INRIA

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

2004

Table of contents

1. Team

Head of project team

Olivier Temam [Professor, Paris-11 University, until September 30th, 2004, then Research Director (DR) Inria since October 1st, 2004]

Administrative assistants

Stéphanie Meunier [TR Inria, with Gemo]

Staff members, Inria

Hugues Berry [Research Associate (CR) Inria, on secondment from the Cergy-Pontoise University since September 1st, 2004]
Albert Cohen [Research Associate (CR) Inria]
Christine Eisenbeis [Research Director (DR) Inria]
Claire Pagetti [Postdoctoral Fellow, since September 1st, 2004]

Staff members, Paris-11 University

Julien Cohen [1/2 ATER, since October 1st, 2004]
Nathalie Drach [Assistant Professor until January 31st, 2004]
Frédéric Gruau [Assistant Professor]

Visiting fellows

Grigori Fursin [postdoc fellow, University of Edinburgh, since November 1st, 2004]
Lawrence Rauchwerger [Associate professor on sabbatical from Texas A&M University, CNRS invited professor, September to December 2004]

Junior technical staff

Marc Gonzalez-Sigler [Associate engineer]

Ph. D. students

Pierre Amiranoff [professeur certifié de mathématiques (on secondment), PRAG, Université de Nanterre]
Cédric Bastoul [bourse 1/2 ATER, University of Versailles-Saint-Quentin until August 31st, then 1/2 ATER, Université de Clermont-Ferrand]
Patrick Carribault [Bull fellowship (Cifre), University of Versailles-Saint-Quentin]
Alexandre Coveliers [MENRT scolarship, University of Paris-Sud]
Michaël Dupré [MENRT scolarship, University of Paris-Sud]
Sébastien Favre [Inria scolarship, since April 1st, 2004]
Sylvain Girbal [CEA scolarship, University of Paris-Sud]
Daniel Gracia Pérez [Grant of the University of Paris-Sud]
Yves Lhuillier [MENRT scolarship, University of Paris-Sud]
Gilles Mouchard [1/2 ATER, University of Paris-Sud, until August 31th, 2004]
Pierre Palatin [CNRS BDI scolarship, since October 1st, 2004]
David Parello [HP grant, then 1/2 ATER, University of Paris-Sud, since September 7th, 2004]
Nicolas Vasilache [MENRT scolarship, University of Paris-Sud, since September 2004]
Sami Yehia [1/2 ATER, University of Paris-Sud, until July 2004]

Student interns

Sébastien Donadio [Master of computer science, University of Versailles-Saint-Quentin, March to September 2004]
Pierre Palatin [Master of Computer Science, Universitéde Paris 6, from April 6th till September 2004]
Emmanuel Varoquaux [Summer internship, École Polytechnique, April to August 2004]
Nicolas Vasilache [Master of computer science, University of Paris-Sud, March to September 2004]
Yufei Wang [Master of Computer science, University of Paris-Sud, March to September 2004]

External collaborator

Nathalie Drach [Professor, Paris-6 University, since February 1st, 2004]

2. 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 developping 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

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 [8][26]. 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 [20].

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 [21], 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 [21]. We are also investigating applications of this approach to the specialization/idiomization of embedded processors [9]. 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 [6] 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 [22][24]. 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 [25][27].

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

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 in (X)Emacs. Thousands of installations worldwide, distributed as part of Debian GNU/Linux 3.0.

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) [4]; 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) [11].

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

Participants: Sylvain Girbal, Gilles Mouchard.

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.

5.10. WRaP-IT/URUK

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

This work is one of the cornerstones of our *Center for Program Tuning* (RNTL project, 2004–2006) described in section 7.3. 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 ot top of the Open64/ORC compiler. In addition, we are beginning the design of a robust infrastructure for iterative optimization, based on machine learing 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.

5.11. CLooG

Participant: Cédric Bastoul.

