IN PARTNERSHIP WITH:
CNRS

Université de Strasbourg

Activity Report 2014

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, HdR]
- Alain Ketterlin [Univ. Strasbourg, Associate Professor]
- Vincent Loechner [Univ. Strasbourg, Associate Professor]
- Nicolas Magaud [Univ. Strasbourg I, Associate Professor]
- Julien Narboux [Univ. Strasbourg I, Associate Professor]
- Éric Violard [Univ. Strasbourg, Associate Professor, HdR]

PhD Students
- Yann Barsamian [Univ. Strasbourg, from Oct 2014]
- Jean-François Dollinger [Univ. Strasbourg, from Oct 2011]
- Imen Fassi [Inria, Univ. Strasbourg and Univ. El Manar, from Jun 2013]
- Juan Manuel Martinez Caamaño [Univ. Strasbourg, from Nov 2013]
- Aravind Sukumaran-Rajam [Inria, from Nov 2012]

Administrative Assistant
- Isabelle Blanchard [Inria]

Others
- Luis Esteban Campostrini [Inria, Master student, from Aug 2014 until Dec 2014]
- Matías Perez [Inria, Master student, from May 2014 until Nov 2014]
- César Sabater [Inria, Master student, from May 2014 until Oct 2014]
- Willy Wolff [Inria, Master student, , from Jan 2014 until Dec 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).

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 [28]. 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).

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

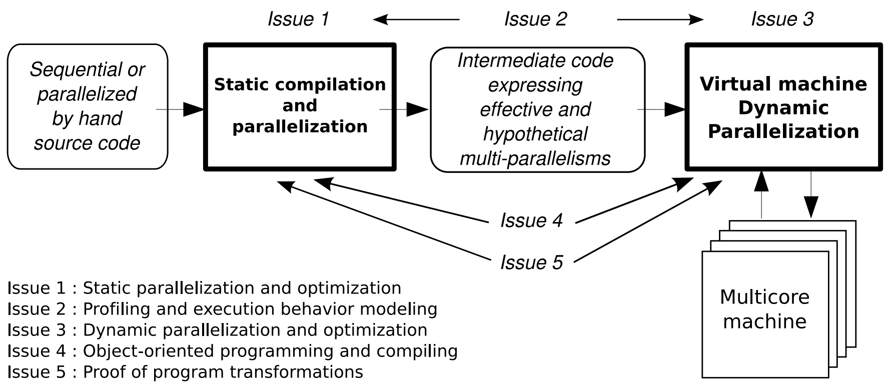
Issue 1
Issue 2
Issue 3
Sequential or parallelized by hand source code
Static compilation and parallelization
Intermediate code expressing effective and hypothetical multi-parallelisms
Virtual machine Dynamic Parallelization
Issue 4
Issue 5
Multicore machine
Issue 1 : Static parallelization and optimization
Issue 2 : Profiling and execution behavior modeling
Issue 3 : Dynamic parallelization and optimization
Issue 4 : Object-oriented programming and compiling
Issue 5 : Proof of program transformations

Figure 1. Automatic parallelizing steps for multicore architectures

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 [27]. 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 [29]. 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. New 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. 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. Vincent Loechner is the maintainer of this software. It is distributed under GNU General Public License version 3 or later.

Apart from normal maintenance, it was parallelized using OpenMP with the support of Master student Adilla Susungi, funded by the ICPS team (ICube laboratory, University of Strasbourg).

5.2. APOLLO software and LLVM

Participants: Aravind Sukumaran-Rajam, Juan Manuel Martinez Caamaño, Willy Wolff, Luis Esteban Campostrini, Matías Perez, Alexandra Jimborean, Philippe Clauss.

We are developing a 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. 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. Its last extensions are presented in subsection 6.2.

5.3. IBB source-to-source xfor compiler

Participants: Imen Fassi, Philippe Clauss, Cédric Bastoul.

