis of programs behaviors on a complex processor architecture, we have designed a systematic and iterative optimization process [13], [12]. While it is not yet implemented as a fully automatic iterative environment, it is systematic, and it has already been (and is still being) used successfully at HP France for the task of quickly optimizing programs on complex processors for prospective customers (on the Alpha for the moment; extension to x86 is planned).

The second big issue is how to let iterative optimizations control the application of program transformations themselves. If program optimizations have to be applied/selected automatically, the search space now includes not only optimization parameters but optimization themselves, and especially compositions of optimizations. While compilers include rigid sequences of optimizations, an iterative process can seek the best sequence for each code section. However, for that purpose, we must be able to compose long sequences of program transformations, and the current syntactic implementation of program transformations strongly limits that ability. Therefore, we are currently developing and implementing a framework based on the polyhedral representation of programs (and transformations) for easily composing very long sequences of program transformations [15].

The third issue is the software environment for scanning the search space, apply program transformations, collect feedback information and deduce the most appropriate next transformations to be tested. Moreover, for a practical application of iterative optimization, performance information deduced from one program execution (i.e., one data set) must be somehow exploited for other/next executions (i.e., other data sets). We have gathered preliminary results on iterative low-level optimizations for embedded processors which tend to show this approach is feasible [8], but we still have to confirm it with larger scale experiments. Besides that, we have a prototype version of our software environment currently running and for which we are now investigating search space strategies.

Increasingly, we are now moving toward automatizing the whole process. A first step in that direction is to show that, for simple low-level optimizations, it is possible to directly rely on the detailed architecture description implicitly embedded in a cycle-level processor simulator to replace the static analysis of compilers. We have shown that a modified simulator of an embedded VLIW processor can automatically schedule assembly instructions, much like complex and costly Out-of-Order superscalar processors, but without any additional hardware cost [8]. We are also investigating applications of this approach to the specialization/idiomization of embedded processors [14]. Next steps include fully automated runs of the abovementioned software environment with the prospect of finding complex compositions of high-level program transformations.

Related activities: Esprit LTR MHAOTEU [9] and OCEANS [1], Digital/Compaq/HP France grants, ACI grant, RNTL COP; in the past 4 years, 4 PhD students have been or are still working on these topics.

3.1.2. Revisiting the processor architecture/programming approach

In the long term, we consider that both excessive processor architecture complexity and low sustained performance are rooted in the current architecture/programming model itself. More precisely, the current model fails in two ways: passing enough program semantics to the compiler and the architecture, and efficiently managing the increasing chip space brought by technology.

Passing additional semantic to the architecture using appropriate languages. For the first part, we can notice that both compiler and architecture optimizations often implement a form of *reverse-engineering*: compiler optimizations attempt to dig up program properties, and architecture optimizations often seek regularity properties in the program (branch prediction) or data (caches, prefetching). 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/he had an adequate programming language; for instance, more explicit information on the nature of data structures would help understanding both the control and data flow. As a result, both the architecture and the compiler would be simpler and potentially more effective.

Domain-specific synchronous language for high-performance video processing. We revisit 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 (CAT-IP team) and with Marc Pouzet from LIP6. In particular, we extend the classical clock calculus and data types of the Lucid Synchrone synchronous language to address the following issues: natural description of operations on compound types with multiple stream semantics (beyond FIFO, hierarchy); handling relaxed synchronous operators like jittering and bursty streams within synchronous bounds; combining time-triggered scheduling of predictable computation kernels (as in systolic architectures) with data-flow scheduling of data-dependent operators; partitioning computations between software threads on general purpose cores and custom hardware units, and mapping media streams to statically allocated buffers. We focus on warrantable (as opposed to best-effort) usage of hardware resources with respect to real-time constraints.

Related activities: Esprit LTR OCEANS [1], Philips grants and Marie-Curie postdoc fellowship in the former SANDRA project [2]; in the past 4 years, 2 PhD students and 2 postdocs have been or are still working on these topics.

Spatial architectures and programming. For the second part, it is likely that the centralized control used in current processor architectures may no longer be acceptable with a very large chip space. One of the main challenges then becomes the management of one or several programs on a very large space. We are investigating very regular/homogeneous architectures, such as large arrays of processors. Note that tiled architectures have received increased attention in the past few years as a testimony to the increasing difficulty of scaling up current superscalar processors . However, while several researchers agree that regular and space-oriented architectures are much easier to scale, most researchers still attempt to program them with conventional programming styles and approaches. We are trying to explore different programming styles which are, possibly, more compatible with these space-oriented architectures. We are particularly investigating programming approaches which break down a program into a set of coordinated "local" actions (local to a node and data), and which relieve the programmer from thinking through the global management of his program on the architecture [10], [16]. The architecture is augmented with a support aware of this local program breakdown and it actually manages the program execution, instead of the compiler or the programmer. We advocate that, provided we strike the right balance between the architecture, compiler and *user* effort, it is possible to unveil relatively coarse-grain parallelism (coarser than ILP) and to take advantage of it without excessively complex architectures or compilers.

We call this combination of local programming and hardware support "self-organized" architectures. Such self-organized systems are in fact widespread in nature, especially in biological systems, and part of our (longest-term) work consists in understanding/extracting the simple rules used in complex natural systems (in cooperation with researchers in biology) that can serve to implement desired properties in computing systems. For instance, pressure and elasticity can respectively express the necessity to be allocated more space and to get closer during communications. While this research direction may seem futuristic, and while part of our objective is effectively to investigate alternative computing models, possibly compatible with future technologies like molecular electronics or biologically-assembled chips, another shorter-term goal is also to apply such local rules to very contemporary architectures, such as SMTs and CMPs.

Related activities: 2 PhD students are working on thesis topics.

3.1.3. A transversal research direction: methodology and simulation

Simulators are needed for both architecture and program optimization research. For architecture research, they serve to implement detailed cycle-level models to evaluate new ideas; for program optimization research, they serve to get a better understanding of the detailed behavior of program on processor architectures. As a result, they are key tools for our research group. However, as processor architecture and program complexity increase, so does the development and execution time of these simulators. Therefore, we have progressively invested on methodology issues for the sake of efficiency of our other research activities. The main principles of our approach is to argue for the development of *modular* simulators, and for the increased sharing and reuse of researchers expertise and efforts through a common library of simulator components 5.3.

Simulators are used in most processor architecture research works, and, while most research papers include some performance measurements (often IPC and more specific metrics), these numbers tend to be distrusted because the simulator associated with the newly proposed mechanism is rarely publicly available, or at least not in a standard and reusable form; and as a result, it is not easy to check for design and implementation hypotheses, potential simplifications or errors. However, since the goal of most processor architecture research works is to *improve* performance, i.e., do better than previous research works, it is rather frustrating not to be able to clearly quantify the benefit of a new architecture mechanism with respect to previously proposed mechanisms. Many researchers wonder, at some point, how their mechanism fares with respect to previously proposed ones and what is the best mechanism, at least for a given processor architecture and benchmark suite (or even a single benchmark); but many consider, with reason, that it is excessively time-consuming to implement a significant array of past mechanisms based on the research articles only. We argue that, provided a few groups start populating a common library of modular simulator components, a broad and systematic quantitative comparison of architecture ideas may not be that unrealistic, at least for certain research topics and ideas; and we are in the process of developing such a library, called MicroLib. Because the main flaw of modular simulators is their poor speed of execution, we are also working on techniques to speed up simulation, using parallelization and sampling.

Initially intended for internal purposes only, several simulator models, components and tools are now being disseminated through a general web site (www.microlib.org), with some of the tools being rather heavily used (2000+ downloads for the PowerPC simulator on June 2004, several hundred downloads for other tools, several articles using/referencing these tools). Besides program optimization and architecture research, this work is also being applied to the software test of large embedded systems in collaboration with CEA.

Related activities: RNTL ATLAS, ACI grant. 2 PhD students have been or are working on these topics.

3.1.4. Beyond performance

We have recently started a new research project which is both fairly different but also fairly complementary with our mainstream projects. All the above projects are focused on the issue of “performance”, especially the performance of processors and systems. However, to a large extent, we are not sufficiently taking advantage of the knowledge we have in processor and system architecture. Because we know several years in advance what will be the capabilities of future processors, we can anticipate as well the possible structures of systems, and especially their *applications*. For instance, architects would have known several years in advance approximately when cheap processors or circuits would be capable of doing real-time MPEG-2 encoding, and thus when hard-drive based VCRs could become a reality, a product that may soon become widespread. We want to open up more to system architecture in the general sense, and especially its applications, and possibly propose either software or hardware prototypes if needs be. While this type of work may be considered borderline research and harder to publish, we also believe it is a healthy (and fun) exercise for a research group to think about the potential applications of its mainstream research projects. As much as possible, we want to consider these projects as federating projects within the research group where multiple faculty/PhDs can participate. We have currently started one first project on the evolution of the home PC, towards more O/S-level simplicity and the geographic distribution of peripherals and computing elements.

Related activities: HP France grant. 1 PhD student is working on these topics.

4. Application Domains

4.1. Application Domains

Why doing research on high-performance processors? The rapid evolution of electronic systems, whether consumer/specialized electronics or computers, is due, to a large extent to the rapid improvement of electronic circuit (VLSI) performance, and increasingly to the rapid improvement of processor performance. While computers are by nature tied to the notion of processor, consumer/specialized electronics increasingly rely on processors, either as a complement or as a replacement for ASICs, in order to increase the flexibility and time-to-market of their new products. And in order to achieve the high performance required by new embedded applications (e.g., multimedia applications, network processing,...), these processors are increasingly high-performance processors.