CLooG (*Chunky LOOp Generator*) is a software and a library that generates the loop code for scanning integer points of polyhedra http://www.prism.uvsq.fr/~cedb/bastools/cloog.html.

6. New Results

6.1. Practical approach to program optimizations

6.1.1. Toward a Systematic, Pragmatic and Architecture-Aware Program Optimization Process for Complex Processors

Participants: David Parello, Olivier Temam, Albert Cohen, Jean-Marie Verdun.

(Proceedings of SC2004,Pittsburgh PA, USA, November 2004)

Because processor architectures are increasingly complex, it is increasingly difficult to embed accurate machine models within compilers. As a result, compiler efficiency tends to decrease. Currently, the trend is on top-down approaches: static compilers are progressively augmented with information from the architecture as in profile-based, iterative or dynamic compilation techniques. However, for the moment, fairly elementary architectural information is used. In this article, we adopt a bottom-up approach to the architecture complexity issue: we assume we know everything about the behavior of the program on the architecture. We present a manual but systematic process for optimizing a program on a complex processor architecture using extensive dynamic analysis, and we find that a small set of run-time information is sufficient to drive an efficient process. We have experimentally observed on an Alpha 21264 that this approach can yield significant performance improvement on Spec benchmarks, beyond peak Spec. We are currently using this approach for optimizing customer applications [12][26].

6.1.2. 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*, 2004–2006) 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. 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 ot top of the Open64/ORC compiler. In addition, we are beginning the design of a robust infrastructure for iterative optimization, based on machine learing 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 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.

6.1.3. Low-level optimization

Participants: Patrick Carribault, Albert Cohen, Nicolas Vasilache.

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) [17]. 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). We also obtained excellent results when optimizing a movie encoding benchmark (H263 from ffmpeg), combining iterative optimization, program generation, low-level vectroriation, and various loop transformations. 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 [18].

6.1.4. Virtual Hardware Compiler

Participants: Nathalie Drach, Michaël Dupré, Olivier Temam.

To meet the high demand for powerful embedded processors, VLIW architectures are increasingly complex (e.g., multiple clusters), and moreover, they now run increasingly sophisticated control-intensive applications. As a result, developing architecture-specific compiler optimizations is becoming both increasingly critical and complex, while time-to-market constraints remain very tight.

In this article[21], we present a novel program optimization approach, called the *Virtual Hardware Compiler* (VHC), that can perform as well as static compiler optimizations, but which requires far less compiler development effort, even for complex VLIW architectures and complex target applications. The principle is to augment the target processor *simulator* with superscalar-like features, observe how the target program is dynamically optimized during execution, and deduce an optimized binary for the static VLIW architecture. Developing an architecture-specific optimizer then amounts to modifying the processor simulator which is very fast compared to adapting static compiler optimizations to an architecture. We also show that a VHC-optimized binary trained on a number of data sets performs as well as a statically-optimized binary on other test data sets. The only drawback of the approach is a largely increased compilation time, which is often acceptable for embedded applications and devices. Using the Texas Instruments C62 VLIW processor and the associated compiler, we experimentally show that this approach performs as well as static compiler optimizations for a much lower research and development effort. Using a single-core C60 and a dual-core clustered C62 processors, we also show that the same approach can be used for efficiently retargeting binary programs within a family of processors.

6.1.5. Generative programming

Participants: Albert Cohen, Sébastien Donadio.

The quality of compiler-optimized code for high-performance applications lags way behind what optimization and domain experts can achieve by hand. We beleive 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. Our early results, presented in [19] outlines early results and research directions, using MetaOCaml for the design of a generative tool-box to design portable optimized code. 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.

6.2. Revisiting the processor architecture/progamming approach

6.2.1. The Blob computing paradigm

Participants: Frédéric Gruau, Yves Lhuillier, 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.

In this article, we propose a different computing model, defining both an architecture and a language, that is intrinsically designed to exploit *space*; we then investigate the implementation issues of a computer based on this model, and we provide simulation results for small programs and a simplified architecture as a first proof of concept. Through this model, we also want to outline that revisiting some of the principles of today's computing paradigm has the potential of overcoming major limitations of current architectures [22][23].

