IN PARTNERSHIP WITH:
CNRS

Université de Strasbourg

Activity Report 2013

Team CAMUS

Compilation pour les Architectures MUlti-coeurS

IN COLLABORATION WITH: ICube

RESEARCH CENTER
Nancy - Grand Est

THEME
Architecture, Languages and Compilation

Table of contents

Team CAMUS

Keywords: Compiling, Embedded Systems, Hardware Accelerators, Proofs Of Programs, Formal Methods, Processors

Creation of the Team: 2009 July 01.

1. Members

Faculty Members

Philippe Clauss [Team leader, Univ. Strasbourg, Professor, HdR]
Cédric Bastoul [Univ. Strasbourg, Professor, from Sep 2013, HdR]
Alain Ketterlin [Univ. Strasbourg, Associate Professor]
Vincent Loechner [Univ. Strasbourg, Associate Professor]
Nicolas Magaud [Univ. Strasbourg, Associate Professor]
Julien Narboux [Univ. Strasbourg, Associate Professor]
Éric Violard [Univ. Strasbourg, Associate Professor, HdR]

External Collaborator

Alexandra Jimborean [Post-doc, until Dec 2013]

PhD Students

Jean-François Dollinger [Univ. Strasbourg]
Imen Fassi [Univ. Strasbourg and Univ. El Manar, from Jun 2013]
Juan Manuel Martinez Caamaño [Univ. Strasbourg]
Aravind Sukumaran-Rajam [Inria, CORDI-S]

Administrative Assistant

Isabelle Blanchard [Inria]

Others

Harrison Capera [Inria, Master student, from May 2013 until Oct 2013]
Javier Corti [Inria, Master student, from Mar 2013 until Jul 2013]
Dhruva Tirumala Bukkapatnam [Inria, Bachelor student, from Apr 2013 until Oct 2013]
Willy Wolff [Master student, from Sep 2013 until Aug 2014]

2. Overall Objectives

2.1. Overall Objectives

The CAMUS team is focusing on developping, adapting and extending automatic parallelizing and optimizing techniques, as well as proof and certification methods, for the efficient use of current and future multicore processors.

The team's research activities are organized into five main issues that are closely related to reach the following objectives: performance, correction and productivity. These issues are: static parallelization and optimization of programs (where all statically detected parallelisms are expressed as well as all "hypothetical" parallelisms which would be eventually taken advantage of at runtime), profiling and execution behavior modeling (where expressive representation models of the program execution behavior will be used as engines for dynamic parallelizing processes), dynamic parallelization and optimization of programs (such transformation processes running inside a virtual machine), object-oriented programming and compiling for multicores (where object parallelism, expressed or detected, has to result in efficient runs), and finally program transformations proof (where the correction of many static and dynamic program transformations has to be ensured).

2.2. Highlights of the Year

- Sept. 2013, Cédric Bastoul joined the CAMUS team as a Professor of the University of Strasbourg.

3. Research Program

3.1. Research directions

The various objectives we are expecting to reach are directly related to the search of adequacy between the sofware and the new multicore processors evolution. They also correspond to the main research directions suggested by Hall, Padua and Pingali in [23]. Performance, correction and productivity must be the users' perceived effects. They will be the consequences of research works dealing with the following issues:

- Issue 1: Static parallelization and optimization
- Issue 2: Profiling and execution behavior modeling
- Issue 3: Dynamic program parallelization and optimization, virtual machine
- Issue 4: Object-oriented programming and compiling for multicores
- Issue 5: Proof of program transformations for multicores

Efficient and correct applications development for multicore processors needs stepping in every application development phase, from the initial conception to the final run.

Upstream, all potential parallelism of the application has to be exhibited. Here static analysis and transformation approaches (issue 1) must be processed, resulting in a *multi-parallel* intermediate code advising the running virtual machine about all the parallelism that can be taken advantage of. However the compiler does not have much knowledge about the execution environment. It obviously knows the instruction set, it can be aware of the number of available cores, but it does not know the effective available resources at any time during the execution (memory, number of free cores, etc.).

That is the reason why a "virtual machine" mechanism will have to adapt the application to the resources (issue 3). Moreover the compiler will be able to take advantage only of a part of the parallelism induced by the application. Indeed some program information (variables values, accessed memory adresses, etc.) being available only at runtime, another part of the available parallelism will have to be generated on-the-fly during the execution, here also, thanks to a dynamic mechanism.

This on-the-fly parallelism extraction will be performed using speculative behavior models (issue 2), such models allowing to generate speculative parallel code (issue 3). Between our behavior modeling objectives, we can add the behavior monitoring, or profiling, of a program version. Indeed current and future architectures complexity avoids assuming an optimal behavior regarding a given program version. A monitoring process will allow to select on-the-fly the best parallelization.

These different parallelizing steps are schematized on figure 1.

The more and more widespread usage of object-oriented approaches and languages emphasizes the need for specific multicore programming tools. The object and method formalism implies specific execution schemes that translate in the final binary by quite distant elementary schemes. Hence the execution behavior control is far more difficult. Analysis and optimization, either static or dynamic, must take into account from the outset this distortion between object-oriented specification and final binary code: how can object or method parallelization be translated (issue 4).

Our project lies on the conception of a production chain for efficient execution of an application on a multicore architecture. Each link of this chain has to be formally verified in order to ensure correction as well as efficiency. More precisely, it has to be ensured that the compiler produces a correct intermediate code, and that the virtual machine actually performs the parallel execution semantically equivalent to the source code: every transformation applied to the application, either statically by the compiler or dynamically by the virtual machine, must preserve the initial semantics. They must be proved formally (issue 5).

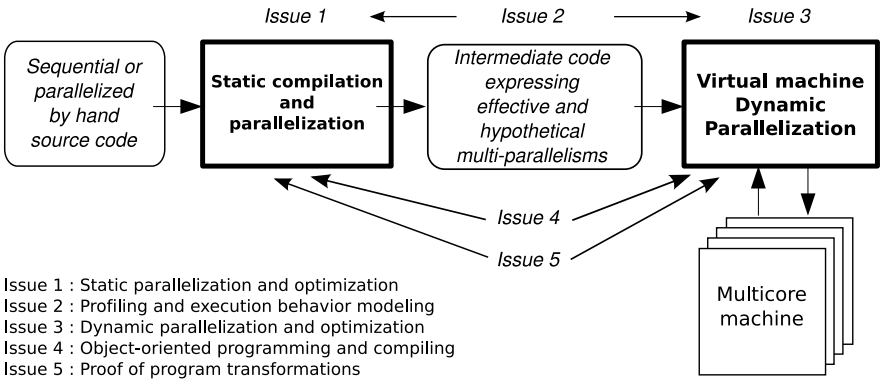

Figure 1. Automatic parallelizing steps for multicore architectures

In the following, those different issues are detailed while forming our global and long term vision of what has to be done.

