Activity Report 2014

Team GCG

Grenoble's Compiler Group

RESEARCH CENTER
Grenoble - Rhône-Alpes

THEME
Architecture, Languages and Compilation

Table of contents

Team GCG

Keywords: Compilation, Static Analysis, Optimization, Energy Consumption, Performance, Graph Theory, Scheduling, Memory

Creation of the Team: 2014 January 01, end of the Team: 2014 October 31.

1. Members

Research Scientist
Fabrice Rastello [Team leader, Inria, Researcher, HdR]

Engineers
Lukasz Domagala [Inria]
Fabian Gruber [Inria, from Oct 2014]
Alexandros Lamprineas [Inria]

PhD Students
François Gindraud [Univ. Grenoble I, Allocation Normalien]
Diogo Nunes Sampaio [Univ. Grenoble I, Bourse Brésil]
Duco Van Amstel [Kalray, CIFRE]

Visiting Scientist
Fernando Magno Quintao Pereira [Inria, UFMG, Prof]

Administrative Assistant
Maria Immaculada Presseguer [Inria]

2. Overall Objectives

2.1. Overall Objectives

Languages, compilers, and run-time systems are some of the most important components to bridge the gap between applications and hardware. With the continuously increasing power of computers, expectations are evolving, with more and more ambitious, computational intensive and complex appli- cations. As desktop PCs are becoming a niche and servers mainstream, three categories of computing impose themselves for the next decade: mobile, cloud, and super-computing. Thus diversity, heterogeneity (even on a single chip) and thus also hardware virtualization is putting more and more pressure on compilers. However, because of the energy wall, architectures are becoming more and more complex and parallelism ubiquitous at every level. Unfortunately, the memory-CPU gap continues to increase and energy consumption remains an important issue for future platforms. To address the challenge of performance and energy consumption raised by silicon companies, compilers must evolve and, in particular, interact, taking into account the complexity of the host architecture. The overall objective of GCG is to address this challenge by combining static and dynamic compilation techniques.

3. Research Program

3.1. Foundations

It has been ten years now since Intel bumped on the energy wall. Parallelism is now ubiquitous, not only restricted to expensive servers dedicated to some regular scientific computation. Also, the panel of possible mainstream architectures became extremely diverse. The use of byte-codes (e.g. nVIDIA PTX) along with Just-In-Time (JIT) compilation allowed fast evolution of designs. Quite recently, silicon companies understood that this heterogeneity should be integrated into the same chip (e.g. ARM big.LITTLE, nVIDIA Tegra K1); also re-configurable architectures (from FPGA to CGRA) are becoming present in such design as specialization is clearly useful to increase performance with less increase in energy consumption. Even cache-size, crossbar will be dynamically re-configurable; distributed DVFS being now mainstream... Postponing the decision of where and how (depending on the context) to execute part of an application, involves the use of late/adaptive compilation so as to avoid code size blowing. This observation is amplified by the fact that application behavior gets more and more dominated by data-characteristics. This is precisely what motivated more than fifteen years ago the development of dynamic compiler optimization technology. Many transformations, decisions, code-generation phases done by a compiler are now critically required to be postponed at run-time when the information is becoming available. But, this is not to mention the need of auto-tuning and adaptive compilation that imposes itself to address the increasing complexity (and hard to model) of each individual core.

The research direction of GCG is motivated by the perspective of optimizing (sometimes complex and irregular) micro-kernels for a single core (SIMD/VLIW). It starts from the observation that despite the clear motivation for JIT/dynamic compilation, despite its clear maturity, we lost the battle of performance portability: such technologies are not as optimizing as we pretended it would be. The reason for this defeat is that there is no perfect place to analyze, optimize, transform. On one hand " JIT-ing" source-level code would usually be too slow, while on the other hand byte-code close to machine-level lost high-level semantics. Apart from spending its time to retrieve somehow obvious information, the JIT-compiler has to deal with limited resources, with realistic time constraints. Thus the need for being hybrid, in other words combine static and dynamic compilation/analysis techniques using rich intermediate languages.