Does it matter in Europe? The United States largely dominate the computer industry, and especially the general-purpose processor industry (Intel, AMD, IBM) used in PCs, workstations and supercomputers. On the other hand, Europe has a very strong position in the embedded processor domain (ARM, Infineon, Philips, STMicroelectronics). Currently, there are still major technical differences between the processors manufactured by Intel and those manufactured by STMicroelectronics. However, the regular decrease of the cost/performance ratio is blurring the threshold between the computer industry (general-purpose processors) and the embedded system industry (embedded processors). Intel is now targeting the phone (and other embedded) market with processors (such as the XScale) which retain several innovations introduced in general-purpose processors (caches, longer pipelines, branch prediction, multimedia extensions,...), while the embedded processors designed by ST and Infineon (and other European companies) are now powerful enough to drive new products like the N-Gage phone game console by Nokia. There are multiple examples of this increasing confusion between general-purpose processors and embedded processors. For instance, the latest IBM supercomputer (Blue Gene/L) is based on a sophisticated embedded processor (PowerPC 440; partly due to power dissipation issues) and not their latest high-performance processor (the Power G5). On the other hand, IBM teamed up with Sony and Toshiba to design a new high-performance processor (general-purpose? embedded?) for the next PlayStation3. Thus, it is increasingly difficult to flag some processors as general-purpose or embedded. On the other hand, the increasing overlap between the general-purpose and embedded domains is clearly a rare opportunity for the European industry to enter the computer market again, and it is also a significant challenge as the computer industry is now targeting the embedded system market.

What is the impact on research? For the past 30 years, processor performance has apparently regularly and smoothly increased in parallel with the technology improvements. In fact these improvements both bred and result from a considerable increase in architecture complexity. For the industry, this complexity means that architecture design is an increasingly sophisticated process; not surprisingly American companies increasingly rely on PhD-level engineers to design such processors, while European companies are only starting to edge in that direction.

Scientifically, this complexity has two serious consequences. A short-term consequence is that it is increasingly difficult to generate an efficient program for such complex architectures. As a result, current architectures only exploit a fraction of their potential/peak performance; this is especially true for the more sophisticated general-purpose processors, but it is increasingly true for recent embedded processors. Current optimization techniques rely on static optimization (program analysis at compile-time only) which requires to embed a detailed architecture model in the compiler in order to generate a code which best matches the underlying architecture; naturally, as the architecture complexity increases, it is increasingly difficult to achieve and the resulting optimization efficiency does not increase as fast.

A long-term consequence of this complexity is that it is increasingly difficult to scale up such architectures. Each new component added to the architecture improves the overall architecture performance, but it also creates potential bottlenecks that will need to be addressed by updated or new mechanisms, and so on. The current approach essentially relies on speeding up a Von Neumann-like centralized architecture (a single/centralized processing unit, and a single memory), and to devote most of the available on-chip space to the new components required to achieve the targeted speed. Besides the increasing architecture hardships, technology may ultimately limit as well this speed-oriented approach. For both reasons, some researchers are investigating alternative approaches to exploit available on-chip space and to translate on-chip space into performance.

5. Software

5.1. Tuareg

Participant: Albert Cohen.

Extensive Environment for writing, running and debugging OCaml programs within (X)Emacs. Thousands of installations worldwide, distributed as part of Debian GNU/Linux.

5.2. DigLC2

Participants: Albert Cohen, Olivier Temam.

Gate-level simulator of the LC-2 microprocessor and computer architecture (*Little Computer 2* from Yale Patt and Sanjay Patel) [6]; dedicated to computer architecture and education; based on the free Chipmunk tool suite (DigLog).

5.3. MicroLib

Participants: Daniel Gracia Pérez, Gilles Mouchard, Pierre Palatin, Olivier Temam.

MicroLib is a library of modular simulator components freely distributed on a web site (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). In 2005, Microlib has evolved into the Fraternité project in collaboration with Princeton University (see sections 8.3 and 8.2). Details can be found at the Web site www.unisim.org - in construction.

5.4. FastSysC

Participants: Daniel Gracia Pérez, Gilles Mouchard.

FastSysC 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.

As part of our efforts on speeding up simulation execution, we have developed a tool for parallelizing simulators, 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.

5.5. GenISSLib

Participant: Gilles Mouchard.

GenISSLib is a builder of instruction set libraries. It allows the writer of emulators and simulators to write easy to read instruction set descriptions and associate functionalities to the instructions, to create a library of services that can be used to write emulators or simulators. This tool can be found at the MicroLib web page.

5.6. AlphaISS

Participant: Daniel Gracia Pérez.

AlphaISS is a Alpha 21264 program level emulator. It was built using the GenISSLib tool. This emulator can be found at the MicroLib web page.

5.7. PPCISS

Participant: Éric Renard.

PPCISS is a PowerPC 750 program level emulator. It was built using the GenISSLib tool. This tool is being updated to make it a full system emulator by Gilles Mouchard. This emulator can be found at the MicroLib web page.

5.8. OoOSysC

Participants: Daniel Gracia Pérez, Gilles Mouchard.

OoOSysC is a generic superscalar processor simulator, based on different architectures: Alpha, Intel Pentium 4 and AMD Athlon. It uses the Alpha 21264 instruction set. OoOSysC can be found at the MicroLib web page.

It is built in a modular environment based on SystemC (it can use FastSysC to increase simulation speed), and every component of the architecture is described as a module that connects with other modules through signals.

Additionally to the base architecture, different cache mechanisms have been implemented as modules that can easily replace the baseline cache modules.

5.9. DiST: Distributed (parallel) simulation of microprocessors with dynamic warmup and trace partitioning

Participants: Albert Cohen, Sylvain Girbal, Gilles Mouchard, Olivier Temam.

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

5.10. WRaP-IT/URUK

Participants: Cédric Bastoul, Albert Cohen, Sylvain Girbal, Marc Gonzalez-Sigler, David Parello, Olivier Temam, Nicolas Vasilache.

This work is one of the cornerstones of our *Center for Program Tuning* (RNTL project, 2004–2006) described in section 7.2. The main goal is to facilitate the expression and search of compositions of program transformations. This framework relies on a unified polyhedral representation of loops and statements. The key to our framework is to clearly separate the impact of each program transformation on the following three components: the iteration domain, the statements schedule and the memory access functions. Within this framework, composing a long sequence of program transformations induces no code explosion. As a result, searching for compositions of transformations is not hampered by the multiplicity of compositions, and ultimately, it is equivalent to testing different values of the matrices parameters in many cases. Our techniques have been implemented on top of the Open64/ORC compiler. In addition, we are beginning the design of a robust infrastructure for iterative optimization, based on machine learning techniques (operation research, e.g., genetic algorithms). This infrastructure distributes simulations, dynamic profiles, compilations, transformations, while interacting with a machine-learning component or with an expert user. Validation of these concepts and application of the tools will be a critical issue in the center for program tuning. Marc Gonzalez-Sigler, associate engineer at INRIA, has produced a large part of the platform effort to integrate our research tools into the free Open64 and PathScale EKO compilers. He also conducted systematic iterative optimization experiments with genetic algorithms.

5.11. CLooG

Participant: Cédric Bastoul.

CLOOG (*Chunky LOOp Generator*, http://www.cloog.org) is a free software and library generating codes for scanning Z-polyhedra. That is, it finds a code (e.g. in C, FORTRAN...) that reaches each integral point of one or more parametrized polyhedra. CLOOG has been originally written to solve the code generation problem for optimizing compilers based on the polytope model. Nevertheless it is used now in various area e.g. to build control automata for high-level synthesis or to find the best polynomial approximation of a function. While the user has full control on generated code quality, CLOOG is designed to avoid control overhead and to produce a very effective code. It is very *compilable* code oriented and provides powerful program transformation facilities. Mainly, it allows the user to specify very general schedules, e.g. where unimodularity or invertibility does not matter.

5.12. PIP

Participants: Cédric Bastoul, Paul Feautrier.

PIP/PIPLIB (http://www.piplib.org) is a parametric integer linear programming solver: it finds the lexicographic minimum (or maximum) in the set of integer points belonging to a convex polyhedron. The very big difference with well known integer programming tools like lp_solve or CPLEX is the polyhedron may depend linearly on one or more integral parameters. If the user asks for a non integral solution, PIP can give the exact solution as an integral quotient. The heart of PIP is the parametrized Gomory's cuts algorithm followed by the parametrized dual simplex method. It is used in many projects, mostly but not exclusively in automatic optimizing/parallelizing compilation (e.g. to compute data dependences).

6. New Results

6.1. Practical approach to program optimizations

6.1.1. Iterative optimization meets the polytope model

Participants: Albert Cohen, Sylvain Girbal, David Parello, Olivier Temam, Nicolas Vasilache.

Static cost models have a hard time coping with hardware components exhibiting complex run-time behaviors, calling for alternative solutions. Iterative optimization is emerging as a promising research direction, but currently, it is mostly limited to finding the parameters of program transformations or selecting whole optimization phases. One of the cornerstones of our *Center for Program Tuning* (RNTL project, *Centre d'Optimisation de Programmes*, 2003–2005) is to facilitate the expression and search of compositions of program transformations. Our framework relies on a unified polyhedral representation of loops and statements. The key is to clearly separate the impact of each program transformation on the following three components: the iteration domain, the statements schedule and the memory access functions [29]. Within this framework, composing a long sequence of program transformations induces no code explosion. As a result, searching for compositions of transformations is not hampered by the multiplicity of compositions, and ultimately, it is equivalent to testing different values of the matrices parameters in many cases. Our techniques have been implemented on top of the Open64/ORC compiler. In addition, we have designed a prototype iterative optimization infrastructure for iterative optimization, based on genetic algorithms. This infrastructure distributes simulations, dynamic profiles, compilations, transformations, while interacting with a machine-learning component or with an expert user. Validation of these concepts and application of the tools is beginning on the SPEC CPU2000 benchmarks; showing the ability of our tools and framework to scale to larger codes is a critical phase in the center for program tuning. Recent research addresses the automatic search of program transformations in a multidimensional space, combining Lagrangian relaxation (e.g., Farkas Lemma), operation research algorithms and iterative optimization.