3.2. Static parallelization and optimization

Participants: Vincent Loechner, Philippe Clauss, Éric Violard, Jean-François Dollinger, Aravind Sukumaran-Rajam, Juan Manuel Martinez Caamaño.

Static optimizations, from source code at compile time, benefit from two decades of research in automatic parallelization: many works address the parallelization of loop nests accessing multi-dimensional arrays, and these works are now mature enough to generate efficient parallel code [21]. Low-level optimizations, in the assembly code generated by the compiler, have also been extensively dealt for single-core and require few adaptations to support multicore architectures. Concerning multicore specific parallelization, we propose to explore two research directions to take full advantage of these architectures: adapting parallelization to multicore architecture and expressing many potential parallelisms.

3.3. Profiling and execution behavior modeling

Participants: Alain Ketterlin, Philippe Clauss, Aravind Sukumaran-Rajam.

The increasing complexity of programs and hardware architectures makes it ever harder to characterize beforehand a given program's run time behavior. The sophistication of current compilers and the variety of transformations they are able to apply cannot hide their intrinsic limitations. As new abstractions like transactional memories appear, the dynamic behavior of a program strongly conditions its observed performance. All these reasons explain why empirical studies of sequential and parallel program executions have been considered increasingly relevant. Such studies aim at characterizing various facets of one or several program runs, *e.g.*, memory behavior, execution phases, etc. In some cases, such studies characterize more the compiler than the program itself. These works are of tremendous importance to highlight all aspects that escape static analysis, even though their results may have a narrow scope, due to the possible incompleteness of their input data sets.

3.4. Dynamic parallelization and optimization, virtual machine

Participants: Aravind Sukumaran-Rajam, Juan Manuel Martinez Caamaño, Jean-François Dollinger, Alexandra Jimborean, Philippe Clauss, Vincent Loechner, Alain Ketterlin.

This link in the programming chain has become essential with the advent of the new multicore architectures. Still being considered as secondary with mono-core architectures, dynamic analysis and optimization are now one of the keys for controling those new mechanisms complexity. From now on, performed instructions are not only dedicated to the application functionalities, but also to its control and its transformation, and so in its own interest. Behaving like a computer virus, such a process should rather be qualified as a "vitamin". It perfectly knows the current characteristics of the execution environment and owns some qualitative information thanks to a behavior modeling process (issue 2). It appends a significant part of optimizing ability compared to a static compiler, while observing live resources availability evolution.

3.5. Proof of program transformations for multicores

Participants: Éric Violard, Julien Narboux, Nicolas Magaud.

Our main objective consists in certifying the critical modules of our optimization tools (the compiler and the virtual machine). First we will prove the main loop transformation algorithms which constitute the core of our system.

The optimization process can be separated into two stages: the transformations consisting in optimizing the sequential code and in exhibiting parallelism, and those consisting in optimizing the parallel code itself. The first category of optimizations can be proved within a sequential semantics. For the other optimizations, we need to work within a concurrent semantics. We expect the first stage of optimizations to produce data-race free code. For the second stage of optimizations, we will first assume that the input code is data-race free. We will prove those transformations using Appel's concurrent separation logic [24]. Proving transformations involving program which are not data-race free will constitute a longer term research goal.

4. Application Domains

4.1. Application Domains

Performance being our main objective, our developments' target applications are characterized by intensive computation phases. Such applications are numerous in the domains of scientific computations, optimization, data mining and multimedia.

Applications involving intensive computations are necessarily high energy consumers. However this consumption can be significantly reduced thanks to optimization and parallelization. Although this issue is not our prior objective, we can expect some positive effects for the following reasons:

- Program parallelization tries to distribute the workload equally among the cores. Thus an equivalent performance, or even a better performance, to a sequential higher frequency execution on one single core, can be obtained.
- Memory and memory accesses are high energy consumers. Lowering the memory consumption, lowering the number of memory accesses and maximizing the number of accesses in the low levels of the memory hierarchy (registers, cache memories) have a positive consequence on execution speed, but also on energy consumption.

5. Software and Platforms

5.1. PolyLib

Participant: Vincent Loechner.

PolyLib [1] is a C library of polyhedral functions, that can manipulate unions of rational polyhedra of any dimension, through the following operations: intersection, difference, union, convex hull, simplify, image and preimage. It was the first to provide an implementation of the computation of parametric vertices of a parametric polyhedron, and the computation of an Ehrhart polynomial (expressing the number of integer points contained in a parametric polytope) based on an interpolation method.

It is used by an important community of researchers (in France and the rest of the world) in the area of compilation and optimization using the polyhedral model. Vincent Loechner is the maintainer of this software. It is distributed under GNU General Public License version 3 or later, and it has a Debian package maintained by Serge Guelton (Parkas Projet, Inria Paris - Rocquencourt).

5.2. ZPolyTrans

Participant: Vincent Loechner.

ZPolyTrans [2] is a C library and a set of executables, that permits to compute the integer transformation of a union of parametric $\mathbb{Z}$-polyhedra (the intersection between lattices and parametric polyhedra), as a union of parametric $\mathbb{Z}$-polyhedra. The number of integer points of the result can also be computed. It is build upon PolyLib and Barvinok library. This work is based on some theoretical results obtained by Rachid Seghir and Vincent Loechner [29].

It allows for example to compute the number of solutions of a Presburger formula by eliminating existential integer variables, or to compute the number of different data accessed by some array accesses contained in an affine parametric loop nest.

The authors of this software are Rachid Seghir (Univ. Batna, Algeria) and Vincent Loechner. It is distributed under GNU General Public License version 3 or later.

5.3. NLR

Participants: Alain Ketterlin, Philippe Clauss.

We have developed a program implementing our loop-nest recognition algorithm, detailed in [7]. This standalone, filter-like application takes as input a raw trace and builds a sequence of loop nests that, when executed, reproduce the trace. It is also able to predict forthcoming values at an arbitrary distance in the future. Its simple, text-based input format makes it applicable to all kinds of data. These data can take the form of simple numeric values, or have more elaborate structure, and can include symbols. The program is written in standard ANSI C. The code can also be used as a library.

We have used this code to evaluate the compression potential of loop nest recognition on memory address traces, with very good results. We have also shown that the predictive power of our model is competitive with other models on average.