Hybrid compilation consists in combining in any possible ways static analysis with profiling and run-time tests, but also ahead-of-time with run-time code optimization. This leads GCG to put efforts on researches on hybrid compilation frameworks but also on compiler architecture design. This last is to address the difficult problem of information telescoping (maintain of information of different type) and the problem of code size.

Current projects include:

- characterization of applications (I/O complexity) and profiling feedback using trace analyses;
- combined scheduling and memory allocation for irregular applications;
- extension of the polyhedral model using hybrid analysis and compilation;
- design, promotion and development of an hybrid and extensible byte-code, Tirex;
- design of a run-time handling communications, scheduling and placement for distributed memory parallel architectures.

4. Application Domains

4.1. Transfer

The main industrial sector related to the research activities of GCG is the one of semi-conductor (programmable architectures spanning from embedded systems to servers). Obviously any computing application which has the objective of exploiting as much as possible the resources (in terms of high-performance but also low energy consumption) of the host architecture is intended to take advantage of advances in compiler and runtime technology. These applications are based over numerical kernels (linear algebra, FFT, convolution...) that can be adapted on a large spectrum of architectures. Members of GCG already maintain fruitful and strong collaborations with several companies such as STMicroelectronics, Kalray, plus a recent (not yet formal) collaboration with Intel.

5. New Software and Platforms

5.1. Givy

Givy is a runtime currently developed as part of the phd thesis of François Gindraud. It is designed for architectures with distributed memories, with the Kalray MPPA as the main target. It will execute dynamic data-flow task graphs, annotated with memory dependencies. It will automatically handle scheduling and placement of tasks (using the memory dependency hints), and generate memory transfers between distributed memory nodes when needed by using a software cache coherence protocol. Most of the work this year was done on implementing and testing a memory allocator with specific properties that is a building block of the whole runtime. This memory allocator is also tuned to work on the MPPA and its constraints, turning with very little memory and being efficient in the context of multithreaded calls.

5.2. Tirex

The Tirex Intermediate Representation has previously been generated from within both the Path64 and GCC compilers. In order to increase the usability of Tirex and to decrease the amount of required code maintenance that is induced by compiler evolutions a Tirex-generator has been written that is capable of creating the Tirex representation of a program based on its corresponding assembler code.

5.3. LLVM plugins

Work has been started on multiple plugins for the LLVM compiler framework that implement the code optimisations that have been elaborated by the team. While being work in progress this already provides us with crucial information for program analysis such as data-dependencies.

6. New Results

6.1. Highlights of the Year

Graduate Research Award of the OSU department in 2015 for Venmugil Elango (co-advised by Fabrice Rastello)

6.2. An interval constrained memory allocator for a GAS runtime

Participants: François Gindraud, Fabrice Rastello, Albert Cohen [ENS Ulm].

This work presents a memory allocator for global address space (GAS) runtime targeting distributed memory embedded architectures (MPSoC). MPSoC we are interested in are relatively new architectures, composed of several nodes with multiple general purpose cores and a local memory, linked by a network, all on one chip (NoC). They have promising energy and computing performances, but are hard to program due to the multilevel parallelism and the hardware constraints (limited memory, network structure). Existing programming framework are either thin but let the programmer do the hard choices (OpenMP + MPI) or heavy and automatic but target specific kind of applications on big systems (Global Arrays).

Givy 5.1 is a runtime currently developed to execute dynamic task graphs with data-flow dependencies on MPSoC. It has a focus on supporting irregular applications, using the dependencies to perform data-aware dynamic task scheduling and data transfer. Data blocks live in a GAS, and thus requires a GAS-aware memory allocator to avoid address collisions when they are dynamically allocated. The allocator implementation proposed in this paper does this with zero synchronization between nodes, while being memory efficient in the small distributed memories, and fast on each multithreaded node.