6.2.2. AP+SOMT: Agent-Programming Combined with Self-Organized Multi-Threading

Participants: Yves Lhuillier, Olivier Temam.

In order to scale up processors beyond ILP, we explore the exploitation of coarser-grain parallelism. We advocate that a slightly different programming approach, called agent programming (AP), can unveil a large amount of parallelism, potentially simplify the task of optimizing compilers and empower the architecture with the ability to exploit potential parallelism based on available resources. We show that an SMT, augmented with dynamic steering strategies and thread swapping features, is an appropriate solution for such self-organized architectures; self-organized SMT is called SOMT. Using a set of specially written agent-like programs corresponding to classic algorithms, we show that AP+SOMT exhibit better performance, stability and scalability for a large array of data sets, and makes compiler optimizations less necessary. Finally, we outline that the approach can be progressively adopted as a combination of a hardware add-on and C language extensions, much like multimedia support in current superscalar processors [24].

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

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

In the continuity of the SANDRA project (section 7.1) about high performance architectures for video-processing we consider the problem of specifying, controlling, verifying physical time properties directly in programs. Last year we had started a collaboration with Marc Pouzet of the *Université de Paris 6*, aimed at specifying our videoprocessing programs in his LUCID SYNCHRONE synchronous and functional programming language. This year we have developed a new automata-based approach for computing the minimal resource requirements (buffers' sizes) for an implementation of videoprocessing algorithms. Once the sizes are known it is possible to derive a specification of these algorithms in LUCID SYNCHRONE.

The main issue is to be able to handle properties of quasi-synchronism, namely the burstiness property, meaning that arrival of data can be delayed but only within a bounded time window. The automata-based approach manages it by completely flattening the computations; we are now looking for higher level approachs that will avoid this flattening.

6.2.4. Instancewise program analysis

Participants: Pierre Amiranoff, Albert Cohen.

We futher developed a static analysis framework to define and compute static program properties at an infinite number of control points, called instances [10]. Infinite sets of instances are represented by rational languages. Based on this instancewise framework, we can extend the concept of induction variables to recursive programs. For a class of monoid-based data structures, including arrays and trees, induction variables capture the exact memory location accessed at every step of the execution. This compile-time characterization is computed in polynomial time as a rational function (but vectors of induction variables require exponential time). A language called MoGuL (*Monoid Guided Language*) has been proposed to perform this analysis, and an analyser implemented in OCaml has been applied to classical recursive programs. We are also designing an extended and improved dependence test, providing exact results in a subclass of programs with affine conditional expressions and array accesses, based on an encoding of reachability in Minsky machines through integer linear programming [31].

6.3. Processors architectures and simulation

6.3.1. FastSysC: A Fast SystemC Engine

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

SystemC is rapidly gaining wide acceptance as a simulation framework for SoC and embedded processors. While its main assets are modularity and the very fact it is becoming a *de facto* standard, the evolution of the SystemC framework (from version 0.9 to version 2.0.1) suggests the environment is particularly geared toward increasing the framework functionalities rather than improving simulation speed. For cycle-level simulation, speed is a critical factor as simulation can be extremely slow, affecting the extent of design space exploration.

We propose in a fast SystemC engine that, in our experience, can speed up simulations by a factor of 1.93 to 3.56 over SystemC 2.0.1. This SystemC engine is designed for cycle-level simulators and for the moment, it only supports the subset of the SystemC syntax (signals, methods) that is most often used for such simulators. We achieved greater speed (1) by completely rewriting the SystemC engine and improving the implementation software engineering, and (2) by proposing a new scheduling technique, intermediate between SystemC dynamic scheduling technique and existing static scheduling schemes. Unlike SystemC dynamic scheduling, our technique removes many if not all useless process wake-ups, while using a simpler scheduling algorithm than in existing static scheduling techniques [25]

6.3.2. *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 in [28] and [27]. It is part of a broader effort, called *MicroLib*, an open library of modular simulators aimed at promoting the disclosure and sharing of simulator models.