The software is available upon request to anybody interested in trying to apply loop nest recognition. It has been distributed to a dozen of colleagues around the world. In particular, it has been used by Andres Charif-Rubial for his PhD work (Université de Versailles Saint-Quentin en Yvelines), and is now included in a released tool called MAQAO (http://www.maqao.org). Our code is also used by Jean-Thomas ACQUAVIVA, at Commissariat à l'Énergie Atomique, for work on compressing instruction traces. These colleagues have slightly modified the code we gave them. We plan to release a stable version incorporating most of their changes in the near future. We also plan to change the license to avoid such drifts in the future.

5.4. Binary files decompiler

Participant: Alain Ketterlin.

[1] http://icps.u-strasbg.fr/PolyLib
[2] http://ZPolyTrans.gforge.inria.fr

Our research on efficient memory profiling has led us to develop a sophisticated decompiler. This tool analyzes x86-64 binary programs and libraries, and extracts various structured representations of the code. It works on a routine per routine basis, and first builds a loop hierarchy to characterize the overall structure of the algorithm. It then puts the code into Static Single Assignment (SSA) form to highlight the fine-grain data-flow between registers and memory. Building on these, it performs the following analyzes:

- All memory addresses are expressed as symbolic expressions involving specific versions of register contents, as well as loop counters. Loop counter definitions are recovered by resolving linearly incremented registers and memory cells, i.e., registers that act as induction variables.
- Most conditional branches are also expressed symbolically (with registers, memory contents, and loop counters). This captures the control-flow of the program, but also helps in defining what amounts to loop "trip-counts", even though our model is slightly more general, because it can represent any kind of iterative structure.

This tool embodies several passes that, as far as we know, do not exist in any existing similar tool. For instance, it is able to track data-flow through stack slots in most cases. It has been specially designed to extract a representation that can be useful in looking for parallel (or parallelizable) loops [27]. It is the basis of several of our studies.

Because binary program decompilation is especially useful to reduce the cost of memory profiling, our current implementation is based on the Pin binary instrumenter. It uses Pin's API to analyze binary code, and directly interfaces with the upper layers we have developed (e.g., program skeletonization, or minimal profiling). However, we have been careful to clearly decouple the various layers, and to not use any specific mechanism in designing the binary analysis component. Therefore, we believe that it could be ported with minimal effort, by using a binary file format extractor and a suitable binary code parser. It is also designed to abstract away the detailed instruction set, and should be easy to port (even though we have no practical experience in doing so).

We feel that such a tool could be useful to other researchers, because it makes binary code available under abstractions that have been traditionally available for source code only. If sufficient interest emerges, e.g., from the embedded systems community, or from researchers working on WCET, or from teams working on software security, we are willing to distribute and/or to help make it available under other environments.

5.5. Parwiz: a dynamic dependency analyser

Participant: Alain Ketterlin.

We have developed a dynamic dependence analyzer. Such a tool consumes the trace of memory (or object) accesses, and uses the program structure to list all the data dependences appearing during execution. Data dependences in turn are central to the search for parallel sections of code, with the search for parallel loops being only a particular case of the general problem. Most current works of these questions are either specific to a particular analysis (e.g., computing dependence densities to select code portions for thread-level speculation), or restricted to particular forms of parallelism (e.g., typically to fully parallel loops). Our tool tries to generalize existing approaches, and focuses on the program structures to provide helpful feedback either to a user (as some kind of "smart profiler"), or to a compiler (for feedback-directed compilation). For example, the tool is able to produce a dependence schema for a complete loop nest (instead of just a loop). It also targets irregular parallelism, for example analyzing a loop execution to estimate the expected gain of parallelization strategies like inspector-executor.

We have developed this tool in relation to our minimal profiling research project. However, the tool itself has been kept independent of our profiling infrastructure, getting data from it via a well-defined trace format. This intentional design decision has been motivated by our work on distinct execution environments: first on our usual x86-64 benchmark programs, and second on less regular, more often written in Java, real-world applications. The latter type of applications is likely the one that will most benefit from such tools, because their intrinsic execution environment does not offer enough structure to allow effective static analysis techniques. Parallelization efforts in this context will most likely rely on code annotations, or specific

programming language constructs. Programmers will therefore need tools to help them choose between various constructs. Our tool has this ambition. We already have a working tool-chain for C/C++/Fortran programs (or any binary program). We are in the process of developing the necessary infrastructure to connect the dynamic dependence profiler to instrumented Java programs. Other managed execution environments could be targeted as well, e.g., Microsoft's .Net architecture, but we have no time and/or workforce to devote to such time-consuming engineering efforts.

5.6. APOLLO software and LLVM

Participants: Aravind Sukumaran-Rajam, Juan Manuel Martinez Caamaño, Willy Wolff, Alexandra Jimborean, Jean-François Dollinger, Philippe Clauss.

We are developing a new framework called APOLLO (Automatic speculative POLyhedral Loop Optimizer) whose main concepts are based on our previous framework VMAD. However, several important implementation issues are now handled differently in order to improve the performance and usability of the framework, and also to open its evolution to new interesting perspectives. Thus VMAD played the role of a prototype which is now being re-written as a sustainable tool named APOLLO. As VMAD, APOLLO is dedicated to automatic, dynamic and speculative parallelization of loop nests that cannot be handled efficiently at compile-time. It is composed of a static part consisting of specific passes in the LLVM compiler suite, plus a modified Clang frontend, and a dynamic part consisting of a runtime system. It is described in more details in subsection 6.1.

Aravind Sukumaran-Rajam (PhD student), Juan Manuel Martinez Caamaño (PhD student), Jean-François Dollinger (PhD student), Willy Wolff (Master student) and Philippe Clauss are the main contributors of APOLLO. It will soon be distributed.

5.7. IBB source-to-source compiler

Participants: Imen Fassi, Philippe Clauss.

We have developed a multifor-compiler called IBB for Iterate-But-Better. IBB translates any C program containing multifor-loops into an equivalent C program in which all multifor-loops are replaced with equivalent for-loops. The resulting source code can then be compiled using any C compiler to produce executable code. IBB will soon be distributed.

5.8. Polyhedral prover

Participants: Nicolas Magaud, Julien Narboux, Éric Violard [correspondant].

We are currently developing a formal proof of program transformations based on the polyhedral model. We use the CompCert verified compiler [28] as a framework. This tool is written in the specification language of Coq.

5.9. CLooG

Participant: Cédric Bastoul.