This work will be submitted at ACM ISMM Symposium.

6.3. A Framework for Enhancing Data Reuse via Associative Reordering

Participants: Kevin Stock [OSU], Martin Kong [OSU], Tobias Grosser [ENS Ulm], Louis-Noël Pouchet [UCLA], Fabrice Rastello, J. Ramanujam [LSU], P. Sadayappan [OSU].

The freedom to reorder computations involving associative operators has been widely recognized and exploited in designing parallel algorithms and to a more limited extent in optimizing compilers.

In this work, we develop a novel framework utilizing the associativity and commutativity of operations in regular loop computations to enhance register reuse. Stencils represent a particular class of important computations where the optimization framework can be applied to enhance performance. We show how stencil operations can be implemented to better exploit register reuse and reduce load/stores. We develop a multi-dimensional retiming formalism to characterize the space of valid implementations in conjunction with other program transformations. Experimental results demonstrate the effectiveness of the framework on a collection of high-order stencils.

This work is the fruit of the collaboration 8.1 with OSU and has been presented at the conference ACM PLDI'14.

6.4. Beyond Reuse Distance Analysis: Dynamic Analysis for Characterization of Data Locality Potential

Participants: Naznin Fauzia [OSU], Venmugil Elango [OSU], Mahesh Ravishankar [OSU], J. Ramanujam [LSU], Fabrice Rastello, Atanas Routnev [OSU], Louis-Noël Pouchet [UCLA], P. Sadayappan [OSU].

Emerging computer architectures will feature drastically decreased flops/byte (ratio of peak processing rate to memory bandwidth) as highlighted by recent studies on Exascale architectural trends. Further, flops are getting cheaper while the energy cost of data movement is increasingly dominant. The understanding and characterization of data locality properties of computations is critical in order to guide efforts to enhance data locality.

Reuse distance analysis of memory address traces is a valuable tool to perform data locality characterization of programs. A single reuse distance analysis can be used to estimate the number of cache misses in a fully associative LRU cache of any size, thereby providing estimates on the minimum bandwidth requirements at different levels of the memory hierarchy to avoid being bandwidth bound. However, such an analysis only holds for the particular execution order that produced the trace. It cannot estimate potential improvement in data locality through dependence preserving transformations that change the execution schedule of the operations in the computation.

In this work, we develop a novel dynamic analysis approach to characterize the inherent locality properties of a computation and thereby assess the potential for data locality enhancement via dependence preserving transformations.

This work is the fruit of the collaboration 8.1 with OSU and has been published at ACM TACO'14.

6.5. On Using the Roofline Model with Lower Bounds on Data Movement

Participants: Venmugil Elango [OSU], Naser Sedaghati [OSU], Fabrice Rastello, Louis-Noël Pouchet [UCLA], J. Ramanujam [LSU], Radu Teodorescu [OSU], P. Sadayappan [OSU].

The roofline model is a popular approach to “bounds and bottleneck” performance analysis. It focuses on the limits to performance of processors because of limited bandwidth to off-chip memory. It models upper bounds on performance as a function of operational intensity, the ratio of computational operations per byte of data moved from/to memory. While operational intensity can be directly measured for a specific implementation of an algorithm on a particular target platform, it is of interest to obtain broader insights on bottlenecks, where various semantically equivalent implementations of an algorithm are considered, along with analysis for variations in architectural parameters. This is currently very cumbersome and requires performance modeling and analysis of many variants.

In this work, we alleviate this problem by using the roofline model in conjunction with upper bounds on the operational intensity of computations as a function of cache capacity, derived using lower bounds on data movement. This enables bottleneck analysis that holds across all dependence-preserving semantically equivalent implementations of an algorithm. We demonstrate the utility of the approach in in assessing fundamental limits to performance and energy efficiency for several benchmark algorithms across a design space of architectural variations.