Imen Fassi has developped a source-to-source xfor compiler called IBB (Iterate-But-Better) which is automatically translating any C source code containing xfor-loops into an equivalent source code where xfor-loops have been transformed into equivalent for-loops. The polyhedral code generator CLooG [27] is used to generate the corresponding code. The IBB compiler has been improved in some aspects in 2014: loop bounds can now be min and max functions, IBB uses the OpenScop format to encode statements and iteration domains.

The xfor structure is also currently incorporated in the polyhedral parser Clan [2], opening the door of xfor usage to polyhedral compilation tools. Additionally, an xfor programming wizard is currently being developed, providing automatic dependence analysis and code verification to users, thanks to the dependence analyzer Candl [3].

5.4. CLooG

Participant: Cédric Bastoul.

CLooG [4] 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.

5.5. OpenScop

Participant: Cédric Bastoul.

[1] http://icps.u-strasbg.fr/PolyLib
[2] http://icps.u-strasbg.fr/~bastoul/development/clan
[3] http://icps.u-strasbg.fr/~bastoul/development/candl
[4] http://www.cloog.org

OpenScop [5] 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.

5.6. Clan

Participants: Cédric Bastoul, Imen Fassi.

Clan [6] 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.

5.7. 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. Highlights of the Year

One of Philippe Clauss' early papers on Ehrhart polynomials has been celebrated, 18 years later, in a selection of papers for the International Conference on Supercomputing (ICS) 25th anniversary retrospective [13]. 35 papers have been selected among roughly 1800 papers published between 1987 and 2011. The paper is:

"Counting Solutions to Linear and Nonlinear Constraints Through Ehrhart Polynomials: Applications to Analyze and Transform Scientific Programs", by Philippe Clauss, ICS'96, which introduced Ehrhart polynomials in the field of program analysis and optimization.

Philippe Clauss wrote an additional retrospective [12] related to this research which complements the paper in the ICS special issue.

6.2. APOLLO (Automatic speculative POLyhedral Loop Optimizer)

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 named VMAD which was developed previously by the team between 2009 and 2012. Alexandra Jimborean defended her PhD thesis on this topic in 2012 [30].

[5] http://icps.u-strasbg.fr/~bastoul/development/openscop
[6] http://icps.u-strasbg.fr/~bastoul/development/clan
[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, multi-versioning and code skeletons 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. These latter versions are generated on-the-fly through fast instantiation of the code skeletons. After each chunk execution, decisions can be taken relatively to the current optimization strategy. APOLLO is handling advanced memory access profiling through linear interpolation of the addresses, dynamic dependence analysis, version selection and speculative polyhedral parallelization [9].

Several extensions and improvements have been implemented inside Apollo in 2014:

- the scheduler of the polyhedral compiler Pluto has been integrated inside the framework. Thus, the runtime decision regarding what optimizing and parallelizing transformation is now entirely depending on Pluto, whose input is generated by the instrumentation and interpolation phase of Apollo [20].
- the static compilation phase of Apollo has been significantly enforced. Linear dependencies between values of scalars and memory addresses are identified in order to alleviate the cost of the instrumented code version. Additionally, memory reference functions that can be disambiguated at compile-time are now fully handled. These improvements are using analysis passes of the LLVM compiler, as well as passes that were specifically developed.
- Apollo is now using the LLVM JIT compiler to further optimize the instantiated code skeletons. Previously, code skeletons were generated as binary executable at compile-time with global variables instantiated at runtime. This approach yielded sub-optimal code including unnecessary or invariant computations. Code skeletons are now kept in LLVM intermediate form until being instantiated and compiled at runtime using the LLVM JIT compiler, thus resulting in faster optimized codes.
- Other memory behavior modeling approaches are now being studied and implemented, in order to allow Apollo handling codes that do not have a completely linear behavior. Three main cases are addressed:
  - quasi-linear behavior in which memory accesses which do not fit the linear prediction are checked on-the-fly, i.e., if these delinquent accesses do not invalidate the current parallel schedule.
  - linear regression behavior in which memory accesses are staying inside a “tube” bordered by linear functions.
  - behavior in which memory accesses are staying inside disjointed address ranges.