CLooG [3] is a free software and library to generate code (or an abstract syntax tree of a code) for scanning Z-polyhedra. That is, it finds a code (e.g. in C, FORTRAN...) that reaches each integral point of one or more parameterized polyhedra. CLooG has been originally written to solve the code generation problem for optimizing compilers based on the polyhedral model. Nevertheless it is used now in various area e.g. to build control automata for high-level synthesis or to find the best polynomial approximation of a function. CLooG may help in any situation where scanning polyhedra matters. While the user has full control on generated code quality, CLooG is designed to avoid control overhead and to produce a very effective code. CLooG is widely used (including by GCC and LLVM compilers), disseminated (it is installed by default by the main Linux distributions) and considered as the state of the art in polyhedral code generation.

[3] http://www.cloog.org

5.10. OpenScop

Participant: Cédric Bastoul.

OpenScop [4] is an open specification that defines a file format and a set of data structures to represent a static control part (SCoP for short), i.e., a program part that can be represented in the polyhedral model. The goal of OpenScop is to provide a common interface to the different polyhedral compilation tools in order to simplify their interaction. To help the tool developers to adopt this specification, OpenScop comes with an example library (under 3-clause BSD license) that provides an implementation of the most important functionalities necessary to work with OpenScop.

5.11. Clan

Participant: Cédric Bastoul.

Clan [5] is a free software and library which translates some particular parts of high level programs written in C, C++, C# or Java into a polyhedral representation called OpenScop. This representation may be manipulated by other tools to, e.g., achieve complex analyses or program restructurations (for optimization, parallelization or any other kind of manipulation). It has been created to avoid tedious and error-prone input file writing for polyhedral tools (such as CLooG, LeTSeE, Candl etc.). Using Clan, the user has to deal with source codes based on C grammar only (as C, C++, C# or Java). Clan is notably the frontend of the two major high-level compilers Pluto and PoCC.

5.12. Candl

Participant: Cédric Bastoul.

Candl [6] is a free software and a library devoted to data dependences computation. It has been developed to be a basic bloc of our optimizing compilation tool chain in the polyhedral model. From a polyhedral representation of a static control part of a program, it is able to compute exactly the set of statement instances in dependence relation. Hence, its output is useful to build program transformations respecting the original program semantics. This tool has been designed to be robust and precise. It implements some usual techniques for data dependence removal, as array privatization or array expansion, offers simplified abstractions like dependence vectors and performs violation dependence analysis. Candl is notably the dependence analyzer of the two major high-level compilers Pluto and PoCC.

5.13. Clay

Participant: Cédric Bastoul.

Clay [7] is a free software and library devoted to semi-automatic optimization using the polyhedral model. It can input a high-level program or its polyhedral representation and transform it according to a transformation script. Classic loop transformations primitives are provided. Clay is able to check for the legality of the complete sequence of transformation and to suggest corrections to the user if the original semantics is not preserved. Clay is still experimental at this report redaction time but is already used during advanced compilation labs at Paris-Sud University and is one of the foundations of our ongoing work on simplifying code manipulation by programmers.

6. New Results

6.1. VMAD and APOLLO

The goal of the APOLLO project is to provide a set of annotations (pragmas) that the user can insert in the source code to perform advanced analyses and optimizations, for example dynamic speculative parallelization. It is based on the prototype VMAD developed previously by the team between 2009 and 2012.

[4] http://icps.u-strasbg.fr/~bastoul/development/openscop
[5] http://icps.u-strasbg.fr/~bastoul/development/clan
[6] http://icps.u-strasbg.fr/~bastoul/development/candl
[7] http://icps.u-strasbg.fr/~bastoul/development/clay

APOLLO includes a modified LLVM compiler and a runtime system. The program binary files are first generated by our compiler to include necessary data, instrumentation instructions, parallel code skeletons, and callbacks to the runtime system which is implemented as a dynamic library. External modules associated to specific analyses and transformations are dynamically loaded when required at runtime.

APOLLO uses sampling and multi-versioning to limit the runtime overhead (profiling, analysis, and code generation). At runtime, targeted codes are launched by successive chunks that can be either original, instrumented or optimized/parallelized versions. After each chunk execution, decisions can be taken relatively to the current optimization strategy. APOLLO is handling advanced memory access profiling [26], [17] through linear interpolation of the addresses, dynamic dependence analysis [18], version selection [26] and speculative polyhedral parallelization [22], [17].

Alexandra Jimborean defended her PhD thesis on this topic in 2012 [25].

In 2012, Aravind Sukumaran-Rajam started his PhD in our team to extend this work in order to handle more general programs which do not exhibit a pure polyhedral memory behavior. The investigated approach will explore approximative modelling of dependences still allowing advanced optimizing transformations of loop nests. A main issue concerns speculation verification when using approximative modelling.

Juan Manuel Martinez started his PhD in our team in 2013, with the goal of improving the flexibility of the parallel code generation phase inside Apollo. Indeed, although code skeletons are a good solution to fast dynamic parallel code generation, their shapes limit the kind of optimizing transformations that may be applied at runtime. Juan Manuel's work consists in defining elementary code skeletons that may be assembled at runtime to form a large panel of possible codes. These elementary skeletons will be defined as the objects forming the Apollo specific intermediate representation. Juan Manuel Martinez is a former master student of the University of Buenos Aires, Argentina (associate team EA-Ancome), and has already been working on VMAD to make the code generation support tiling. He defended his master thesis on this subject in October 2013 at the University of Buenos Aires.

Jean-Fran cois Dollinger will extend the framework to handle heterogeneous architectures (GPGPUs) in 2014. Willy Wolff, a master student from the University of Strasbourg, joined the APOLLO group in September 2013. His work is to implement just-in-time compilation in the APOLLO framework.

6.2. The Multifor programming construct

We have proposed a new programming control structure called "multifor", allowing to take advantage of optimization and parallelization opportunities that are not easily attainable using the standard programming structures.

In a multifor-loop, several loops whose bodies are run in interleaved fashion can be defined. Respective iteration domains are mapped onto each other according to a run frequency – the grain – and a relative position – the offset. Imen Fassi developped a source-to-source compiler called IBB (Iterate-But-Better) which is automatically translating any C source code containing multifor-loops into an equivalent source code where multifor-loops have been transformed into equivalent for-loops. Traditional polyhedral software tools, and particularly CLooG [21], are used to generate the corresponding code. Additionally, a promising perspective related to non-linear mapping of iteration spaces has also been developed, yielding to run a loop nest inside any other one by solving the problem of inverting "ranking Ehrhart polynomials".