This work is the fruit of the collaboration 8.1 with OSU and is to be published at ACM TACO’15.

6.6. On Characterizing the Data Access Complexity of Programs

Participants: Venmugil Elango [OSU], Fabrice Rastello, Louis-Noël Pouchet [UCLA], J. Ramanujam [LSU], P. Sadayappan [OSU].

Technology trends will cause data movement to account for the majority of energy expenditure and execution time on emerging computers. Therefore, computational complexity will no longer be a sufficient metric for comparing algorithms, and a fundamental characterization of data access complexity will be increasingly important. The problem of developing lower bounds for data access complexity has been modeled using the formalism of Hong & Kung’s red/blue pebble game for computational directed acyclic graphs (CDAGs). However, previously developed approaches to lower bounds analysis for the red/blue pebble game are very limited in effectiveness when applied to CDAGs of real programs, with computations comprised of multiple sub-computations with differing DAG structure. We address this problem by developing an approach for effectively composing lower bounds based on graph decomposition. We also develop a static analysis algorithm to derive the asymptotic data-access lower bounds of programs, as a function of the problem size and cache size.

This work is the fruit of the collaboration 8.1 with OSU and is to be presented at ACM POPL’15.

6.7. PolyCheck: Dynamic Verification of Iteration Space Transformations on Affine Programs

Participants: Sriram Krishnamoorthy [PNNL], Bao Wenlei [OSU], Louis-Noël Pouchet [UCLA], P. Sadayappan [OSU], Fabrice Rastello.

High-level compiler transformations, especially loop transformations, are widely recognized as critical optimizations to restructure programs to improve data locality and expose parallelism.

Guaranteeing the correctness of program transformations is essential, and to date three main approaches have been developed: proof of equivalence of affine programs, matching the execution traces of programs, and checking bit-by-bit equivalence of the outputs of the programs. Each technique suffers from limitations in either the kind of transformations supported, space complexity, or the sensitivity to the testing dataset. In this paper, we take a novel approach addressing all three limitations to provide an automatic bug checker to verify any iteration reordering transformations on affine programs, including non-affine transformations, with space consumption proportional to the original program data, and robust to arbitrary datasets of a given size. We achieve this by exploiting the structure of affine program control- and data-flow to generate at compile-time a lightweight checker code to be executed within the transformed program. Experimental results assess the correctness and effectiveness of our method, and its increased coverage over previous approaches.

This work is the result of the collaboration 8.1 with OSU.

6.8. On Using Lower Bounds for Discrimination of Utility/Futility of Loop Fusion

Participants: Samyam Rajbhandari [OSU], Martin Konk [OSU], P. Sadayappan [OSU], Robert J. Harrison [Stonybrook], Fabrice Rastello.

Fusion is an important loop transformation for data locality enhancement. However, it is very challenging to determine which of a set of possible fusion choices is best. In this paper, we pursue a novel approach to addressing this problem. Instead of the conventional approach of explicitly modeling different possible fused loop configurations and modeling the expected performance with each, we instead use lower bounds modeling to characterize conditions where fusion might have utility and where it will be futile because the maximal possible improvement from fusion is much lower than the minimal data movement overheads for each of the unfused components. We successfully demonstrate the use of such a methodology with two practically important codes from the quantum chemistry domain, i) with the affine 4-index transform code, and ii) unstructured tree operations with the MADNESS framework.

This work is the result of the collaboration 8.1 with OSU.

6.9. A Tiling Perspective for Register Optimization

Participants: Duco Van Amstel, Lukasz Domagala, P. Sadayappan [OSU], Fabrice Rastello.

Register allocation is a much studied problem. A particularly important context for optimizing register allocation is within loops, since a significant fraction of the execution time of programs is often inside loop code. A variety of algorithms have been proposed in the past for register allocation, but the complexity of the problem has resulted in a decoupling of several important aspects, including loop unrolling, register promotion, and instruction reordering.