6.3.3. *Alternative Approaches to Improve Performance without ILP*

Participant: Sami Yehia.

Current integration technologies and advances in semiconductor manufacturing open the way to an unprecedented number of transistors on a single processor die. Still, few approaches address applications that have complex data structures or irregular data access patterns. Integer applications particularly suffer from such properties. The main bottlenecks lying in non-numeric applications are the low ILP and irregular data structures that leads to irregular memory accesses having low spatial locality.

In this thesis we propose alternative approaches to exploit on-chip space and reduce the memory wall effect. For codes that have little ILP, we propose a novel approach that collapses dependent instructions to functions that execute independently and in parallel.

Because the collapsing approach is limited by dependent memory accesses, we propose the "load squared", an approach that improves performance of dependent loads that have high miss ratios by adding logic closer to memory. We also investigate a generalization of this concept by presenting a decoupled architecture associated with a language extension that explicitly separates execution from data accesses [13].

6.3.4. *From Sequences of Dependent Instructions to Functions: An Approach for Improving Performance without ILP or Speculation*

Participants: Sami Yehia, Olivier Temam.

In this article, we present an approach for improving the performance of sequences of *dependent* instructions. We observe that many sequences of instructions can be interpreted as *functions*. Unlike sequences of instructions, functions can be translated into very fast but exponentially costly two-level combinational circuits. We present an approach that exploits this principle, speeds up programs thanks to circuit-level parallelism/redundancy, but avoids the exponential costs.

We analyze the potential of this approach, and then we propose an implementation that consists of a superscalar processor with a large specific functional unit associated with specific back-end transformations. The performance of the SpecInt2000 benchmarks and selected programs from the Olden and MiBench benchmark suites improves on average from 2.4% to 12% depending on the latency of the functional units, and up to 39.6%; more precisely, the performance of optimized code sections improves on average from 3.5% to 19%, and up to 49% [30].

6.3.5. Load Squared: Adding Logic Close to Memory to Reduce the Latency of Indirect Loads with High Miss Ratios

Participants: Sami Yehia, Jean-Francois Collard, Olivier Temam.

MEDEA Workshop, held in conjunction with the International Conference of Parallel Architectures and Compilation Techniques (PACT), October 2004

Indirect memory accesses, where a load is fed by another load, are ubiquitous because of rich data structures and sophisticated software conventions, such as the use of linkage tables and position independent code. Unfortunately, they can be costly: if both loads miss, two round trips to memory are required even though the role of the first load is often limited to fetching the address of the second load. To reduce the total latency of such indirect accesses, a new instruction called load squared is introduced. A load squared does two fetches, the first fetch reading the target address of the second. (An offset is optionally added to the result of the first fetch.) The load squared operation is performed by memory-side logic (typically, the memory controller if it isn't located on the main processor chip). In this study, load squared is not an architecturally visible instruction: the micro-architecture transparently decides which loads should be replaced by loads squared. We show that performance is sometimes improved significantly, and never degraded [29].

6.4. Beyond performance

6.4.1. SimpleOS, a simple way to use a computer

Participant: Sébastien Favre.

Within the past 10 years, domestic computer usage has raised in a very important manner. While this was happening, computer operating systems and software became much easier to use, even for a person who knows nothing about computing. That said, there are still some things which aren't really easy to do such as creating a network, installing new hardware, or even, because of the many existing codecs, looking at a movie. SimpleOS is built upon the Linux kernel, and the main idea behind it is to make it really easy to install it on a machine, add new hardware to that machine and use it. The goal of the development we already achieved is to make it work in a zero configuration behavior. For the moment, installation is made without asking the user anything more complicated than the date and time, and hardware upgrades are mostly managed without user intervention within configuration of the OS. We are now developing a transparent way of managing multiple hard disks as if they were one single big storage space to simplify hard disk management by the user.