6.3. The XFOR programming structure

We have proposed a new programming control structure called “xfor” or “multifor”, providing users a way to schedule explicitly the statements of a loop nest, and take advantage of optimization and parallelization opportunities that are not easily attainable using the standard programming structures. 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.

Data locality optimization is a well-known goal when handling programs that must run as fast as possible or use a minimum amount of energy. However, usual techniques never address the significant impact of numerous stalled processor cycles that may occur when consecutive load and store instructions are accessing the same memory location. In [15], we show that two versions of the same program may exhibit similar memory performance, while performing very differently regarding their execution times because of the stalled processor cycles generated by many pipeline hazards. The xfor structure enables the explicit control of the way data locality is optimized in a program and thus, to control the amount of stalled processor cycles. In [15], we also show the benefits of xfor regarding execution time and energy saving.

While many advanced and fully automatic program analysis and optimization techniques have been developed thanks to the accuracy and expressiveness of the polyhedral model, these techniques may fail in producing efficient codes in some circumstances. The xfor structure eases the manual application of optimizing transformations on loop nests for expert programmers and allows to generate executable codes that may be significantly faster than those generated automatically using well-established polyhedral strategies. we highlight five main gaps regarding these strategies and discuss some ideas on how to bridge them in [14].

6.4. CPU+GPU adaptive computation

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 of the target code and online prediction.

This is mainly the work 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. Our latest advances, load balancing code between multiple cores CPUs and multiple GPUs will be presented at the IMPACT 2015 workshop [25] in conjunction with the HiPEAC conference. We are currently preparing an extended journal paper to present this work, and Jean-François Dollinger will defend his PhD in 2015.

6.5. 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 were 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 work focused on synchronization primitives of X10. 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 showed 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.

We have studied the converse transformation, i.e. starting with an unclocked X10 program, obtaining a system of sequential threads executing in parallel and synchronizing with clocks. This transformation is interesting since it yields to further optimization opportunities. We have elaborated a system of rules to execute the transformation. Applying these rules to "regular" programs gives good results, but fails on some paradigmatic X10 codes. For irregular programs, some parallelism may be lost. We now are investigating a new set of rules to give a correct result for arbitrary X10 programs. A main difficulty is bringing the proof that the set of upgraded rules will give a correct result.

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 Feautrier one more time for one week in June 2014 in Strasbourg.

6.6. Hardware/Software helper thread prefetching

Heterogeneous Many Cores (HMC) architectures that mix many simple/small cores with a few complex/large cores are emerging as a design alternative that can provide both fast sequential performance for single threaded workloads and power-efficient execution for through-put oriented parallel workloads. The availability of many small cores in a HMC presents an opportunity to utilize them as low-power helper cores to accelerate memory-intensive sequential programs mapped to a large core. However, the latency overhead of accessing small cores in a loosely coupled system limits their utility as helper cores. Also, it is not clear if small cores can execute helper threads sufficiently in advance to benefit applications running on a larger, much powerful, core.

In this project, we designed a hardware/software framework called core-tethering to support efficient helper threading on heterogeneous manycores. Core-tethering provides a co-processor like interface to the small cores that (a) enables a large core to directly initiate and control helper execution on the helper core and (b) allows efficient transfer of execution context between the cores, thereby reducing the performance overhead of accessing small cores for helper execution. Our evaluation on a set of memory intensive programs chosen from the standard benchmark suites shows that helper threads using moderately sized small cores can significantly accelerate a larger core compared to using a hardware prefetcher alone. We find that a small core provides a good trade-off against using an equivalent large core to run helper threads in a HMC. Additionally, helper prefetching on small cores when used along with hardware prefetching, can provide an alternate design point to growing instruction window size for achieving higher sequential performance on memory intensive applications.