In this work, we develop an approach to register allocation and promotion in a unified optimization framework that simultaneously considers the impact of loop unrolling and instruction scheduling. This is done via a novel instruction tiling approach where instructions within a loop are represented along one dimension and innermost loop iterations along the other dimension. By exploiting the regularity along the loop dimension, and imposing essential dependence based constraints on intra-tile execution order, the problem of optimizing register pressure is cast in a constraint programming formalism. Experimental results are provided from thousands of innermost loops extracted from the SPEC benchmarks, demonstrating improvements over the current state-of-the-art.

This work is the fruit of both the collaboration 8.1 with OSU and with Kalray 7.1 7.2.

6.10. Hybrid Pointer Disambiguation

Participants: Fernando Pereira, Alexandros Labrineas, Péricles Alves, Fabian Gruber, Fabrice Rastello.

In order to provide effective optimizations, compilers must deal with memory dependences. However, the state-of-the-art heuristics available in the literature to track memory dependencies are inherently imprecise and computationally expensive. Consequently, the most advanced code transformations that compilers have today are ineffective when applied on real-world programs. The goal of this paper is to solve this conundrum - a goal that we accomplish through the hybrid disambiguation of pointers. We provide a static analysis that generates dynamic tests to determine when two memory locations can overlap. We then produce two versions of a loop: one that is aliasing-free - hence, easy to optimize - and another that is not. Our checks lets us safely branch to the optimizable region. We have applied these ideas on Polly-LLVM, a loop optimizer built on top of the LLVM compilation infrastructure. Our experiments indicate that our method is precise, effective and useful: we can disambiguate the vast majority of checks in benchmarks that go from the loop intensive Polybench suite to the more general SPEC CPU 2006 benchmark collection. The result of this precision is code quality: the binaries that we generate are 9.5% faster than those that Polly-LLVM produces without our optimization. Given the current technology to statically solve alias analysis, we believe that our ideas are a necessary step to make modern compiler optimizations useful in practice.

This work is the fruit of the collaboration with UFMG 8.1 and Kalray 7.1 7.2.

6.11. Parameterized Construction of Program Representations for Sparse Dataflow Analyses

Participants: André Tavares [ENS Lyon], Benoit Boissinot [ENS Lyon], Fernando Pereira, Fabrice Rastello.

Data-flow analyses usually associate information with control flow regions. Informally, if these regions are too small, like a point between two consecutive statements, we call the analysis dense. On the other hand, if these regions include many such points, then we call it sparse. This paper presents a systematic method to build program representations that support sparse analyses. To pave the way to this framework we clarify the bibliography about well-known intermediate program representations. We show that our approach, up to parameter choice, subsumes many of these representations, such as the SSA, SSI and e-SSA forms. In particular, our algorithms are faster, simpler and more frugal than the previous techniques used to construct SSI - Static Single Information - form programs. We produce intermediate representations isomorphic to Choi *et al.*'s Sparse Evaluation Graphs (SEG) for the family of data-flow problems that can be partitioned per variables. However, contrary to SEGs, we can handle - sparsely - problems that are not in this family. We have tested our ideas in the LLVM compiler, comparing different program representations in terms of size and construction time.

This work is the fruit of the collaboration with UFMG 8.1 and has been presented at Springer CC'14.

6.12. Time-critical Computing on a Single-chip Massively Parallel Processor

Participants: Benoit Dupont-de-Dinechin [Kalray], Duco Van Amstel, Marc Poulhiès [Kalray], Guillaume Lager [Kalray].

In this work we demonstrate the capabilities of the MPPA(TM)-256 chip in the field of time-critical computations. This manycore chip features amongst others a Network-on-Chip (NoC) linking the seperate computational clusters each disposing of its own local memory and processing elements (PEs). The PEs architectural features induce a locally deterministic behaviour and the memory access arbitration that is used allows for a Worst-Case Execution Time (WCET) that is achieved for the combination of all local worst-cases. As such, in order to achieve a WCET analysis for a full MPPA(TM)-256 chip, we provide a Worst-Case Traversal Time (WCTT) analysis for the NoC to link the WCETs provided by each computational cluster. This part of the work is based on the (sigma, rho) model used for general network flow analysis and Quality-of-Service (QoS) parametrization.