6.1.2. Iterative compilation and continuous optimizations

Participants: Albert Cohen, Grigori Fursin, Olivier Temam.

Currently we are working on iterative compilation and continuous optimizations. For iterative compilation we attempt to further improve existing compiler infrastructure such as gcc or PathScale to be able to apply a greater variety of program transformations systematically and automatically to considerably improve program performance and/or reduce power consumption for embedded systems. We investigate both static and run-time optimizations and adaptation towards various data inputs. For continuous optimizations we attempt to collect all information available during program optimization and its multiple executions with various datasets. We further intend to use machine-learning techniques to quickly and automatically optimize new programs or adapt towards new datasets by exploiting all previously gathered knowledge.

This year we published 3 papers on the above topics.

The paper [33], ranked first at the International Conference on High Performance Embedded Architectures and Compilers (HIPEAC 2005) shows a method to make iterative optimization practical and usable by speeding up the evaluation of a large range of optimizations. Instead of using a full run to evaluate a single program optimization, we take advantage of periods of stable performance, called phases. For that purpose, we propose a low-overhead phase detection scheme geared toward fast optimization space pruning, using code instrumentation and versioning implemented in a production compiler .We demonstrate that it is possible to search for complex optimizations at run-time without resorting to sophisticated dynamic compilation frameworks. In addition to that, our approach also enables to quickly design self-tuned applications.

The two other papers [36] and [32] (in collaboration with Edinburgh University) explore the ways to search for best transformations in large optimizations spaces. The first paper uses Pugh's Unified Transformation Framework to exploit the performance improvement potential of complex transformation compositions and presents a heuristic search algorithm capable of efficiently locating good program optimizations within such a space. The second paper empirically evaluates source-level transformations and the probabilistic feedback-driven search for "good" transformation sequences within a large optimization space.

6.1.3. Low-level optimization

Participants: Patrick Carribault, Albert Cohen.

This work is done in collaboration with William Jalby from University of Versailles-Saint-Quentin. To achieve the best performance on single processors, optimizations need to target most components of the architecture simultaneously, focusing on the memory hierarchy (including registers), branch prediction, instruction-level parallelism and vector (SIMD) parallelism. Typical examples of good candidates for aggressive optimization technologies include regular and numerical computations from scientific, signal processing or multimedia applications.

More irregular programs can also be data and compute intensive, but less architecture-aware optimizations have been proposed for such programs. Still, speculative and very complex transformations are available for such codes in the context of massively parallel computers. We investigated the applicability and extension/adaptation of some of these techniques for the optimization on uniprocessors, and our results were extremely promising in the case of two approximate string-matching codes (for computational biology). Hybrid static-dynamic optimizations for such programs are also being considered, driving the selection of optimization parameters at run-time through the fine-grain tracking of the behaviour of the application (performance counters).

Finally, we studied even more irregular codes: decision trees in control-intensive emulators, text processors or memory management functions. We showed that, surprisingly, high quality performance predictions could be achieved at compile time, helping the compiler to take the right code generation decisions. This study also led to the design of a new program transformation, called Deep Jam, generalizing the unroll-and-jam optimization to nested irregular loops with conditionals and early exits [26].

6.1.4. Advanced analysis and optimization for the end-user

Participants: Albert Cohen, Sebastian Pop.

This work is done in collaboration with Georges Silber, Pierre Jouvelot and François Irigoin from École Nationale Supérieure des Mines de Paris.

We designed an induction variable analyzer suitable for the analysis of typed, low-level, three address representations in SSA form. At the heart of our analyzer is a new algorithm recognizing scalar evolutions. We define a representation called trees of recurrences that is able to capture different levels of abstractions: from the finer level that is a subset of the SSA representation restricted to arithmetic operations on scalar variables, to the coarser levels such as the evolution envelopes that abstract sets of possible evolutions in loops. Unlike previous work, our algorithm tracks induction variables without prior classification of a few evolution patterns: different levels of abstraction can be obtained on demand [39]. The low complexity of the algorithm fits the constraints of a production compiler, and roots the mainline dependence analysis framework in the Gnu Compiler Collection (GCC), as illustrated by the evaluation of our implementation on standard benchmark programs [49].

6.2. Revisiting the processor architecture/programming approach

6.2.1. Generative programming

Participants: Albert Cohen, Sébastien Donadio.

This work is done in collaboration with Denis Barthou and William Jalby from the University of Versailles St-Quentin.

The quality of compiler-optimized code for high-performance applications lags way behind what optimization and domain experts can achieve by hand. We believe that program generation and metaprogramming approaches are effective means to bring domain-specific and aggressive optimization knowledge together in a productive programming environment. However, the language and framework for that does not exist yet and we investigate possible directions.

- Considering loop nest optimizations, we study how generative approaches can help the design and optimization of supercomputing applications. We obtained numerous results, using MetaOCaml for the design of a generative tool-box to design portable optimized code [42], and in the design of a domain-specific language to describe, drive and search for program transformations for high-performance computing [31]. We also identify some limitations of the MetaOCaml system. We finally advocate for an offshoring approach (direct translation to C) to bring high-level and safe metaprogramming to imperative languages. This work is done in collaboration with colleagues from the University of Illinois.

- Parallel processing is an important case for such approaches, due to the low productivity of implementing and debugging efficient parallel code. In collaboration with our Colleagues from the University of Passau, we try to design a domain-specific adaptive library to solve branch-&-bound problems in parallel. Such a library would not be a new thing, but we aim at raising simultaneously the level of abstraction for the programmer and the quality/efficiency of the generated code. In particular, we rely on aggressive specialization (partial evaluation) of memory management and communication primitives. Our early results are very positive on marshaling (serialization) and partial evaluation [30].

6.2.2. MGS interpreter

Participant: Julien Cohen.

As a continuation of our work in the LaMI laboratory of the University of Evry, we have extended our work in the MGS (Modèle Général de Simulation [50]) interpreter in two directions:

- We have formalized the use of an higher-order abstract syntax in the implementation of the MGS interpreter. Basic combinators are used to transform functions of the interpreted language into functions of the host language. This provides an alternate way to quickly implement interpreters [27].
- Topological collections are a means to view many data structures in a single framework. They can be handled into a programming language with functions defined by pattern matching called transformations. Collections and transformations are very useful in biological simulations where the collections used are often heterogeneous. This means that the collections contain values of different types. We present here a type system for heterogeneous topological collections and transformations, which uses a set based subtype relation and mixes static type inference and dynamic type tests [21].

6.2.3. The Blob computing paradigm

Participants: Julien Cohen, Frédéric Gruau, Yves Lhuillier, Noureddine Oulagha, Olivier Temam.

(joint work with Philippe Reitz, Université de Montpellier)

Current processor and multiprocessor architectures are almost all based on the Von Neumann paradigm. Based on this paradigm, one can build a general-purpose computer using very few transistors, e.g., 2250 transistors in the first Intel 4004 microprocessor. In other terms, the notion that on-chip space is a scarce resource is at the root of this paradigm which trades on-chip space for program execution time. Today, technology considerably relaxed this space constraint. Still, few research works question this paradigm as the most adequate basis for high-performance computers, even though the paradigm was *not* initially designed to scale with technology and space.

Blob is a different computing model, defining both an architecture and a language, that is intrinsically designed to exploit *space*. This year we have investigated the movement of blobs with membranes through the use of the MGS programming language designed at the LaMi laboratory [47].

Blob computing is also part of the PhD work of Yves Lhuillier [16]

6.2.4. Symbiotic processing

Participants: Yves Lhuillier, Pierre Palatin, Olivier Temam.

Because clock frequency may no longer increase as quickly, there is a growing consensus on on-chip concurrent architectures being a major route for performance scalability. However, the fact that scalability will be limited by the compiler ability to automatically extract program parallelism (a task known to be particularly difficult in programs with complex control and data structures) tends to be overlooked. As a result, this scalability alternative may be heading for the same issues as ILP-based architectures: excessively complex hardware speculation, or an excessive burden on the compiler.

In [46], we advocate that research efforts should further focus on striking the right balance between architecture, compiler and *user* effort. Research in parallel/spatial programming paradigms has been addressing this issue and making significant progress which can now benefit micro-architectures in two ways: (1) by letting the user reasonably effortlessly pass information on program parallel properties to the architecture, thereby simplifying the task of the compiler and the architecture, and (2) by making the architecture "aware" of the running program and empowering it with the ability to dynamically allocate resources to the concurrently executing program parts, thereby better exploiting hardware resources, a task normally assigned to the compiler. We illustrate the performance and scalability of this combined programming/architecture approach, called *Symbiotic Processing*, using SMTs and several appropriately written programs corresponding to a set of classic and non-trivial algorithms. Finally, we outline that, much like SIMD, this approach can be progressively adopted through hardware add-ons and C/C++ language extensions.

6.2.5. Synchronous languages and high-performance real-time applications

Participants: Albert Cohen, Marc Duranton, Christine Eisenbeis, Claire Pagetti.

This work is done in collaboration with Florence Plateau and Marc Pouzet from Paris 6 University, then University of Paris-Sud since september 1st, 2005. The design of high-performance stream-processing systems is a fast growing domain, driven by markets such like high-end TV, gaming, 3D animation and medical imaging. It is also a surprisingly demanding task, with respect to the algorithmic and conceptual simplicity of streaming applications. It needs the close cooperation between numerical analysts, parallel programming experts, real-time control experts and computer architects, and incurs a very high level of quality insurance and optimization.