This work is a collaboration between the ALF team in Rennes and CAMUS in Strasbourg. Our contribution is mainly on the generation of helper thread code (as a followup to our work on program skeletonization). The result of the work has been published in October 2014 in the Proceedings of the SBAC-PAD conference [17].

6.7. Loop-based Modeling of Parallel Communication Traces

Parallel communication traces are traces of the various actions performed by parallel programs (typically written using MPI or some such library). The traces usually contain actions like message sending and receiving, and entering and exiting collective operations. The goal of this project is to build a model of the parallel program from the traces of the various processes that form the program. Consolidating on our previous work on sequential traces, we have developed an algorithm that takes the traces of the individual processes and merges them into a global model.

The main characteristics of our algorithm is that the result takes the form of loops enclosing various parallel constructs and communication actions. The driving goal of this work is to use the model for various analyzes, mainly to draw qualitative conclusions on the program (like the affinity of the various processes involved), but also to extract quantitative information (like communication matrices). A long term goal is to use the parallel loops to suggest program optimizations.

As of today, our algorithm has been evaluated on several applications. The most obvious is trace compression, with spectacular results because of the underlying loop-nest model (as was already the case for our sequential trace analysis algorithm). Another application is replay, where the program's (actual, i.e., traced) behavior can be simulated on a different parallel architecture. The last application is to build a lightweight model from a subset of trace data, and use the model to index into potentially massive quantitative data associated to the various events.

It turns out that it is difficult to publish such algorithms without evaluating them in “realistic” settings, on applications running on massively parallel hardware, something we don’t have easy access to. Also, there are currently a few algorithms that provide similar solutions to practitioners, in a way that we think are fundamentally inferior to our proposition but that seem to be good enough for their current use. Waiting for better opportunities to illustrate the power of our method, we have published a research report summarizing our work [26].

6.8. 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 is an ongoing work with the PhD student Lénaïc Bagnères (POSTALE Team at Inria Saclay-Île-de-France, co-advised by Christine Eisenbeis and Cédric Bastoul). The first results have been presented in 2014 at the Euro-Par International Conference [11].

6.9. 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, 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. The first results have been presented in 2014 to the IEEE VL/HCC Conference [22]. Moreover, another paper on the topic has been accepted to the International IMPACT 2015 Workshop to be held in conjunction with the HiPEAC International Conference.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

The CAMUS team is taking part of the NANO 2017 national research program and its sub-project PSAIC (Performance and Size Auto-tuning thru Iterative Compilation) with the company STMicroelectronics, starting January 2015. Luis Esteban Campostrini has been recruited as PhD student in this project. His work will focus on extending the Apollo framework to dynamic analysis providing useful feedbacks to users regarding code optimization opportunities, and to code generation for ARM Cortex platforms.

8. Partnerships and Cooperations

8.1. National Initiatives

Philippe Clauss, Alain Ketterlin, Cédric Bastoul and Vincent Loechner are involved in the Inria Project Lab entitled “Large scale multicore virtualization for performance scaling and portability” and regrouping several french researchers in compilers, parallel computing and program optimization [8]. 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.

8.2. European Initiatives

8.2.1. Collaborations in European Programs, except FP7 & H2020

Program: ITEA

Project acronym: MANY

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

Duration: 09/2011 - 12/2014

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

International Partner (Institution - Laboratory - Researcher):

Universidad de Buenos Aires (ARGENTINE)

Duration: 2011 - ___AT.ANNEEMOISFIN???___

See also: http://lafhis.dc.uba.ar/wiki/index.php/EA-Ancome

[8] https://team.inria.fr/multicore

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

8.4.1.1. Internships

Matías Hernando Pérez Matías

Date: May 2014 - Nov 2014

Institution: Universidad de Buenos Aires (Argentina)

Sabater César Rufino

Date: May 2014 - Oct 2014

Institution: Universidad Nacional de Rosario (Argentina)

Campostrini Luis Esteban

Date: Jul 2014 - Dec 2014

Institution: Universidad Nacional de Rosario (Argentina)

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific events organisation

9.1.1.1. member of the organizing committee

Cédric Bastoul has been co-organizing the HIP3ES 2015 workshop (High Performance Energy Efficient Embedded Systems) held in conjunction with the international conference HiPEAC 2015.