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. Together with Zbigniew Chamski and colleagues from Philips, helped us define an INRIA ARC proposal with Marc Pouzet from Paris

VI 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. HP

Sami Yehia has spent two months summer internship at HP Palo Alto from July to September 2004.

7.3. 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

8. Other Grants and Activities

8.1. National Initiatives

Véronique Donzeau-Gouge, professor at CNAM is the official supervisor of Pierre Amiranoff.

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).

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

Sébastien Donadio is an external PhD stuent 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 2004 are given below.

- january 12th, *Things to Watch Out for in Benchmarks: Fragility and Redundancy* , Hans Vandierendonck, Université de Ghent, Belgique.
- march 15th, 2004: *"Calculer = se déplacer" : une métaphore topologique pour un langage dédié à la simulation des processus dynamiques à structure dynamique*, Jean-Louis Giavitto, LaMI, Evry.

- april 28th, 200: *Combining Program Recovery, Auto-parallelisation and Locality Analysis for C programs on Multi-processor Embedded Systems*, Mike O'Boyle, Edinburgh University.
- may 25th, 2004, *Optimizing sorting with machine learning* , Xiaoming Li, University of Illinois at Urbana-Champaign.
- june 21st, *Extension temps réel d'AltaRica*, Claire Pagetti, Labri, Bordeaux.
- july 8th, *Mediaprocessing et architecture TriMedia chez Philips*, Ciaran O'Donnell.
- September 16th, *Distributed I-Structure memory system*, Alfredo Cristobal, University of Baja California, Mexico.
- September 23rd, *Frame-based Instruction Processing with the rePLay Framewor*, Sanjay Patel, University of Illinois at Urbana-Champaign.
- october 12th, *Generating Message-Passing Parallel Programs from Abstract Specifications by Partial Evaluation*, Christoph Herrmann, Université de Passau.
- october 19th, *SmartApps: Adaptive Applications for High Productivity / High Performance*, Lawrence Rauchwerger, Texas A&M University.
- november 9th, *The "Blob computing" project*, Frédéric Gruau, ALCHEMY, Inria.
- november 16th, 2004: *Toward a standard representation of C++ in C++*, Gabriel dos Reis, Texas A&M University.
- november 19th, 2004: *Syntol et la synthèse comportementale*, Paul Feautrier, LIP, ÉNS Lyon.
- november 23rd, 200: *On the Urgency Expressiveness*, Claire Pagetti, ALCHEMY, Inria.
- november 30th, 2004, *Architectures reconfigurables et réseaux de communication sur silicium*, Gilles Sassatelli, Université de Montpellier.
- december 3rd, 2004: *On Register Requirement in Sofware Pipelining*, Sid Touati, University of Versailles-Saint-Quentin.

8.2. European initiatives

8.2.1. 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.

8.2.2. 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-Saint-Quentin and from ENS-Lyon (CompSys project-team). The topic of the collaboration is metaprogramming and domain-specific optimization for high-performance computing. One joint paper was published in a workshop [19], and we gave two presentations at the first IFIP 2.11 meeting. The collaboration is renewed for 2005.

In the context of the SANDRA project (voir 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 synhronous extensions for high-performance video processing. He also collaborates with Olivier Temam and Nathalie Drach on future applications of the virtual hardware compiler approach.

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 Daniel Gracia Pérez visited the Liberty Reseach Group laboratory on July during 15 days.

8.3.3. University of Illinois at Urbana-Champaign

Thanks to the renewal of a CNRS-UIUC collaboration contract (until 2006), 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 perdiodic 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). Patrick Carribault and Sébastien Donadio visited UIUC in May (one month), then Xiaoming Li, PhD student of Professor Padua, visited us in June (one month). Albert Cohen visited UIUC in January, then in October (with Professor Denis Barthou from University of Versailles-Saint-Quentin). One joint paper was published in a workshop [19].