This work is the PhD work of Imen Fassi, who started her work in 2013 and who is co-advised by Yosr Slama, Assistant Professor at the University El Manar in Tunis, Tunisia, and Philippe Clauss. A first paper [15] on this topic has been published at the IMPACT workshop that was held in conjunction with the HIPEAC conference in Berlin, Germany, in January 2013. Another paper describing the IBB compiler and showing the efficiency of multifor codes has been submitted to an international conference.

Obviously, reasoning on such a syntactic sugar suppose an associated precise and unambiguous meaning. Therefore a denotational semantics has been defined that resolves all such semantic issues and that is well-

suited to prove code transformations. It has been presented to the French community of Compilation during the sixth meeting in Annecy [8].

6.3. CPU+GPU adaptive computation

In this work, we aim to automatically use CPU and GPU to jointly execute a parallel code. To ensure load balance between different PUs, thus to preserve performance, it is necessary to consider the underlying hardware and the program parameters. Compiler optimizations, execution context, hardware availability and specification make it difficult to determine execution times statically. To overcome this hurdle we rely on a portable and automatic method for predicting execution times of statically generated codes on multicore CPUs and on CUDA GPUs. This approach relies on three stages: automatic code generation, offline profiling and online prediction.

This is the latest result of PhD student Jean-François Dollinger, advised by Vincent Loechner since 2011. Preliminary results, a "fastest-wins" algorithm between a multicore CPU and the best predicted GPU code version, was published in 2013 in ICPP [14]. We are currently writing a conference paper presenting the latest advances, and preparing a journal paper to be submitted in 2014, before Jean-François Dollinger's PhD defense by the end of the year.

6.4. Minimizing the synchronization overhead of X10 programs

The CAMUS team has for long focused on compiling, optimizing, and parallelizing *sequential* programs. The project described in this section is somewhat unusual in this context, in that it targets programs written in an explicitly parallel language, and applies polyhedral modeling techniques to reschedule computations, effectively introducing parallel-to-parallel program transformations. This work has been done in collaboration with the Inria COMPSYS team at ENS Lyon, and first results will be presented at the *Compiler Construction* conference (CC'14) in April 2014.

The need to leverage the computing power of multi-core processors (and distributed computers) has lead to the design of explicitly parallel programming languages. Such languages often employ a fork/join model, and include syntax to launch and synchronize tasks (also called activities) with well-defined semantics. This brings parallel constructions under the control of the compiler, and introduces new optimization opportunities. Our work has focused on the various synchronization primitives available to the programmer, and more specifically on how one type of synchronization can be replaced with another for specific classes of programs, the goal being to minimize the synchronization overhead. We have demonstrated significant speedups on programs written using the X10 programming language, and have obtained similar results on equivalent Habanero-Java programs.

More specifically, our proposed optimization works by eliminating the use of clocks in X10 programs whose activities can be characterized with a polyhedral time-domain. The X10 language basically has two activity synchronization primitives: one is the explicit use of "clocks" (synchronization barriers) during activity execution, the other is the implicit use of activity containers that synchronize only on the end of activities. Under reasonable conditions on the patterns of activity creation and control, we have shown that long-running activities using clocks can be replaced by short-lived activities synchronized only on the end of their containers, and that this transformation provides a significant gain at run time. This work has two main contributions. First, it extends a known transformation framework to the case where the original program is already parallel. Second, it shows that the polyhedral model has applications far beyond its current use in data dependence and memory locality analyzes. This work also opens up new research directions. First, it turns out that our transformation is far more general than the use we currently make of it, and therefore that it provides a solid basis for other optimizations of parallel programs. Second, the polyhedral model we have developed provides an immediate cost model for synchronization primitives, which is not used in our current work, but may provide sound heuristics to adapt the optimization phase to the characteristics of specific run time components. We plan to explore these aspects in the near future.

[8] http://compilation.gforge.inria.fr/2013_04_Annecy

This work has been done in collaboration with Paul Feautrier, member of the COMPSYS Inria team, in ENS Lyon. The CAMUS team has invited Paul Featrier for one week in June 2013 in Strasbourg. We are currently seeking funding to organize more frequent stays at either Lyon or Strasbourg.

This work has been invited for presentation at the LCPC workshop held in Lyon in July 2013 (http://labexcompilation.ens-lyon.fr/cpc2013). An extended version of this work has been accepted for publication at the *Compiler Construction* conference, to be held in April 2014.

6.5. Switcheable scheduling

Parallel applications used to be executed alone until their termination on partitions of supercomputers. The recent shift to multicore architectures for desktop and embedded systems is raising the problem of the coexistence of several parallel programs. Operating systems already take into account the *affinity* mechanism to ensure a thread will run only onto a subset of available processors (e.g., to reuse data remaining in the cache since its previous execution). But this is not enough, as demonstrated by the large performance gaps between executions of a given parallel program on desktop computers running several processes. To support many parallel applications, advances must be made on the system side (scheduling policies, runtimes, memory management...). However, automatic optimization and parallelization can play a significant role by generating programs with dynamic-auto-tuning capabilities to adapt themselves to the complete execution context, including the system load.

Our approach is to design at compile-time programs that can adapt at run-time to the execution context. The originality of our solution is to rely on *switcheable scheduling*, a selected set of program restructuring which allows to swap between program versions at some meeting points without backtracking. A first step selects pertinent versions according to their performance behavior on some execution contexts. The second step builds the auto-adaptive program with the various versions. Then at runtime the program selects the best version by a low overhead sampling and profiling of the versions, ensuring every computation is useful.

This work is an addition to the research directions of CAMUS related to dynamic optimization. It has been started at Paris-Sud University by Cédric Bastoul before he joined CAMUS during this year. This is an ongoing work with the PhD student Lénaïc Bagnères (GRAND-LARGE Team at Inria Saclay-Île-de-France, co-advised by Christine Eisenbeis and Cédric Bastoul). The first results have been presented in 2013 at the HiPEAC Computing System Week [9] and at the Rencontres Françaises de Compilation [10].

6.6. Interactive Code Restructuring

This work falls within the exploration and development of semi-automatic programs optimization techniques. It consists in designing and evaluating new visualization and interaction techniques for code restructuring, by defining and taking advantage of visual representations of the underlying mathematical model. The main goal is to assist programmers during program optimization tasks in a safe and efficient way, even if they neither have expertise into code restructuring nor knowledge of the underlying theories. This project is an important step for the efficient use and wider acceptance of semi-automatic optimization techniques, which are still tedious to use and incomprehensible for most programmers. More generally, this research is also investigating new presentation and manipulation techniques for code, algorithms and programs, which could lead to many practical applications: collaboration, tracking and verification of changes, visual search in large amount of code, teaching, etc.