9.1.2. Scientific events selection

9.1.2.1. member of the conference program committee

Philippe Clauss, Cédric Bastoul and Vincent Loechner have been part of the program committee of IMPACT 2014 (International Workshop on Polyhedral Compilation Techniques), held in conjunction with the international conferences HiPEAC 2014. Philippe Clauss, and Vincent Loechner have been part of the program committee of IMPACT 2015, held in conjunction with HiPEAC 2015.

Vincent Loechner and Cédric Bastoul have been part of the program committee of HIP3ES 2015 workshop (High Performance Energy Efficient Embedded Systems), held in conjunction with the international conference HiPEAC 2015.

Cédric Bastoul has been part of the program committee of PARMA+DITAM 2015 (6th Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures 4th Workshop on Design Tools and Architerctures for Multicore Embedded Computing Platforms), held in conjunction with HiPEAC 2015.

9.1.2.2. reviewer

Vincent Loechner has been reviewer for the IMPACT 2015 workshop.

Philippe Clauss has been reviewer for the following conferences: ISMM '14 (International Symposium on Memory Management), IMPACT 2015.

Cédric Bastoul has been reviewer for the following conferences: CGO 2014 (International Conference on Code Generation and Optimization), PARMA 2014 and 2015 (International Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures), IMPACT 2014 (International Workshop on Polyhedral Compilation Techniques), HIP3ES 2015 (International Workshop on High Performance Energy Efficient Embedded Systems).

9.1.3. Journal

9.1.3.1. reviewer

Philippe Clauss has been reviewer for the following journals: IEEE Transactions on Computers, ACM Transactions on Programming Languages and Systems (TOPLAS), ACM Transactions on Architecture and Code Optimization (TACO), IEEE Transactions on Parallel and Distributed Systems (TPDS).

Cédric Bastoul has been reviewer for the following journals: Journal of Parallel and Distributed Computing (JPDC), ACM Transactions on Architecture and Code Optimization (TACO), Parallel Computing (ParCo).

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Philippe Clauss and Alain Ketterlin did not teach during the first semester of 2014 since they were detached at Inria (*délégation*).

Licence : Vincent Loechner, Operating Systems, 38h, L2, Strasbourg University, France

Master : Vincent Loechner, Python Programming, 38h, M1, Strasbourg University, France

Master : Vincent Loechner, Parallelism, 15h, M1, Strasbourg University, France

Master : Vincent Loechner, Parallel Programming, 32h, M1, Strasbourg University, France

Master : Vincent Loechner, Parallel Programming, 30h, M2, Strasbourg University, France

Master : Vincent Loechner, Advanced compilation, 8h, M1, Strasbourg University, France

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

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

Licence : Philippe Clauss, Computer Architecture, 45h, L2, Strasbourg University, France

Master : Philippe Clauss, Compilation, 78h, M1, Strasbourg University, France

Licence : Cédric Bastoul, System Programming, 49h, L2, Strasbourg University, France

Master : Cédric Bastoul, Compiler Design, 48h, M1, Strasbourg University, France

Master : Cédric Bastoul, Advanced compilation, 29h, M1, Strasbourg University, France

Master : Cédric Bastoul, Parallelism, 20h, M2, Strasbourg University, France

Master : Cédric Bastoul, Introduction to Research, 7h, M1, Strasbourg University, France

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

Licence : Éric Violard, Computer Architectures, 23h, L2, Strasbourg University, France

Licence : Éric Violard, Algorithms and Data Structures, 34h, L2, Strasbourg University, France

Master : Éric Violard, Compiler Design, 54h, M1, Strasbourg University, France

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

Master : Éric Violard, Introduction to Research, 6h, M1, Strasbourg University, France

Licence : Alain Ketterlin, Computer Networks, 66h, L3, Strasbourg University, France