8.3.4. Other Collaborations

We collaborate with Jean-Luc Gaudiot (University of Irvine at Los Angeles) and Guang Gao (University of Delaware) on “I-structures” and their use in program optimization. J.-L. Gaudiot visited Inria on July 2nd and December 20th.

8.4. Visiting scientists

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: Christoph Herrmann and Chris Lengauer (see Section 8.2.2). Students: Xiaoming Li (University of Illinois at Urbana-Champaign, see Section 8.3.3), Peter Faber (University of Passau, see Section 8.2.2).

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

9. Dissemination

9.1. Leadership within scientific community

Hugues Berry is a member of the *Commission de Spécialistes, section CNU 64-68, Université de Cergy-Pontoise*. 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 program comitee for the 8th International Workshop on Software and Compilers for Embedded Systems (SCOPES), Eindhoven, September 2004.

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

Christine Eisenbeis served on the program committe of CGO 2004 (International Symposium on Code Generation and Optimization with Special Emphasis on Feedback-Directed and Runtime Optimization, Palo Alto, March 2004) and CC 2004 (International Conference on Compiler Construction, Barcelona, March 2004). She was in the committee of the PhD defence of Benoît Meister, Université de Strasbourg, December 17th, 2004.

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

- ACM/IEEE International Conference on Parallel Architectures and Compilation Techniques, 2005.
- CGO, ACM/IEEE International Symposium on Code Generation and Optimization, 2005.
- ACM/IEEE International Conference on Parallel Architectures and Compilation Techniques, 2004.
- HiPC, ACM/IEEE International Conference on High Performance Computing, 2004.
- ASPLOS, ACM Conference on Architectural Support for Programming Languages and Operating Systems, 2004.
- ISPASS, IEEE International Symposium on Performance Analysis of Systems and Software, 2004.

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

Olivier Temam was in the PhD Committee of Arnaud Darsch (IRISA), Karine Heydemann (IRISA), Mokhoo Mbobi (Supelec), and Ralph Hoffman (EPFL).

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).

C. Bastoul taught as a “moniteur” then as an “ATER” at the University of Versailles-Saint-Quentin(“programming” and “algorithmics”).

Patrick Carribault gives computer architecture labs to third year students (L3) at the University of Versailles-Saint-Quentin.

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

Julien Cohen: 96 hours for 2004-2005 at University of Paris-Sud.

Alexandre Coveliers: 2 hours (L1) per week (IUT Orsay) (64 hours for 2004-2005).

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.

Gilles Mouchard gives labs on Java programming (L3) and “Architecture et Système” (L3).

Pierre Palatin: 2 hours (L1/L2) per week (IUT Orsay).

David Parello gave labs on Advanced Architectures (M1) until june 2004. He now teaches Processors' Architecture (L3) at University of Paris-Sud.

Olivier Temam teachs a computer architecture course at Ecole Polytechnique to 3rd-year students. He also teaches a course on novel processor architectures at University of Paris Sud to Master's students. Finally, until September 2004, as a professor at University of Paris Sud, he taught the regular workload of 192 hours per year to undergraduate and graduate 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:

- Albert Cohen presented the Alchemy project-team and ongoing work on iterative optimization to the compiler seminar at the department of computer science, University of Illinois at Urbana-Champain, January 2004.
- Daniel Gracia Pérez and Gilles Mouchard participated to the DATE'04 conference in Paris La Défense, France (presentation by Gilles Mouchard: "A fast SystemC Engine", February 17th, 2004).
- Albert Cohen participated to the first meeting of IFIP WG 2.11, St-Émilion, France, March 2004 (talk about ongoing work on generative programming for loop nest optimizations).
- Albert Cohen and Olivier Temam visited several teams and researchers at IBM Watson in May, and presented the Alchemy project-team and ongoing work.
- Daniel Gracia Pérez and Olivier Temam participated to the Workshop on Duplicating, Deconstructing, and Debunking 2004 (WDDD 2004), Munich, Germany, 19-20 June 2004 (during ISCA 2004) (presentation by Daniel Gracia Pérez: "MicroLib: A Case for the Quantitative Comparison of Micro-Architecture Mechanisms").
- Patrick Carribault and Albert Cohen participated to the ACM International Conference on Supercomputing (ICS'04), St-Malo, France, June 2004 (article and presentation by Patrick Carribault about register promotion in the polytope model applied to a pattern-matching code [17]).
- Albert Cohen and Emmanuel Varoquaux participated to the Static Analysis Symposium (SAS'04), Verona, Italy, August.
- Cédric Bastoul and Albert Cohen participated to the EuroPar'04 conference, Pisa, Italy, August (two articles and presentations on memory locality optimization in the polytope model by chunking, and on a compositional program transformation framework in the polytope model).
- Frédéric Gruau was invited at the international workshop on Unconventional Programming Paradigms (UPP' 04).
- Gilles Mouchard defended his PhD thesis on "Modélisation de processeurs et de systèmes" on September 17th, 2004.
- David Parello defended his PhD thesis on "Méthodologie d'optimisation de programmes pour architectures de processeurs complexes" on September 17th, 2004.
- Albert Cohen gave a one-day course on polyhedral loop transformation, code generation and automatic parallelization to the compilation and architecture groups at ST-Micro, AST Lugano, September 2004.
- Albert Cohen presented comparison of generative programming and polyhedral transformation techniques to the architecture seminar at the department of computer science, University of Illinois at Urbana-Champain, October 2004.

- Albert Cohen participated to the Generative Programming and Component Engineering Conference (GPCE'04), the ACM OOPSLA'04 Conference, and the first MetaOCaml Workshop, Vancouver (article and presentation on generative programming for loop nest optimization).
- David Parello participated to Supercomputing 2004 (SC2004),Pittsburgh PA, USA, 6-12 November 2004.
- Claire Pagetti has given a seminar at Inria-Orsay on November 23rd.
- Daniel Gracia Pérez, Gilles Mouchard and Olivier Temam participated to the MICRO 37 conference in Portland, Oregon (December 4-8, 2004) (presentation by Daniel Gracia Pérez: "MicroLib: A Case for the Quantitative Comparison of Micro-Architecture Mechanisms").
- Cédric Bastoul has defended his PhD thesis at Paris 6 on December 7th.
- Julien Cohen has defended his PhD thesis at Évry on December 16th.
- Claire Pagetti, FSTTCS (Foundations of Software Technology and Theoretical Computer Science). Thursday, Dec 16th, to Saturday, Dec 18th, 2004, Chennai (Madras) India, talk on "On the Urgency Expressiveness" (joint work with Michaël Adélaïde).

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

[5] 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.

[6] 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.

[7] 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.

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

[9] S. 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

[10] P. AMIRANOFF. *Une modélisation de l'analyse de programmes par instances à travers la théorie des automates: les transducteurs comme relations des instances aux emplacements-mémoire*, Ph. D. Thesis, Cnam, December 2004.

[11] G. MOUCHARD. *Modélisation de processeurs et de systèmes*, Ph. D. Thesis, Université de Paris-Sud, september 2004.

[12] D. PARELLO. *Méthodologie d'optimisation pour architectures complexes de processeurs*, Ph. D. Thesis, Université de Paris-Sud, september 2004.

[13] S. YEHIA. *Approches alternatives pour améliorer les performances en l'absence de parallélisme d'instructions*, Ph. D. Thesis, Université de Paris-Sud, september 2004.

Articles in referred journals and book chapters

[14] G. FURSIN, M. O'BOYLE, O. TEMAM, G. WATTS. *A fast and accurate method for evaluating the upper-bound of memory performance*, in "Concurrency : Practice and Experience", vol. 16, n^o 2-3, Jan 2004, p. 271-292.