In search for improved productivity, we propose a programming model and language dedicated to high-performance stream processing [28], [43]. This language builds on the synchronous programming model and on domain knowledge — the periodic evolution of streams — to allow correct-by-construction properties to be proven by the compiler. These properties include resource requirements and delays between input and output streams. Automating this task avoids tedious and error-prone engineering, due to the combinatorics of the composition of filters with multiple data rates and formats. Correctness of the implementation is also difficult to assess with traditional (asynchronous, simulation-based) approaches. This language is thus provided with a relaxed notion of synchronous composition, called n-*synchrony*: two processes are n-synchronous if they can communicate in the ordinary (0-)synchronous model with a FIFO buffer of size n.

Technically, we extend a core synchronous data-flow language with a notion of periodic clocks, and design a relaxed clock calculus (a type system for clocks) to allow non strictly synchronous processes to be composed or correlated. This relaxation is associated with two sub-typing rules in the clock calculus. Delay, buffer insertion and control code for these buffers are automatically inferred from the clock types through a systematic transformation into a standard synchronous program. We formally define the semantics of the language and prove the soundness and completeness of its clock calculus and synchronization transformation. Finally, the language is compared with existing formalisms.

6.3. Processors architectures and simulation

6.3.1. MicroLib: A case for the Quantitative Comparison of Micro-Architecture Mechanisms

Participants: Daniel Gracia Pérez, Gilles Mouchard, Olivier Temam.

While most research papers on computer architectures include some performance measurements, these performance numbers tend to be distrusted. Up to the point that, after so many research articles on data cache architectures, for instance, few researchers have a clear view of what are the best data cache mechanisms. To illustrate the usefulness of a fair quantitative comparison, we have picked a target architecture component for which lots of optimizations have been proposed (data caches), and we have implemented most of the performance-oriented hardware data cache optimizations published in top conferences in the past 4 years. Beyond the comparison of data cache ideas, our goals are twofold: (1) to clearly and quantitatively evaluate the impact of methodology shortcomings, such as model precision, benchmark selection, trace selection..., on assessing and comparing research ideas, and to outline how strong is the methodology impact in many cases, (2) to outline that the lack of interoperable simulators and not disclosing simulators at publication time make it difficult if not impossible to fairly assess the benefit of research ideas. This study has been published as part of the PhD work of Daniel Gracia-Perez [17]. It is also part of a broader effort, called *MicroLib/Fraternité* (see section 5.3, an open library of modular simulators aimed at promoting the disclosure and sharing of simulator models.

6.4. A biological approach to computing

The analytical methods traditionally used to predict microprocessor performance (and thus conceive future architectures) are increasingly challenged by the complexity of modern microprocessors, because these methods are based on the idea that knowing and predicting the behavior of every individual component of these computing systems is enough to precisely predict their global large-scale behavior. While this task is already challenging in nowadays microprocessors, it may become daunting in future large-scale multi-core processors. As a result we are turning to novel approaches for designing and operating such large-scale systems.
Understanding global properties of systems with a large number of elements sharing intricate interactions is precisely the ambition of the so-called "complex systems" approaches. Using multidisciplinary tools from applied mathematics, statistical physics and computer sciences, they aim at comprehending and controlling the relations between micro/local behaviors and macro/global levels. They could provide architecture research with suitable methods for the study/development of future computing systems.
More precisely, the challenges faced by the designers of future systems are twofold:

- to control and organize a large set of interacting computing components,
- to find methods for taking advantage of such large-scale networks for computing tasks.

While complex systems can be found in traditional physics, computer or social sciences fields, they are ubiquitous in biology. Recent focus in biology is dedicated to understanding the structure, behavior and operations of large-scale biological systems using a complex systems approach. And many of the properties displayed by such complex biological systems are just what future architectures will have to possess: self-organized and scalable structures, robustness and fault/defect-tolerance.
Among the diverse concerned systems, large-scale biological neural networks share all these properties but also have the capability to take advantage of a large number of slow components to quickly realize complex computing tasks. These complex nets are made of a large number of neurons that are interconnected in an irregular but not random network. Deciphering their structure and the relation to their function is a way toward understanding how they solve complex tasks and demands multidisciplinary approaches, such as those proposed by complex systems. While our research is inspired by a large range of biological systems for addressing the first challenge (structure and control), we specifically focus on large-scale biological neural networks for addressing the second task (usage and programming).
The overall research effort is obviously very tied to the understanding of biological systems and how they realize information processing tasks. Because some of these properties are not yet well enough understood, our research effort constantly oscillates between better understanding/modeling biological systems and applying properties of biological systems to computing systems.
How is this research different from grid computing research ? And how is it different from artificial neural network (ANN) research ?
Grid computing research also targets large-scale parallel systems, but each core executes an independent task, which does not interact with other tasks during its execution. Our challenge is to speed up a single task by distributing it on a large number of cores. Grid computers are loosely connected networks of computers, while our systems belong to a single chip with communication latencies of the order of a few cycles, allowing many different organizations.
Artificial neural networks are also inspired from biological neural networks, but they have branched out in the 1970s or so from biology, and have mostly relied on simplistic structures (feedforward layers, regular, fully connected) and learning strategies (supervised) which are very different from biological neural networks. While ANNs have interesting properties, they are missing features of biological neural networks, such as

the ability to learn unsupervised, to abstract complex notions, to grow networks capable of performing a large range of tasks, to learn and perform at the same time, and so on. We wish to better understand how such capabilities are implemented in biological neural networks in order to derive novel computing and programming approaches. Effectively using biological neural networks as a support for computations may even be a remote but tractable goal.

6.4.1. *Self-developing biological neural networks: A first step towards application to computing systems*

Participants: Hugues Berry, Olivier Temam.

Carbon nanotubes are often seen as the only alternative technology to silicon transistors. While they are the most likely short-term alternative, other longer-term alternatives should be studied as well, even if their properties are less familiar to chip designers. While contemplating biological neurons as an alternative component may seem preposterous at first sight, significant recent progress in CMOS-neuron interface suggests this direction may not be unrealistic; moreover, biological neurons are known to self-assemble into very large networks capable of complex information processing tasks, something that has yet to be achieved with other emerging technologies.

The first step to designing computing systems on top of biological neurons is to build an abstract model of self-assembled biological neural networks, much like computer architects manipulate abstract models of transistors and circuits. We have proposed a first model of the *structure* of biological neural networks[19], [25]. We provide empirical evidence that this model matches the biological neural networks found in living organisms, and exhibits the *small-world* graph structure properties commonly found in many large and self-organized systems, including biological neural networks.

More importantly, we extract the simple local rules and characteristics governing the growth of such networks, enabling the development of potentially large but realistic biological neural networks, as would be needed for complex information processing/computing tasks. Based on this model, future work will be targeted to understanding the evolution and learning properties of such networks, and how they can be used to build computing systems.

6.4.2. *Structure and dynamics of random recurrent neural networks*

Participants: Hugues Berry, Mathias Quoy.

Recurrent neural networks (in which all neurons are interconnected) include all possible backward connections, which endows them with a rich variety of dynamical behaviors. Many real neuron networks show high proportions of such backward connections. Recurrent neural networks appear thus more interesting to the understanding of computation in large neural networks than the classical feed-forward structures used in most artificial neural networks. Unlike Hopfield-like networks, Random Recurrent Neural Networks (RRNNs), where the couplings are random, exhibit complex dynamics (limit cycles, chaos) and transitions between them. It is possible to store information in these networks through Hebbian learning. For instance, they can be used as models for learning in the olfactory bulb, where the dimension of the dynamics attractor reduces on a more simple attractor (limit cycle) when a known odor is recognized. These experimental findings can be replicated using RRNNs with a classical Hebbian learning rule.

Many studies have focused on the evolution of neuron dynamics in RRNNs, but the emergence of the statistical weight network structure during learning remains poorly understood. We propose here to use both a dynamical system and a graph theory approach in order to understand the dynamical and structural changes occurring through learning in(RRNNs) [24].

Several recent studies have focused on the topological structure of large networks using graph theory approaches. They have proven successful in understanding the global properties of several complex systems originating from highly disparate fields, from the biological to social and technological domain. The most common statistical structures are the so-called *small-world* and *scale-free* networks. Small-world properties characterize networks with both small average shortest path and a large degree of clustering, while scale-free networks are defined by a connectivity probability distribution that decreases as a power law.

We show that beginning with a randomly connected neural network, and running a Hebbian learning rule produces the following behaviors:

- the dynamics reduces from chaos to a limit cycle and finally a fixed point, a phenomenon reminiscent of learning experiments in the olfactory bulb
- the set of “active” neurons is reinforced, and this set is specific for the input fed to the network
- the resulting graph structure is looking like a small-world one

These results show that, in parallel to changing the network dynamics, Hebbian learning in RRNN provokes an auto-organization of the weights on the network, where the local and global properties of the topology are optimized. Future work will study in greater details this auto-organization and will attempt to decipher the emergence of the weight structure, in relation with neuron dynamics and their modifications during learning.

6.4.3. Complex systems analysis of computer performance traces

Participants: Hugues Berry, Daniel Gracia Pérez, Olivier Temam.