Licence : Alain Ketterlin, Algorithms and Data Structures, 40h, L3, 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

PhD in progress : Yann Barsamian, A Parallel Programming Environment based on the Xfor Control Structure, October 2014, Éric Violard

9.2.3. Juries

Philippe Clauss participated to the following PhD jurys in 2014:

Date	**Candidate**	**Place**	**Role**
Sept. 22	Michael KRUSE	Univ. Paris Sud	Reviewer
June 20	Pierre ESTERIE	Univ. Paris Sud	Reviewer

Cédric Bastoul participated to the following PhD jurys in 2014:

Date	**Candidate**	**Place**	**Role**
November 19	Rachid Habel	École des Mines de Paris	Reviewer

9.3. Popularization

Cédric Bastoul participated to the event *Kids University* at the University of Strasbourg in November 2014

Cédric Bastoul prepared activities for *Fête de la Science* at University of Paris-Sud in October 2014

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, 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^o 8, pp. 983-996

[4] 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", Boston, USA, ACM, April 2008, pp. 94-103, http://dx.doi.org/10.1145/1356058.1356071

[5] A. Ketterlin, P. Clauss. *Profiling Data-Dependence to Assist Parallelization: Framework, Scope, and Optimization*, in "MICRO-45 – Proceedings of the 2012 IEEE/ACM 45th International Symposium on Microarchitecture", Vancouver, Canada, December 2012

[6] B. Pradelle, A. Ketterlin, P. Clauss. *Polyhedral parallelization of binary code*, in "ACM Transactions on Architecture and Code Optimization", January 2012, vol. 8, n^o 4, pp. 39:1–39:21 [*DOI :* 10.1145/2086696.2086718], http://hal.inria.fr/hal-00664370

[7] 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

[8] 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^o 1, pp. 37–66, http://dx.doi.org/10.1007/s00453-006-1231-0

Publications of the year

Articles in International Peer-Reviewed Journals

[9] A. Jimborean, P. Clauss, J.-F. Dollinger, V. Loechner, M. Juan Manuel. *Dynamic and Speculative Polyhedral Parallelization Using Compiler-Generated Skeletons*, in "International Journal of Parallel Programming", August 2014, vol. 42, n^o 4, pp. 529-545, https://hal.inria.fr/hal-01003744

[10] 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], https://hal.inria.fr/hal-00909961

International Conferences with Proceedings

[11] L. BAGNÈRES, C. BASTOUL. *Switchable Scheduling for Runtime Adaptation of Optimization*, in "Euro-Par 2014 Parallel Processing", Porto, Portugal, Lecture Notes in Computer Science, Springer International Publishing, August 2014, vol. 8632, pp. 222 - 233 [*DOI :* 10.1007/978-3-319-09873-9_19], https://hal.inria.fr/hal-01097200

[12] P. CLAUSS. *Author Retrospective for Counting Solutions to Linear and Nonlinear Constraints Through Ehrhart Polynomials: Applications to Analyze and Transform Scientific Programs*, in "ICS, International Conference on Supercomputing", New York, United States, ACM ICS 25th Anniversary Volume, ACM, 2014 [*DOI :* 10.1145/2591635.2591654], https://hal.inria.fr/hal-01100296

[13] P. CLAUSS. *Counting Solutions to Linear and Nonlinear Constraints Through Ehrhart Polynomials: Applications to Analyze and Transform Scientific Programs*, in "ICS, International Conference on Supercomputing", Munich, Germany, ACM ICS 25th Anniversary Volume, 2014 [*DOI :* 10.1145/2591635.2667172], https://hal.inria.fr/hal-01100306

[14] P. CLAUSS. *Mind The Gap! A study of some pitfalls preventing peak performance in polyhedral compilation using a polyhedral antidote*, in "IMPACT - Fifth International Workshop on Polyhedral Compilation Techniques, In conjunction with HiPEAC", Amsterdam, Netherlands, January 2015, https://hal.inria.fr/hal-01099583