This is a rather new research direction which strengthen CAMUS's static parallelization and optimization issue. It has been initiated at Paris-Sud University as a collaboration between Compilation, represented by Cédric Bastoul before he joined CAMUS during this year, and Human-Machine Interaction, represented by Stéphane Huot from the IN-SITU Team at Inria Saclay-Île-de-France. This work is essentially the PhD topic of Alexander Zinenko (IN-SITU Team at Inria Saclay-Île-de-France, co-advised by Stéphane Huot and Cédric Bastoul, CORDI Grant) which started in 2013.

[9] http://www.hipeac.net/thematic-session/let-us-push-thread-level-speculation
[10] http://compil13.cri.mines-paristech.fr

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

A contract with the French company Kalray (http://www.kalray.eu) was established early 2013. It provided to the team the Kalray 256-core MPPA platform and necessary funding to recruit a student for a 6-months internship: Dhruva Tirumala Bukkapatnam. A deep evaluation of the platform regarding performance and programming strategies has been accomplished. Moreover, an extension of the source-to-source compiler Pluto (http://pluto-compiler.sourceforge.net), allowing to automatically generate code adapted for the MPPA has been mostly implemented.

8. Partnerships and Cooperations

8.1. National Initiatives

Philippe Clauss, Alain Ketterlin, Cédric Bastoul and Vincent Loechner are involved in the Inria Large Scale Initiative entitled "Large scale multicore virtualization for performance scaling and portability" and regrouping several french researchers in compilers, parallel computing and program optimization. The project started officially in January 2013. In this context and since January 2013, Philippe Clauss is co-advising with Erven Rohou of the Inria team ALF, Nabil Hallou's PhD thesis focusing on dynamic optimization of binary code.

The CAMUS team is taking part of the NANO 2017 national research program with the company STMicroelectronics, starting January 2014.

8.2. European Initiatives

8.2.1. Collaborations in European Programs, except FP7

Program: ITEA

Project acronym: MANY

Project title: Many-core Programming and Resource Management for High-Performance Embedded Systems

Duration: 09/2011 - 08/2014

Coordinator: XDIN

Other partners: France: Thales Communications and Security, CAPS Entreprise, Telecom SudParis; Spain: UAB; Sweden: XDIN; Korea: ETRI, TestMidas, SevenCore; Netherlands: Vector Fabrics, ST-Ericsson, TU Eindhoven; Belgium: UMONS.

Abstract: Adapting Industry for the for the disruptive landing of many-core processors in Embedded Systems in order to provide scalable, reusable and very fast sofware development.

8.3. International Initiatives

8.3.1. Inria Associate Teams

8.3.1.1. ANCOME

Title: Memory and applications memory behavior

Inria principal investigator: Philippe Clauss

International Partner (Institution - Laboratory - Researcher):

University of Buenos Aires (Argentina) - Departamento de Computación, Facultad de Ciencias Exactas y Naturales - Philippe Clauss

Duration: 2011 - 2013
See also: http://lafhis.dc.uba.ar/wiki/index.php/EA-Ancome
This associate team focuses on developing original methods for the analysis of programs memory behavior, in particular in the context of applications using dynamic memory allocation. The proposed approaches consist in analyzing and modeling the runtime behavior, where extracted properties are then verified thanks to static analysis processes. Thus pure static approaches limits will be overpassed. Further, the case of multi-threaded applications run on multi-core architectures will be studied in order to elaborate and extend our analysis techniques and to extract properties specific to this context. The issues are mainly concerned with the conception of real-time applications using dynamic memory allocation.

8.3.2. Inria International Partners

8.3.2.1. Informal International Partners

The CAMUS team maintains regular contacts with the following entities:

- Reservoir Labs, New York, NY, USA
- Intel, Santa Clara, CA, USA
- UPMARC, University of Uppsala, Sweden
- University of Batna, Algeria
- University El Manar, Tunis, Tunisia
- Ohio State University, Colombus, USA
- Louisiana State University, Baton Rouge, USA
- Indian Institute of Science (IIIS) Bangalore, India
- University of Delaware, DE, USA

8.4. International Research Visitors

8.4.1. Visits of International Scientists

Diego Garbervetsky, University of Buenos Aires, Argentina, has made three visits in the CAMUS team at the following dates: Dec. 1-14, Oct. 14-20 and Jan. 15-23.

Rachid Seghir, University of Batna, Algeria, visited the team from May the 30th to June the 13th.

8.4.1.1. Internships

Javier Corti
Subject: Certified Compiler for polyhedral transformations
Date: from Mar 2013 until Aug 2013
Institution: Universidad National de Rosario (Argentina)

Imen Fassi
Subject: Multifor for Multicore
Date: from Mar 2013 until Aug 2013
Institution: Université de Tunis El Manar - Faculté des Sciences (Tunisia)

Dhruva Tirumala Bukkapatnam
Subject: Evaluation of the Kalray MPPA and extension of the Pluto compiler
Date: from Apr 2013 until Oct 2013
Institution: Birla Institute of Technology and Science, Birla (India)

8.4.2. Visits to International Teams

Philippe Clauss has spent one week in the LAFHIS team, University of Buenos Aires, Argentina, in October 2013.

9. Dissemination

9.1. Scientific Animation

Philippe Clauss, Cédric Bastoul and Vincent Loechner have been part of the program committee of IMPACT 2013 and IMPACT 2014 workshops (International Workshop on Polyhedral Compilation Techniques), held in conjunction with the international conferences HiPEAC 2013 and HiPEAC 2014.

Cédric Bastoul has been co-organizing the HIP3ES 2014 workshop (High Performance Energy Efficient Embedded Systems) held in conjunction with the international conference HiPEAC 2014. Vincent Loechner has been part of its program committee.

Cédric Bastoul is part of the program committee of PARMA+DITAM 2014 (5th Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures 3rd Workshop on Design Tools and Architerctures for Multicore Embedded Computing Platforms), Vienna, Austria.

Cédric Bastoul has been part of the program committee of PACT 2013 (International Conference on Parallel Architecture and Compilation Techniques), Edinburg, Scotland.

Cédric Bastoul has been part of the program committee of PARMA 2013 (Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures), Berlin, Germany.

Philippe Clauss has been part of the program committee of the MSPC 2013 workshop (Memory Systems Performance and Correctness) held in conjunction with the international conference PLDI 2013, Seattle, USA.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Philippe Clauss did not teach in 2013 since he was detached at Inria during the whole year (*délégation*). Alain Ketterlin had teaching activities during the first semester of 2013, while he was detached in the Inria team ALF in Rennes during the second semester.