Today computer processors rely on amazingly high numbers of transistors: the widespread Intel® Pentium® 4 contains 42 million transistors but the more recent Itanium® 2 possesses 410 million of them. Furthermore, a constant of this evolution is that processor speed (especially, its clock rate) by far outperforms memory operations. Hence, most recent advances in the field have mainly aimed at hiding memory latencies using engineering solutions (parallel executions, pipelining, cache memory systems). But this necessarily came with further increases of the processor complexity. As a consequence, traces recording instantaneous performance during the execution of certain programs can be highly variable and difficult to predict.
In this context, we study the time-evolution of the performance during execution of several prototypical programs on prototypical modern microprocessors [41]. We record several metrics characterizing execution performance (number of instruction executed at each processor cycle) and memory operations (cache misses). Treating these traces as time-varying signals, we analyze them using current techniques from complex systems sciences (nonlinear time series analysis in particular). These techniques have been used, for example, to analyze and quantify signals from complex physiological systems, such as heartbeat time series (electrocardiograms) or brain waves (electroencephalograms).
Besides regular periodic behaviors, we evidence highly variable performance evolutions for several programs. More interestingly, we show that, although the high variability displayed by several programs can be attributed to stochastic-like sources, the evolution of performance during the execution of several others displays clear evidences of deterministic chaos, with sensitivities to initial conditions that are comparable to textbook chaotic systems.
These results seem important because they imply that performance predictability based on short sampled sequences might be hardly possible for several programs and because in a more general perspective, it reveals the high intricacy of the processes determining instantaneous microprocessor performances. Our results may also have some practical importance in the field of performance modeling. Hence, for the programs that display chaotic performance traces, our results suggest that an efficient strategy for predicting the actual average value of the metric under consideration on the ground of a sample of its real trace would be to base the estimation on several samples extracted from the real trace, even in a random way. Actually this method is used by one of the most powerful tool developed for performance prediction. Finally, this work shows the necessity to adapt the performance simulation/sampling technique as a function of the program under consideration, which has recently been pointed out by others. Our results might account for a rationale of this necessity.

6.4.4. Protein surface-aggregation: an exemple of biological self-organized emergent behavior

Participants: Delphine Pellenc, Olivier Gallet, Hugues Berry.

Protein aggregation on two-dimensional surfaces constitutes a challenging complex system in which local protein-protein interactions give rise to specific large-scale behaviors over the aggregation surface (aggregation patterns). Our interest here is that this constitutes a typical biological example for the formation of *large-scale*

spatial structures on the basis of simple local rules. This self-organized phenomenon can be studied using agent-based simulations derived from the classical diffusion-aggregation models of statistical physics.
Two-dimensional rigid spheres aggregation models may be applied to protein aggregation when no large conformational change (unfolding) is involved. Yet, following adsorption, several proteins undergo an unfolding transition that may be involved in aggregative structures. Our focus here is thus how such conformational changes might influence the self-organized large-scale spatial structures, using a diffusion-aggregation model. We propose a model including diffusion, aggregation, and unfolding of proteins that are randomly adsorbed onto a surface [22], [37], [38]. Our model allows simulating the case where protein-protein interaction favors unfolding and the case where this interaction prevents it. We study the effect of a simple disk-to-rod unidirectional unfolding and investigate the morphology of the resulting clusters in the diffusion- and reaction-limited regimes. A rich variety of structures is produced, with fractal dimension differing from that in universal diffusive aggregation models. Increasing unfolding probability shifts the system from the neighbor-induced to the neighbor-prevented unfolding regime. The intermediate structures that arise from the model could be helpful in understanding the assembly of the different large-scale structures that are observed experimentally [48].

7. Contracts and Grants with Industry

7.1. Philips Research

Participants: Albert Cohen, Marc Duranton, Christine Eisenbeis.

Following the SANDRA project [2], Marc Duranton from Philips Research (Eindhoven) devotes 10% of his time (officially) to pursue collaborative work with us. He visited us on a regular basis.

A Marie-Curie Transfer-of-Knowledge Industry-Academia Partnership was granted (2006 to 2008) between Philips Research, our group at INRIA Futurs, and the compilation and architecture group at UPC (Barcelona, Spain).

Together with Zbigniew Chamski and colleagues from Philips, Marc Duranton helped us define an INRIA ARC proposal and an ANR RNTL proposal, together with Marc Pouzet from Paris-Sud 11 University on long term research issues. We also prepare two European project proposals (one on the topic of this collaboration and another as part of a wider Integrated Project proposal).

7.2. The Cop project

Participants: Cédric Bastoul, Albert Cohen, Sylvain Girbal, Marc Gonzalez-Sigler, Saurabh Sharma, Olivier Temam.

In 2004, we had the first year of the exploratory RNTL project (long-term academic-industrial research project, funding from the ministry of research) called "Centre d'Optimisation de Programmes" (COP) or "Center for Program Tuning" (CPT). The partners are University of Paris-Sud, IRIT (Toulouse), CEA Saclay, STMicroelectronics Grenoble and HP France. The goal of the project is to set up a center for program tuning. The center will target general-purpose and embedded processors. Techniques for rapidly optimizing programs are being developed, based on automatic or manual iterative optimization techniques. In 2005 emphasis has been put on machine learning techniques for performance optimization.

8. Other Grants and Activities

8.1. National Initiatives

ASTICO (Apprentissage dans les SysTèmes bIologiques COmplexes/ Learning in complex biological systems) is a 3-year project funded by by the Agence Nationale de la Recherche (ANR). Participants are

Hugues Berry (supervisor), B. Cessac (Institut Non Linéaire de Nice, UMR 6618 CNRS / Université Nice-Sophia Antipolis), B, Delors (ANIM, UMR 742 Inserm / Université Pierre et Marie Curie, Paris), M. Quoy (ETIS, UMR 8051 CNRS / Université de Cergy-Pontoise / ENSEA) and O. Temam.

Alchemy is still involved in the Nanosys (http://nanosys.ief.u-psud.fr) project - this is an ACI (Action Concertée Incitative) on Nanosciences. Nanosys addresses integration of molecular nano-components.

Julien Cohen has done his PhD at the LaMI lab (Évry) and still collaborates with Jean-Louis Giavitto and Olivier Michel from this lab, and with Pierre-Étienne Moreau (Loria, Nancy). He also collaborates with Jean-Pierre Banâtre and Yann Radenac (Irisa).

Sebastian Pop is an external PhD student from École Nationale Supérieure des Mines de Paris, coadvised by Albert Cohen.

Benjamin Dauvergne is a PhD student of the Tropics project-team at Inria Sophia Antipolis on automatic differentiation of programs. He works with Alchemy on program transformations.

Sébastien Donadio is an external PhD student from University of Versailles-Saint-Quentin, since October 1st, 2004, coadvised by Albert Cohen.

Claire Pagetti is still collaborating with Michaël Adélaïde, post doctoral fellow, Sicherheitskritische Eingebettete Systeme, Oldenburg, Germany, Franck Cassez, Researcher CNRS, IRCCyN, Nantes, France, Olivier Roux, Professor, IRCCyN, Nantes, France, Aymeric Vincent, Maître de Conférence, Labri, Bordeaux, France.

Georges Silber, assistant professor at École Nationale Supérieure des Mines de Paris, advises the thesis of Sebastian Pop in collaboration with Albert Cohen.

William Jalby, professor at University of Versailles-Saint-Quentin, advises the thesis of Patrick Carribault in collaboration with Albert Cohen.

Denis Barthou, assistant professor at University of Versailles-Saint-Quentin, advises the thesis of Sébastien Donadio in collaboration with Albert Cohen.

Sid-Ahmed-Ali Touati, assistant professor at University of Versailles-Saint-Quentin, collaborates with Christine Eisenbeis on scheduling and resource allocation problems. He also participate to the organization of the Alchemy seminar.

ALCHEMY organizes a joint seminar with CRI (Centre de Recherches en Informatique, Ecole des Mines de Paris), LRI (Laboratoire de Recherches en Informatique, University of Paris-Sud) and PriSM (University of Versailles-Saint-Quentin). Talks of 2005 are given below.

- january 18th, 2005, *Global compilation scheme for trade-off between code size and performance*, Karine Heydemann, Irisa, Caps project-team.
- january 21st, 2005, *Ordonnancement Modulaire et Hiérarchique*, Paul Feautrier, Ens Lyon, Compsys project-team.
- january 25th, 2005, *Intégration des collections topologiques et des transformations dans un langage de programmation fonctionnel*, Julien Cohen, LRI and Inria Futurs, équipe Alchemy.
- february 8th, 2005, *Polyèdres périodiques pour la compilation de programmes*, Benoît Meister, Université Louis Pasteur, Strasbourg.
- february 22nd, 2005, *Synchronisation d'horloges périodiques dans les réseaux de Kahn*, Claire Pagetti, équipe Alchemy, Inria FutURs.
- april 12th, 2005, *Liberating Threads from Sequential Programs*, David August, Princeton University.
- april 15th, 2005, *Microgrids - scaling ILP to the end of CMOS* , Chris Jesshope, University of Amsterdam.
- june 20th, 2005, *Computation efficiency: the final frontier?!* Krisztian Flautner, Director of Advanced Research at ARM Ltd, Cambridge, UK.
- june 28th, 2005, *Library Generators and Program Optimization*, David Padua, Department of Computer Science, University of Illinois at Urbana-Champaign.

- september 6th, 2005, *Dealing with memory hierarchy when generating high performance libraries*, François Bodin, Université de Rennes, Irisa, Caps project-team.
- december 2nd, 2005, *Multi Processor System-on-Chip research at IMEC: combining disciplines*, Theodore Marescaux, IMEC, Belgium.
- december 6th, 2005, *Scalable scientific visualization : a pixel plumber's perspective*, Mark Shand, ENS/HP Labs.