[15] S.-A.-A. TOUATI, C. EISENBEIS. *Early Periodic Register Allocation on ILP Processors*, in "Parallel Processing Letters", World Scientific, vol. 14, n^o 2, June 2004.

Publications in Conferences and Workshops

[16] C. BASTOUL. *Code Generation in the Polyhedral Model Is Easier Than You Think*, in "International Conference on Parallel Architectures and Compilation Techniques", ACM-IEEE, Sep 2004.

[17] P. CARRIBAULT, A. COHEN. *Application of Storage Mapping Optimization to Register Promotion*, in "International Conference on Supercomputing", ACM, Jun 2004.

[18] P. CARRIBAULT, C. LEMUET, J.-T. ACQUAVIVA, A. COHEN, W. JALBY. *Branch Strategies to Optimize Decision Trees for Wide-Issue Architectures*, in "International Workshop on Languages and Compilers for Parallel Computing", Sep 2004.

[19] 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.

[20] A. COHEN, S. GIRBAL, O. TEMAM. *A Polyhedral Approach to Ease the Composition of Program Transformations*, in "Euro-Par", ACM/IFIP/IEEE, Aug 2004.

[21] 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.

[22] 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.

[23] F. GRUAU, G. MOSZKOWSKI. *The Blob division*, in "Biologically Inspired Approaches to Advanced Information Technology", Lecture Notes in Computer Science, Springer Verlag, Jan 2004.

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

[25] G. MOUCHARD, D. G. PÉREZ, O. TEMAM. *A Fast SystemC Engine*, in "Proceedings of Design Automation and Test Conference in Europe (DATE)", EDA/IEEE, February 2004.

[26] 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.

[27] D. G. PÉREZ, G. MOUCHARD, O. TEMAM. *MicroLib: A Case for the Quantitative Comparison of Micro-Architecture Mechanisms*, in "Proceedings of the 37th international symposium on Microarchitecture (MICRO37), Portland, Oregon, USA", ACM, December 2004.

[28] D. G. PÉREZ, G. MOUCHARD, O. TEMAM. *MicroLib: A Case for the Quantitative Comparison of Micro-Architecture Mechanisms*, in "Third Annual Workshop on Duplicating, Deconstructing, and Debunking (WDDD), Munich, Germany", ISCA, June 2004.

[29] S. YEHIA, J.-F. COLLARD, O. TEMAM. *Load Squared: Adding Logic Close to Memory to Reduce the Latency of Indirect Loads with High Miss Ratios*, in "MEDEA Workshop, held in conjunction with the International Conference of Parallel Architectures and Compilation Techniques (PACT)", 2004 2004.

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

Internal Reports

[31] P. AMIRANOFF, A. COHEN. *Instancewise Program Analysis*, Technical report, n° 5117, Inria, january 2004, http://www.inria.fr/rrrt/rr-5117.html.

Miscellaneous

[32] H. BERRY. *The universality class of self-activated stochastic cyclers with diffusive interactions*, soumis.

[33] G. CARON-LORMIER, H. BERRY. *Amplification and oscillations in the FAK/Src kinase system during integrin signaling*, to appear, vol. 232, 2005.

[34] B. DELORD, H. BERRY, E. GUIGON, S. GENET. *Emergence of plasticity and memory in activity-dependent kinase-phosphatase cycles*, soumis.

[35] S. DONADIO. *Optimisation de code avec les langages multi-niveaux*, rapport de DEA, september 2004.

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

[37] P. PALATIN. *Nouvelles architectures de processeurs : performance et simplicité*, rapport de DEA, september 2004.

[38] É. RENARD. *Application of the MicroLib modular simulation approach to a PowerPC processor*, rapport de DEA, september 2004.

[39] E. VAROQUAUX. *Analyse de pointeurs et logique tri-valuée*, Rapport de stage d'option scientifique de l'École Polytechnique, june 2004.

[40] N. VASILACHE. *Une approche de l'optimisation de programmes compatible avec la complexité des architectures*, rapport de DEA, september 2004.