Licence : Cédric Bastoul, Operating Systems, 30h, L2, Paris-Sud University, France

Licence : Cédric Bastoul, Information Security, 8h, L3, Paris-Sud University, France

Licence : Cédric Bastoul, Networks, 60h, L2, Paris-Sud University, France

Licence : Cédric Bastoul, System Programming, 28h, L2, Strasbourg University, France

Licence : Cédric Bastoul, Network Programming, 42h, L2, Strasbourg University, France

Master : Cédric Bastoul, Compilation, 72h, M1, Strasbourg University, France

Master : Cédric Bastoul, Parallelism, 13h, M2, Strasbourg University, France

Licence : Éric Violard, Functional Programming, 42h, L2, Strasbourg University, France

Licence : Éric Violard, Computer Architectures, 21h, L2, Strasbourg University, France

Licence : Éric Violard, Algorithms and Data Structures, 37h, L2, Strasbourg University, France

Master : Éric Violard, Compiler Design, 57h, M1, Strasbourg University, France

Master : Éric Violard, Semantics, 48h, M1, Strasbourg University, France

Licence : Vincent Loechner, System Programming, 15h, L2, Strasbourg University, France

Licence : Vincent Loechner, Operating Systems, 28h, L2, Strasbourg University, France

Master : Vincent Loechner, Python Programming, 42h, M1, Strasbourg University, France

Master : Vincent Loechner, Real-time Programming, 22h, M1, Strasbourg University, France

Master : Vincent Loechner, Parallelism, 41h, M1, Strasbourg University, France

Master : Vincent Loechner, Parallel Programming, 52h, M2, Strasbourg University, France

Master : Vincent Loechner, Embedded Systems, 32h, M2, Strasbourg University, France

9.2.2. Supervision

PhD in progress : Aravind Sukumaran-Rajam, Enlarging the scope of polyhedral speculative parallelization, November 2012, Philippe Clauss and Alain Ketterlin

PhD in progress : Juan Manuel Martinez Caamaño, Dynamic and flexible generation of parallel loops using a dedicated intermediate representation, November 2013, Philippe Clauss and Philippe Helluy (IRMA lab., University of Strasbourg)

PhD in progress : Jean-François Dollinger, Heterogeneous speculative parallelization, September 2010, Vincent Loechner and Philippe Clauss

PhD in progress : Imen Fassi, Multifor for Multicore, June 2013, Philippe Clauss and Yosr Slama (University El Manar, Tunisia)

PhD in progress : Nabil Hallou, Dynamic binary optimizations, January 2013, Erven Rohou (ALF team) and Philippe Clauss

PhD in progress : Lénaïc Bagnères, Automatic parallelization and optimization for manycore architectures, November 2012, Christine Eisenbeis and Cédric Bastoul

PhD in progress : Alexander Zinenko, Interactive program manipulation, September 2013, Stéphane Huot and Cédric Bastoul

9.2.3. Juries

Philippe Clauss participated to the following HDR jury in 2013:

Date	**Candidate**	**Place**	**Role**
Oct. 11	Fabien Coelho	École des Mines de Paris	Reviewer

Philippe Clauss participated to the following PhD jurys in 2013:

Date	**Candidate**	**Place**	**Role**
Oct. 17	Alexandre Carbon	Univ. Pierre et Marie Curie, Paris	Reviewer
June 28	Antoine Morvan	ENS Cachan	Reviewer
June 12	Damien Hedde	Institut polytechnique de Grenoble	Examiner

Cédric Bastoul participated to the following HDR jury in 2013:

Date	**Candidate**	**Place**	**Role**
Oct. 11	Fabien Coelho	École des Mines de Paris	Examiner

Cédric Bastoul participated to the following PhD jurys in 2013:

Date	**Candidate**	**Place**	**Role**
March 13	Ramakrishna Upadrasta	Paris-Sud University	Examiner
November 27	Dounia Khaldi	École des Mines de Paris	President

Alain Ketterlin participated to the following PhD jury in 2013:

Date	**Candidate**	**Place**	**Role**
Dec. 3	Nathanaël Premilieu	Université de Rennes 1	Examiner

9.3. Popularization

- Cédric Bastoul participated to the *Rencontres Inria-Industrie* in June 2013
- Cédric Bastoul participated to *Fête de la Science* at University of Paris-Sud in October 2013

10. Bibliography

Major publications by the team in recent years

[1] J. C. BEYLER, P. CLAUSS. *Performance driven data cache prefetching in a dynamic software optimization system*, in "ICS '07: Proceedings of the 21st annual international conference on Supercomputing", New York, NY, USA, ACM, 2007, pp. 202–209, http://doi.acm.org/10.1145/1274971.1275000

[2] J. C. BEYLER, M. KLEMM, P. CLAUSS, M. PHILIPPSEN. *A meta-predictor framework for prefetching in object-based DSMs*, in "Concurr. Comput. : Pract. Exper.", September 2009, vol. 21, pp. 1789–1803

[3] P. CLAUSS. *Counting solutions to linear and nonlinear constraints through Ehrhart polynomials: applications to analyze and transform scientific programs*, in "ICS '96: Proceedings of the 10th international conference on Supercomputing", New York, NY, USA, ACM, 1996, pp. 278–285, http://doi.acm.org/10.1145/237578.237617

[4] P. CLAUSS, F. J. FERNÁNDEZ, D. GARBERVETSKY, S. VERDOOLAEGE. *Symbolic polynomial maximization over convex sets and its application to memory requirement estimation*, in "IEEE Transactions on Very Large Scale Integration (VLSI) Systems", Aug 2009, vol. 17, n° 8, pp. 983-996, http://hal.inria.fr/inria-00504617

[5] P. CLAUSS, V. LOECHNER. *Parametric Analysis of Polyhedral Iteration Spaces*, in "J. VLSI Signal Process. Syst.", 1998, vol. 19, n° 2, pp. 179–194, http://dx.doi.org/10.1023/A:1008069920230

[6] P. CLAUSS, I. TCHOUPAEVA. , *A Symbolic Approach to Bernstein Expansion for Program Analysis and Optimization*, LNCS, Springer, April 2004, vol. 2985, pp. 120-133

[7] A. KETTERLIN, P. CLAUSS. *Prediction and trace compression of data access addresses through nested loop recognition*, in "6th annual IEEE/ACM international symposium on Code generation and optimization", États-Unis Boston, ACM, April 2008, pp. 94-103, http://dx.doi.org/10.1145/1356058.1356071, http://hal.inria.fr/inria-00504597/en

[8] V. LOECHNER, B. MEISTER, P. CLAUSS. *Precise data locality optimization of nested loops*, in "Journal of Supercomputing", January 2002, vol. 21, n° 1, pp. 37–76, Kluwer Academic Pub.