8.2. European initiatives

8.2.1. Scala

SCALA is an european integrated project concerned with long term research in advanced computer architecture. It focuses on a systematic scalable approach to systems design ranging from small energy critical embedded systems right up to large scale networked data servers. It comes at a stage where, for the first time, we are unable to increase clock frequency at the same rate as we increase transistors on a chip. Future performance growth of computers from technology miniaturisation is expected to flatten out and we will no longer be able to produce systems with ever increasing performance using existing approaches. As current methods of designing computer systems will no longer be feasible in 10 -15 years time, what is needed is a new innovative approach to architecture design that scales both with advances in underlying technology and with future application domains. This is achieved by fundamental and integrated research in scalable architecture, scalable systems software, interconnection networks and programming models each of which is necessary component in architectures 10+ years from now. If successful, this will allow Europe to capitalise on its dominance in embedded systems and inter-connection networks and gain market share as consumer electronics and general purpose computing continue to converge. The objectives of the proposal are following:

- Develop a scalable integrated architecture applicable to a wide range of applications
- Solve the design crisis by reducing architecture complexity and automating design space exploration
- Develop innovative approaches to compiler construction that can adapt to the architecture evolution.
- Propose new system wide approaches to reduce power consumption and integrate power and performance in architectural development.
- Investigate and develop new programming models and runtime systems to provide scalable exploitation of future architecture.
- Integrate research by bringing processor and interconnection designers, compiler, language and runtime experts together to develop long-term sustainable approaches to advanced computer architecture in Europe. This four year project brings together the best researchers in advanced computer architecture to work on disruptive scalable technology for the 10-15 year time frame. It has a cutting-edge research agenda and Europe's most strategically significant industrials partners to exploit this technology and increase Europe's market share in the changing computing landscape of the future.

8.2.2. HiPEAC

HiPEAC addresses the design and implementation of high-performance commodity computing devices in the 10+ year horizon, covering both the processor design, the optimising compiler infrastructure, and the evaluation of upcoming applications made possible by the increased computing power of future devices. The embedded market evolves rapidly, expanding the capabilities of each new device, and making the previous ones obsolete as technology advances. But performance does not simply increase with technology advances, it is mandatory to find a way to translate technology into performance, and such is the role of the computer architect. Europe holds a strong position in the embedded market, but is ill equipped to compete in new domains that require increasing amounts of computing power. Experience shows that current high-performance processors will become tomorrow's embedded processors, and there are several European institutions among the world experts on these high-performance architectures. The convergence of the high-performance and embedded industry provides a unique opportunity for advancement.

High performance devices require high-performance architectures, but also optimising compilers that automatically generates code that exploits the new architectural features. Increasingly complex architectures require increasingly complex compilers, and so, designing both in conjunction becomes crucial. Also, there is no architecture/compiler pair that scales with technology, making design and development of new architectures very costly. It becomes critical to design architectures that scale well with technology, amortizing the production effort across a wider time period. Europe has a very strong, but largely dispersed research community with expertise on both the embedded and high-performance domains. However, cooperation between the software and hardware communities is badly needed, and there is little cooperation between industry and academia.

The objectives of HiPEAC are to ensure the visibility of European institutions in the high performance embedded marked, and to promote the integration of research efforts in a common direction. Visibility will be achieved through dissemination of our work under a common HiPEAC label that will raise the awareness of our coordinated research effort. Integration will be achieved through a set of coordinated actions targeted at building a strong community of researchers, and the adherence to a commonly agreed research roadmap that will be strongly influenced by European industry and leading worldwide research institutions. HiPEAC will also provide the means for easy collaboration among members, and rapid dissemination of knowledge among the community, as well as strengthening the relationships between academia and European industry.

HiPEAC brings together the leading European experts in computer architecture, coordinating -for the first time- their research effort. HiPEAC will build up European strength by spreading knowledge and expertise to engineers and students, and by transferring this expertise to industry, with the goal of making Europe the worldwide leader in high-performance embedded processor architectures.

The end target is to create a virtual center of excellence in high-performance compilers and architectures for embedded processors. This center will gather the world's largest critical mass of researchers, generate world-leading results in embedded architectures, and offer the best discussion forums (conferences and journals) on our topics of influence, becoming a focal point in the fields of computer architecture and optimizing compilers at the maximum level.

INRIA is one of the steering committee members of the network, and is assisting UPC in the coordination.
In this context Olivier Temam works

- with Mike O'Boyle of Edinburgh University on iterative/adaptive optimization;
- with Per Stenstrom (Chalmers University, Sweden) on structural simulation - this is a joint work with David August from Princeton University;
- with Bruno Jego and Thierry Strudel (ST MicroElectronics), on parallel programming.

8.2.3. Universitat Polytècnica de Catalunya

A Marie-Curie Transfer-of-Knowledge Industry-Academia Partnership was granted (2006 to 2008) between Philips Research, our group at INRIA Futurs, and the compilation and architecture group at UPC (Barcelona, Spain).

8.2.4. Other collaborations

Albert Cohen coordinates a *Procope* colloboration, sponsored by the French ministry of foreign affairs and German DAAD, with Christoph Herrmann (associate researcher), Christian Lengauer (full professor) and Martin Griebl (associate professor) from the University of Passau. It includes partners from the University of Versailles St-Quentin and from ENS-Lyon (CompSys project-team). The topic of the collaboration is metaprogramming and domain-specific optimization for high-performance computing. Two joint papers were published in a journal and workshop [42], [30], and two visits to our team where organized in September (Christoph Herrmann) and October (Martin Griebl and two Master Students).

In the context of our collaboration with Philips (see 7.1), Marc Duranton (Principal scientist at Philips Research, Eindhoven) visited Inria on regular basis. He works with Claire Pagetti, Albert Cohen and Christine Eisenbeis on synchronous extensions for high-performance video processing. He also collaborates with Olivier Temam and Nathalie Drach on future applications of the virtual hardware compiler approach.

Olivier Temam works on Architecture specialization with Sami Yehia fron ARM and Yanos Sazeides from the University of Cyprus.

8.3. International initiatives

8.3.1. IBM Watson

Sebastian Pop (external PhD student coadvised by Albert Cohen, and former DESS intern) was hired as a summer intern (June to September) to work with David Edelsohn (member of the GCC steering comitee) and Kenneth Zadeck (co-inventor of the SSA form and well known compiler algorithms). He has just been nominated for an IBM PhD fellowship and will spend another summer in Watson next year.

8.3.2. University of Princeton

We collaborate with the Liberty Research Group at the University of Princeton on simulators models. The Liberty Research Group has proposed an environment for the construction of modular simulators similar to the one proposed by our group using SystemC in MicroLib. The collaboration involves the creation of a system that can executes modules written for any of the two environments, and in a more global scope the definition of an unified modular simulation framework unifying both environments and new ideas. In the context of this collaboration David August, Neil Vachharajani, Jonathan Chang, visited the Alchemy team in March (1 week) and in July (2 weeks).

8.3.3. University of Illinois at Urbana-Champaign

Thanks to the renewal of a CNRS-UIUC collaboration contract, coordinated by Paul Feautrier (CompSys project team), we pursued active research with David Padua and his team in Urbana-Champaign, focusing on machine-learning compilation and program generation. We also initiated periodic phone meetings with his colleagues Vikram Adve (on the LLVM low-level compilation infrastructure) and Marc Snir (department head, empirical search for program parallelization and optimization).

8.4. Visiting scientists

Theodor Marescaux (IMEC), Belgium Kris Flautner, ARM, UK Chris Jesshope, University of Amsterdam

Lawrence Rauchwerger is an associate professor at Texas A&M University (an expert on run-time, speculative and hybrid static-dynamic parallelization) visiting us as a *chercheur associé* from CNRS, for 3 months, September to December. We are starting a collaboration with his laboratory, which includes Professors Nancy Amato (well known for parallel algorithms and applications) and Bjarne Stroustrup (inventor of C++).

Our joint work addresses the scalability and efficiency of parallel implementations of high-level container abstractions, with both compiler and architecture aspects.

Other visitors: Professors: Professors: Martin Griebl and Christoph Herrmann (see Section 8.2.4). Students: James Brodman (University of Illinois at Urbana-Champaign), Michael Classen and Philip Classen (University of Passau, see Section 8.2.4).

8.4.1. Dissemination

Other visitors are listed in the sections 8.1, 8.2 and 8.3.

9. Dissemination

9.1. Leadership within scientific community

Hugues Berry chaired the session on “Bio-Inspired Systems” of the 8th International Work-Conference on Artificial Neural Networks (IWANN’2005), Barcelona, Spain, 8-10 June 2005.

Hugues Berry is a member of the *Commission de Spécialistes, section CNU 64-68, Université de Cergy-Pontoise*.

Albert Cohen will give a course on practical aspects of advanced compilation research within the GCC compiler, at the second HiPEAC summer school (ACACES), L’Aquila, Italy, July 2006.

Albert Cohen leads the HiPEAC cluster promoting the use of GCC as a platform for compilation research.

Albert Cohen is the Workshop Chair for the IEEE PACT’06 conference in Seatle, September 2006.

Albert Cohen is the Global Chair for topic 4 (compilers for high performance) of the EuroPar 2005 conference, Lisbon, August 2005.

Albert Cohen is a member of the new IFIP workgroup 2.11 on program generation.

Olivier Temam has served or will serve in the following program committees:

- ISCA 2006
- DATE 2006
- ARCS 2006
- HiPEAC 2005
- Micro 2005
- ACM/IEEE International Conference on Parallel Architectures and Compilation Techniques, 2005.
- CGO, ACM/IEEE International Symposium on Code Generation and Optimization, 2005.

Olivier Temam is a steering committee member of the HiPEAC Network of Excellence. He serves on the steering committee of the newly created and upcoming HiPEAC conference and the HiPEAC journal.

9.2. Teaching at university

Pierre Amiranoff is PRAG in the Mathematics department at the University of Nanterre. He gives courses and labs to first and second year students (L1, L2).