[15] P. CLAUSS, I. FASSI, A. JIMBOREAN. *Software-controlled Processor Stalls for Time and Energy Efficient Data Locality Optimization*, in "International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation - SAMOS XIV", Agios Konstantinos, Greece, July 2014, https://hal.inria.fr/hal-01003228

[16] P. FEAUTRIER, E. VIOLARD, A. KETTERLIN. *Improving X10 Program Performances by Clock Removal*, in "CC'14 - 23rd International Conference on Compiler Construction, part of ETAPS'14", Grenoble, France, April 2014, https://hal.inria.fr/hal-00924206

[17] B. NARASIMHA SWAMY, A. KETTERLIN, A. SEZNEC. *Hardware/Software Helper Thread Prefetching On Heterogeneous Many Cores*, in "2014 IEEE 26th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD)", Paris, France, October 2014 [*DOI :* 10.1109/SBAC-PAD.2014.39], https://hal.inria.fr/hal-01087752

[18] 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, https://hal.inria.fr/hal-00917950

[19] S. STOJANOVIC, J. NARBOUX, M. BEZEM, P. JANICIC. *A Vernacular for Coherent Logic*, in "CICM 2014 - Conferences on Intelligent Computer Mathematics", Coimbra, Portugal, Lecture Notes in Computer Science, Springer, July 2014, vol. 8543, https://hal.inria.fr/hal-00983975

[20] A. SUKUMARAN-RAJAM, J. M. MARTINEZ, W. WOLFF, A. JIMBOREAN, P. CLAUSS. *Speculative Program Parallelization with Scalable and Decentralized Runtime Verification*, in "Runtime Verification", Toronto, Canada, B. BONAKDARPOUR, S. A. SMOLKA (editors), Springer, September 2014, vol. 8734, pp. 124-139 [*DOI :* 10.1007/978-3-319-11164-3_11], https://hal.inria.fr/hal-01070610

[21] O. ZINENKO, C. BASTOUL, S. HUOT. *Manipulating Visualization, Not Codes*, in "International Workshop on Polyhedral Compilation Techniques (IMPACT)", Amsterdam, Netherlands, January 2015, 8 p. , https://hal.inria.fr/hal-01100974

[22] O. ZINENKO, S. HUOT, C. BASTOUL. *Clint: A Direct Manipulation Tool for Parallelizing Compute-Intensive Program Parts*, in "IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)", Melbourne, Australia, IEEE, July 2014, https://hal.inria.fr/hal-01055788

Conferences without Proceedings

[23] P. BOUTRY, J. NARBOUX, P. SCHRECK, G. BRAUN. *A short note about case distinctions in Tarski's geometry*, in "Automated Deduction in Geometry 2014", Coimbra, Portugal, F. BOTANA, P. QUARESMA (editors), Proceedings of ADG 2014, July 2014, pp. 1-15, https://hal.inria.fr/hal-00989785

[24] P. BOUTRY, J. NARBOUX, P. SCHRECK, G. BRAUN. *Using small scale automation to improve both accessibility and readability of formal proofs in geometry*, in "Automated Deduction in Geometry 2014", Coimbra, Portugal, F. BOTANA, P. QUARESMA (editors), Proceedings of ADG 2014, July 2014, pp. 1-19, https://hal.inria.fr/hal-00989781

[25] J.-F. DOLLINGER, V. LOECHNER. *CPU+GPU Load Balance Guided by Execution Time Prediction*, in "Fifth International Workshop on Polyhedral Compilation Techniques (IMPACT 2015)", Amsterdam, Netherlands, January 2015, https://hal.inria.fr/hal-01095890

Research Reports

[26] A. KETTERLIN, M. KUHN, S. GENAUD, P. CLAUSS. *Loop-based Modeling of Parallel Communication Traces*, July 2014, n^o RR-8562, 10 p. , https://hal.inria.fr/hal-01044636

References in notes

[27] 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, https://hal.archives-ouvertes.fr/ccsd-00017260

[28] 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

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

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