[9] V. LOECHNER, D. K. WILDE. *Parameterized Polyhedra and their Vertices*, in "International Journal of Parallel Programming", December 1997, vol. 25, n° 6

[10] S. VERDOOLAEGE, R. SEGHIR, K. BEYLS, V. LOECHNER, M. BRUYNOOGHE. *Counting Integer Points in Parametric Polytopes Using Barvinok's Rational Functions*, in "Algorithmica", 2007, vol. 48, n° 1, pp. 37–66, http://dx.doi.org/10.1007/s00453-006-1231-0

[11] É. VIOLARD. *A Semantic Framework to Address Data Locality in Data Parallel Languages*, in "Parallel Computing", 2004, vol. 30, n° 1, pp. 139-161

Publications of the year

Articles in International Peer-Reviewed Journals

[12] A. KETTERLIN, P. CLAUSS. *Recovering memory access patterns of executable programs*, in "Science of Computer Programming", February 2014, vol. 80, pp. 440-456 [*DOI :* 10.1016/J.SCICO.2012.08.002], http://hal.inria.fr/hal-00909961

[13] E. PARK, J. CAVAZOS, L.-N. POUCHET, C. BASTOUL, A. COHEN, P. SADAYAPPAN. *Predictive Modeling in a Polyhedral Optimization Space*, in "International Journal of Parallel Programming", 2013, vol. 41, n^{o} 5, pp. 704–750 [*DOI :* 10.1007/S10766-013-0241-1], http://hal.inria.fr/hal-00918653

International Conferences with Proceedings

[14] J.-F. DOLLINGER, V. LOECHNER. *Adaptive Runtime Selection for GPU*, in "42nd International Conference on Parallel Processing", Lyon, France, IEEE, October 2013, pp. 70-79 [*DOI :* 10.1109/ICPP.2013.16], http://hal.inria.fr/hal-00869652

[15] I. FASSI, P. CLAUSS, M. KUHN, Y. SLAMA. *Multifor for Multicore*, in "IMPACT 2013, Third International Workshop on Polyhedral Compilation Techniques", Berlin, Germany, A. GRÖSSLINGER, L.-N. POUCHET (editors), Epubli, January 2013, pp. 37-44, http://hal.inria.fr/hal-00780748

[16] P. FEAUTRIER, E. VIOLARD, A. KETTERLIN. *Improving X10 Program Performances by Clock Removal*, in "Compiler Construction 2014", Grenoble, France, January 2014, http://hal.inria.fr/hal-00924206

[17] A. JIMBOREAN, P. CLAUSS, J.-F. DOLLINGER, V. LOECHNER, J. M. MARTINEZ. *Dynamic and Speculative Polyhedral Parallelization Using Compiler-Generated Skeletons*, in "International Symposium on High-level Parallel Programming and Applications, HLPP", Paris, France, July 2013, http://hal.inria.fr/hal-00825738

[18] A. JIMBOREAN, P. CLAUSS, J. M. MARTINEZ, A. SUKUMARAN-RAJAM. *Online Dynamic Dependence Analysis for Speculative Polyhedral Parallelization*, in "Euro-Par 2013", Aachen, Germany, F. WOLF, B. MOHR, D. AN MEY (editors), Lecture Notes in Computer Science, Springer, August 2013, vol. 8097, pp. 191-202 [*DOI :* 10.1007/978-3-642-40047-6_21], http://hal.inria.fr/hal-00825744

[19] E. RIOU, E. ROHOU, P. CLAUSS, N. HALLOU, A. KETTERLIN. *PADRONE: a Platform for Online Profiling, Analysis, and Optimization*, in "DCE 2014 - International workshop on Dynamic Compilation Everywhere", Vienne, Austria, January 2014, http://hal.inria.fr/hal-00917950

Scientific Popularization

[20] J. NARBOUX. *Les assistants de preuve, ou comment avoir confiance en ses démonstrations*, in "Séminaire L", Strasbourg, France, March 2013, http://hal.inria.fr/hal-00809448

References in notes

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

[22] P. CLAUSS, A. JIMBOREAN. *Does dynamic and speculative parallelization enable advanced parallelizing and optimizing code transformations?*, in "DCE - 1st International Workshop on Dynamic compilation from SoC to Web Browser via HPC, in conjonction with HiPEAC 2012", Paris, France, Henri-Pierre Charles and Philippe Clauss and Frédéric Pétrot, January 2012, http://hal.inria.fr/hal-00664339

[23] M. HALL, D. PADUA, K. PINGALI. *Compiler research: the next 50 years*, in "Commun. ACM", 2009, vol. 52, n^{o} 2, pp. 60–67, http://doi.acm.org/10.1145/1461928.1461946

[24] A. HOBOR, A. W. APPEL, F. Z. NARDELLI. *Oracle Semantics for Concurrent Separation Logic*, in "ESOP", 2008, pp. 353-367

[25] A. JIMBOREAN. , *Adapting the polytope model for dynamic and speculative parallelization*, Université de Strasbourg, September 2012, http://tel.archives-ouvertes.fr/tel-00733850

[26] A. JIMBOREAN, L. MASTRANGELO, V. LOECHNER, P. CLAUSS. *VMAD: an Advanced Dynamic Program Analysis & Instrumentation Framework*, in "CC - 21st International Conference on Compiler Construction", Tallinn, Estonia, M. F. P. O'BOYLE (editor), Lecture Notes in Computer Science, Springer, March 2012, vol. 7210, pp. 220-237, http://hal.inria.fr/hal-00664345

[27] A. KETTERLIN, P. CLAUSS. *Recovering the Memory Behavior of Executable Programs*, in "10th IEEE Working Conference on Source Code Analysis and Manipulation, SCAM", Roumanie Timisoara, IEEE Computer Society Press, Sep 2010, http://hal.inria.fr/inria-00502813

[28] X. LEROY. , *The Compcert verified compiler, software and commented proof*, January 2010, http://compcert.inria.fr

[29] R. SEGHIR, V. LOECHNER, B. MEISTER. *Integer Affine Transformations of Parametric Z-polytopes and Applications to Loop Nest Optimization*, in "ACM Transactions on Architecture and Code Optimization", June 2012, vol. 9, n^{o} 2, pp. 8.1-8.27 [*DOI :* 10.1145/2207222.2207224], http://hal.inria.fr/inria-00582388