Cédric Bastoul gives Java, network and security lectures and labs at the Paris-11 Institute of Technology to second and third year students (L2 and L3).

Hugues Berry gave a course on “Apport de la modélisation mathématique à l’étude de la dynamique des systèmes cellulaires” in the “mastère de sciences biologiques et médicales” of the “UFR des Sciences Pharmaceutiques de Caen” (3 hours, April 14th, 2005).

Hugues Berry gave a course (3 hours) in the 2nd year of the “mastère de recherche en matière organisée et systèmes vivants” at the university of Cergy-Pontoise (november 9th, 2005).

Patrick Carribault gives Java labs to first year students (L3) at École Polytechnique, and computer architecture labs to third year students (L3) at the University of Versailles St-Quentin.

Albert Cohen teaches a Master of Computer Science Course at Paris-Sud 11 University (M2, compilation and optimization for high-performance and embedded systems). He is also teaching associate at École Polytechnique, for first year Java labs and third year computer architecture (L3 and M1).

Julien Cohen teaches in IUP MIAGE 1 of the University of Paris-Sud (“Approche fonctionnelle de la programmation”, 68 hours) and at the IFIPS engineering school (“Advanced programming languages”, 10 hours).

Sébastien Donadio gives computer architecture labs to fourth year students (M1) at ISTY, the engineering school of the University of Versailles St-Quentin.

Grigori Fursin taught a 2 hours lecture in the Master of Computer Science Course at University of Paris-Sud.

Daniel Gracia Pérez teaches at University of Paris-Sud (Advanced Architectures, (Exercises, M1), Logical Circuits and Physical Operators (Exercises, L1), Computer Architecture (Exercises, L3) until june 2004, and M2 labs on Advanced Architectures since September 2004.

Yves Lhuillier teaches Java progamming (L3) and Introduction to Computer Science (L1) at University of Paris-Sud.

Pierre Palatin: 4 hours (L1/L2) per week (IUT Orsay).

David teaches Processors’ Architecture (L3) at University of Paris-Sud.

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

Nicolas Vasilache gives architecture and programming lectures and labs at ESGI (private school for engineers) in Paris, to first and third year students (L1, L3).

9.3. Workshops, seminars, invitations

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

- Hugues Berry gave two seminars in the working group on “Atelier Fractal (université de Cergy-Pontoise)”. One on “A propos d’objets biologiques fractals” (February 9th, 2005) and one on “Caractérisation de la performance des microprocesseurs pendant l’execution des programmes: Régularité, chaos et autres dynamiques” (November 9th, 2005).
- Hugues Berry attended the International Work-conference on Artificial Neural Networks (IWANN) in Barcelona, Spain, June 8-10, 2005, and gave a conference on “Characterizing Self-developing Biological Neural Networks: A First Step Towards their Application to Computing Systems”.
- Hugues Berry attended the IMACS World Congress 2005 for Scientific Computation, Applied Mathematics and Simulation (Paris, july 11-15, 2005) and gave a talk on “How do surface- and neighbour-induced conformational changes affect the morphological properties of diffusion-aggregation driven surface-assemblies?”.

- Hugues Berry participated to the European Conference on Mathematical and Theoretical Biology (ECMTB) (Dresden, Germany, July 18-22, 2005) and gave a talk on “Two-dimensional protein aggregation: Effect of surface- and neighbour-induced conformational changes”.
- Hugues Berry attended the workshop on Active agents and their environments as dynamical systems (as part of the VIIIth European Conference on Artificial Life (ECAL), University of Kent, Canterbury, UK, September 5-9, 2005) and gave a talk on “Structure and dynamics of random recurrent neural networks”.
- Hugues Berry was invited to give a talk at the Fourth International Congress of Cellular and Molecular Biology (Poitiers, October 7-12, 2005) on “Nonequilibrium Phase transition in auto/paracrine cell signaling”.
- Hugues Berry and Christine Eisenbeis attended the First European Congress on Complex Systems (ECCS’05) (Paris, november 14-18, 2005).
- Albert Cohen participated to the second meeting of IFIP WG 2.11, Houston, Texas, March 2005 (talk about ongoing work on generative programming for loop nest optimizations).
- Albert Cohen presented the Alchemy project-team and ongoing work on iterative optimization at the department of computer science, University of Texas A&M, March 2005.
- Albert Cohen visited the compiler group at the University of Illinois at Urbana-Champain, June 2005.
- Albert Cohen attended the ACM International Conference on Principles and Practice of Parallel Processing (PPoPP’05) and Languages, Compilers and Tools for Embedded Systems, Chicago, Illinois, June 2005.
- Nicolas Vasilache and Albert Cohen participated to the ACM International Conference on Supercomputing (ICS’05), Cambridge, Massachusetts, June 2005 (article and presentation by Albert Cohen about a compositional loop nest transformation framework for iterative optimization).
- Nicolas Vasilache and Christine Eisenbeis attended the MicroGrid workshop in Amsterdam (July 1st and 2nd, 2005) (talk of Nicolas Vasilache on “Characterization of Legal Transformations of Sequences”).
- Albert Cohen, Patrick Carribault and Nicolas Vasilache attended the first HIPEAC summer school on Advanced Computer Architecture and Compilation for Embedded Systems (ACACES), L’Aquila, Italy, July 2005. Gilles Mouchard, Daniel Gracia Pérez and Marek Doniec assisted Olivier Temam in his lecture on Computer Architecture.
- Albert Cohen served as a session chair at the Euro-Par’05 conference, Lisbon, Portugal, August 2005.
- Patrick Carribault and Albert Cohen participated to the IEEE Conference on Parallel Architectures and Compilation Techniques (PACT’04), September 2005 (article and presentation by Patrick Carribault about the Deep Jam transformation for irregular loop nests).
- Albert Cohen participated to the second MetaOCaml Workshop, Tallinn, Estonia (article and presentation on generative programming for compound marshaling).
- Albert Cohen presented the different research projects involving the Open64 compiler to the Chinese Institute for Computing Technology, and was invited as a panelist of the second IFIP conference on Networks and Parallel Computing (NPC’05), Beijing, China, December 2005.
- Albert Cohen presented a talk on continuous optimization at the Parallel Processing Institute of Fudan University, Shanghai, China, December 2005.

- Albert Cohen participated to a meeting of the HIPEAC cluster on GCC as a platform for compilation research, and to the IBM Research Workshop on Compilation and Architecture (talk about program transformation and generation frameworks for adaptive compilation), Haifa, Israel, December 2005.
- Julien Cohen gave a number of seminars on his work: at Irisa (januray, 2005), at Inria Futurs (february, 2005), at Lip6 (université of Paris 6) (april, 2005), at LACL (université of Paris 12) (april, 2005).
- Christine Eisenbeis participated to the meeting of the IFIP Working Group 10.3 in Paris (February 18th, 2005) and gave a talk on “Computation intensive videoprocessing - requirements, architecture, programming language”.
- Christine Eisenbeis participated to the Dagstuhl meeting on “Scheduling for Parallel Architectures: Theory, Applications, Challenges” (March 6-11, 2005) and gave a talk on “Synchronization of periodic streams”.
- Grigori Fursin attended the HIPEAC international conference on High Performance Embedded Architectures and Compilers (Barcelona, november 2005) and gave a talk on “A Practical Method for Quickly Evaluating Program Optimizations”. He presented this work at the HIPEAC European Commission review meeting in Barcelona (November 2005) and in the Institute for Computing Systems Architecture, Edinburgh University, December 2005 (invited talk).
- Frédéric Gruau was invited at the Chalmers University of Technology (Göteborg, Sweden) and gave a talk on “Blob computing: towards a model combining scalability and programmability”.
- Pierre Palatin attended the ACACES’05 summer school (one week in July) at l’Aquila, Italy; Poster of Pierre Palatin “Spatial processing”.

Olivier Temam was invited to give seminars at the following places :

- From Sequences of Dependent Instructions to Functions: An Approach for Improving Performance without ILP or Speculation, TU Delft, The Netherlands, February 2005.
- MicroLib: A case for the quantitative comparison of micro-architecture mechanisms, Ghent University, Belgium, 2004.
- Agent programming and self-organizing architectures, IBM Thomas Watson reseach center, USA, 2000.
- Spatial-oriented architectures, University of Texas, USA, 2004.
- Iterative optimization environment, STMicroelectronics, Lugano, Switzerland, 2004.

10. Bibliography

Major publications by the team in recent years

[1] M. BARRETEAU, F. BODIN, P. BRINKHAUS, Z. CHAMSKI, H.-P. CHARLES, C. EISENBEIS, J. GURD, J. HOOGERBRUGGE, P. HU, W. JALBY, P. M. KNIJNENBURG, M. O'BOYLE, E. ROHOU, R. SAKELLARIOU, A. SEZNEC, E. A. STÖHR, M. TREFFERS, H. A. WIJSHOFF. *OCEANS: Optimizing Compilers for Embedded ApplicatioNS*, in "Euro-Par'98", Springer-Verlag, LNCS, septembre 1998.

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

[3] A. COHEN. *Program Analysis and Transformation: from the Polytope Model to Formal Languages / Analyse et transformation de programmes : du modèle polyédrique aux langages formels*, Ph. D. Thesis, Université Versailles-Saint-Quentin-en-Yvelines, December 1999.

[4] A. COHEN, S. DONADIO, M.-J. GARZARAN, C. HERRMANN. *In Search of a Program Generator to Implement Generic Transformations for High-performance Computing*, in "MetaOCaml Workshop", Oct 2004.

[5] A. COHEN, O. TEMAM. *Digital LC-2: From bits and gates to a little computer*, in "International Workshop on Computer Architecture Education", ISCA, May 2002.

[6] A. COHEN, O. TEMAM. *Digital LC-2: From Bits and Gates to a Little Computer*, in "Proc. of the 9th Workshop on Computer Architecture Education (WCAE'02), Anchorage, Alaska, USA", May 2002.

[7] J.-F. COLLARD, D. BARTHOU, P. FEAUTRIER. *Fuzzy array dataflow analysis*, in "Proc. of 5th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming, Santa Barbara, CA", July 1995.

[8] 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.

[9] C. EISENBEIS, A. GONZALEZ, J. LLOSA, M. O'BOYLE, O. TEMAM, G. WATTS. *MHAOTEU: Tools for memory hierarchy management*, in "Conference on Applications of Computer Algebra", IMACS, Aug 2000.

[10] F. GRUAU, Y. LHUILLIER, P. REITZ, O. TEMAM. *BLOB computing*, in "CF'04: Proceedings of the first conference on computing frontiers on Computing frontiers", ACM Press, 2004, p. 125–139.

[11] F. GRUAU, P. MALBOS. *The Blob: A Basic Topological Concept for "Hardware-free" Distributed Computation*, in "Unconventional Models of Computation", Lecture Notes in Computer Science, Springer Verlag, Oct 2002, p. 151–163.

[12] D. PARELLO, O. TEMAM, A. COHEN, J.-M. VERDUN. *Toward a Systematic, Pragmatic and Architecture-Aware Program Optimization Process for Complex Processors*, in "Supercomputing", IEEE, Nov 2004.

[13] 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.

[14] S. YEHIA, O. TEMAM. *From Sequences of Dependent Instructions to Functions: a Complexity-Effective Approach for Improving Performance without ILP or Speculation*, in "International Workshop on Complexity-Effective Design", ISCA, Jun 2003.

Doctoral dissertations and Habilitation theses

[15] S. GIRBAL. *Optimizing applications – Composing program transformations: model and tools*, Ph. D. Thesis, Université Paris-Sud, September, 30th 2005.

[16] Y. LHUILLIER. *Architecture et programmation spatiale*, Ph. D. Thesis, Université Paris-Sud, September, 25th 2005.

[17] D. G. PÉREZ. *Towards the Adoption of Structural Simulation*, Ph. D. Thesis, Université Paris-Sud, October, 21st 2005.

Articles in refereed journals and book chapters

[18] C. BASTOUL, P. FEAUTRIER. *Adjusting a program transformation for legality*, in "Parallel processing letters", vol. 15, n^o 1, March 2005, p. 3–17.

[19] H. BERRY, O. TEMAM. *Characterizing Self-Developing Biological Neural Networks: A First Step Towards their Application To Computing Systems*, in "Lecture Notes in Computer Science", vol. 3512, 2005, p. 306–317.

[20] G. CARON-LORMIER, H. BERRY. *Amplification and oscillations in the FAK/Src kinase system during integrin signaling*, in "Journal of Theoretical Biology", vol. 232, 2005, p. 235-238.

[21] J. COHEN. *Typage des collections topologiques hétérogènes et des transformations*, in "Technique et Science Informatique", To appear, 2005.

[22] D. PELLENC, O. GALLET, H. BERRY. *Adsorption-induced conformational changes in protein diffusion-aggregation surface assemblies*, in "Physical Review E", vol. 72, 2005, 051904.

Publications in Conferences and Workshops

[23] H. BERRY. *Nonequilibrium Phase transition in auto/paracrine cell signaling*, in "Fourth International Congress of Cellular and Molecular Biology, Poitiers, France", (Invited Talk), October 2005.

[24] H. BERRY, M. QUOY. *Structure and dynamics of random recurrent neural networks*, in "Active agents and their environments as dynamical systems, University of Kent, Canterbury, Kent (UK)", workshop held during the VIIIth European Conference on Artificial Life (ECAL), September 2005.

[25] H. BERRY, O. TEMAM. *Characterizing Self-Developing Biological Neural Networks: A First Step Towards their Application To Computing Systems*, in "International Work-conference on Artifical Neural Networks,

IWANN, Barcelona, Spain", June 2005.

[26] P. CARRIBAULT, A. COHEN, W. JALBY. *Deep Jam: Conversion of Coarse-Grain Parallelism to Instruction-Level and Vector Parallelism for Irregular Applications*, in "Parallel Architectures and Compilation Techniques (PACT'05), St-Louis, Missouri", IEEE Computer Society Press, September 2005.

[27] J. COHEN. *Interprétation par SK-traduction et syntaxe abstraite d'ordre supérieur*, in "Journées Francophones des Langages Applicatifs (JFLA 2005)", O. MICHEL (editor). , INRIA, 2005, p. 17-34.

[28] 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.

[29] A. COHEN, S. GIRBAL, D. PARELLO, M. SIGLER, O. TEMAM, N. VASILACHE. *Facilitating the Search for Compositions of Program Transformations*, in "ACM Int. Conf. on Supercomputing (ICS'05), Boston, Massachusetts", June 2005.

[30] A. COHEN, C. HERRMANN. *Towards a High-Productivity and High-Performance Marshaling Library for Compound Data*, in "2nd MetaOCaml Workshop (associated with GPCE), Tallinn, Estonia", October 2005.

[31] S. DONADIO, J. BRODMAN, T. ROEDER, K. YOTOV, D. BARTHOU, A. COHEN, M. GARZARAN, D. PADUA, K. PINGALI. *A Language for the Compact Representation of Multiple Program Versions*, in "Workshop on Languages and Compilers for Parallel Computing (LCPC'05), Hawthorne, New York", LNCS, Springer-Verlag, October 2005.

[32] B. FRANKE, M. O'BOYLE, J. THOMSON, G. FURSIN.. *Probabilistic Source-Level Optimisation of Embedded Systems Software.*, in "Proceedings of the Conference on Languages, Compilers, and Tools for Embedded Systems (LCTES'05)", june 2005, p. pages 78-86.

[33] G. FURSIN, A. COHEN, M. O'BOYLE, O. TEMAM. *A Practical Method For Quickly Evaluating Program Optimizations*, in "Proceedings of the 1st International Conference on High Performance Embedded Architectures & Compilers (HiPEAC 2005)", LNCS, n^o 3793, Springer Verlag, November 2005, p. 29-46.

[34] D. GRACIA-PÉREZ, H. BERRY, O. TEMAM. *Budgeted Region Sampling (BeeRS): Wisely Allocating Simulated Instructions for a Better Accuracy/Speed/Applicability Tradeoff*, in "5th IEEE International Symposium on Signal Processing and Information Technology, Athens, Greece", December 2005.

[35] D. GRACIA-PÉREZ, H. BERRY, O. TEMAM. *EDCA: A New Clustering Approach For Sampling*, in "MoBS: Workshop on Modeling, Benchmarking, and Simulation, Madison, Wisconsin, USA", June 2005.

[36] S. LONG, G. FURSIN. *A heuristic search algorithm based on Unified Transformation Framework*, in "Proceedings of the 7th International Workshop on High Performance Scientific and Engineering Computing (HPSEC-05)", june 2005, p. pages 137-144.

[37] D. PELLENC, O. GALLET, H. BERRY. *How do surface- and neighbour-induced conformational changes affect the morphological properties of diffusion-aggregation driven surface-assemblies?*, in "IMACS World Congress 2005 for Scientific Computation, Applied Mathematics and Simulation, Paris, France", July 2005.

[38] D. PELLENC, O. GALLET, H. BERRY. *Two-dimensional protein aggregation: Effect of surface- and neighbour-induced conformational changes*, in "European Conference on Mathematical and Theoretical Biology (ECMTB), Dresden, Germany", July 2005.

[39] 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^o 3793, Springer-Verlag, November 2005.

Miscellaneous

[40] J.-C. BARITAUX. *Specialization of embedded processors: a new path to high performance*, Rapport de stage d'option scientifique de l'École Polytechnique, july 2005.

[41] H. BERRY, D. GRACIA-PÉREZ, O. TEMAM. *Chaos in computer performance*, (in press), 2006.

[42] A. COHEN, S. DONADIO, M.-J. GARZARAN, C. HERRMANN, O. KISELYOV, D. PADUA. *In Search of a Program Generator to Implement Generic Transformations for High-performance Computing*, Accepted with minor revisions, to appear, 2006.

[43] A. COHEN, M. DURANTON, C. EISENBEIS, C. PAGETTI, F. PLATEAU, M. POUZET. *N-Sychronous Kahn Networks33th ACM Symp. on Principles of Programming Languages (PoPL'06), Charleston, South Carolina*, to appear, January 2006.

[44] M. DONIEC. *A component library for the simulation of chip multiprocessors*, Rapport de stage d'option scientifique de l'École Polytechnique, august 2005.

[45] G. EMOND. *Symbiotic programming: a new approach for parallel programming*, Rapport de stage d'option scientifique de l'École Polytechnique, july 2005.

[46] Y. LHUILLIER, P. PALATIN, O. TEMAM. *Symbiotic Processing: Toward a Better Balance Between Architecture, Compiler and User Efforts*, soumis.

[47] N. OULAGHA. *Simulation de mouvement de blob avec membranes*, Rapport de DEA, Université Paris-Sud, september 2005.

[48] D. PELLENC, H. BERRY, O. GALLET. *Adsorption-induced Fibronectin Aggregation and Fibrillogenesis*, (in press), 2006.

[49] S. POP, A. COHEN, P. JOUVELOT, G.-A. SILBER. *The New Framework for Loop Nest Optimization in GCC: from Prototyping to EvaluationProc. of the 12th Workshop on Compilers for Parallel Computers (CPC'06), A Coruña, Spain*, January 2006.

Bibliography in notes

[50] 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, http://citeseer.ist.psu.edu/giavitto01mgs.html.