This work has been presented at DATE'14.

6.13. Guaranteed Services of the NoC of a Manycore Processor

Participants: Benoit Dupont-de-Dinechin [Kalray], Yves Durand [CEA], Duco Van Amstel [Kalray], Alexandre Ghiti [Kalray].

In the case of the MPPA(TM)-256 chip the study of the integrated Network-on-Chip (NoC) is a fundamental subject for anyone using this architecture for time-critical purposes or real-time use-cases that need guarantees on the Worst-Case Traversal Time (WCTT) of the NoC. Previous work has already shown that the MPPA(TM)-256 NoC can be modelled using the (sigma, rho)-model. In the current work we will elaborate on this point by providing an indepth analysis of the NoC as well as the method to guarantee Quality-of-Service properties.

This work has been presented at the International Workshop on Network on Chip Architectures 2014.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

- Tirex is a bilateral contract with Kalray. The subject is a prototyping of hybrid alias analysis. The collaboration lead to a recent submission which corresponding work is described in 6.10.
- GCG is involved in another contract with Kalray associated with the CIFRE PhD of Duco van Amstel. The subject of the collaboration is related to fine grain scheduling. Corresponding work is described in 6.9.

7.2. Bilateral Grants with Industry

- ManyCoreLabs is a bilateral Grant (BGLE) with Kalray. GCG is involved in the development of generalized register tiling.

8. Partnerships and Cooperations

8.1. International Initiatives

8.1.1. Inria International Partners

8.1.1.1. Informal International Partners

- P. Sadayappan, OSU, Columbus, Ohio, USA: Collaboration on automatic analysis of I/O complexity (several co-publications); collaboration on code optimization (one join paper + one submitted paper)
- Fernando Pereira, UFMG, Bello Horizonte, Brazil: Collaboration on static analysis (on join paper); collaboration on hybrid analysis (one submitted paper)

8.2. International Research Visitors

8.2.1. Visits of International Scientists

- Prof. Fernando Magno Pereira, 1 months 1/2, UFMG Brazil

8.2.2. Visits to International Teams

8.2.2.1. Research stays abroad

- Fabrice Rastello: 2 months at OSU, Columbus, Ohio with the team of P. Sadayappan.

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific events organisation

9.1.1.1. Member of the organizing committee

- Fabrice Rastello: Publication chair ACM/IEEE CGO 2015
- Fabrice Rastello: Steering committee Journées française de la compilation

9.1.2. Scientific events selection

9.1.2.1. Member of the conference program committee

- Fabrice Rastello: ACM/IEEE CGO 2014, ACM/IEEE CGO 2015

9.1.2.2. Reviewer

- Fabrice Rastello: Springer CC 2014

9.1.3. Journal

9.1.3.1. Reviewer

- Fabrice Rastello: ACM TACO, Elsevier ParCo

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Master : Fabrice Rastello (external intervention), Programming Languages and Compiler Design, 4.5 hours, M1 of international MoSIG, UJF, Grenoble, France

Master : Fabrice Rastello (with Christophe Alias and Florent de Dinechin), Advanced Compilation, 15 hours, M1/M2, ENS Lyon, France

Post-doctoral : Fabrice Rastello, SSA based compiler design, 6 hours, CRI, Yaounde Cameroun

9.2.2. Supervision

- PhD in progress: Duco van Amstel, Scheduling and optimization for memory locality of dataflow programs on many-core processors, April 1st 2013, advised by Fabrice Rastello and Benoit Dupont-de-Dinechin
- PhD in progress: Diogo Sampaio, Profiling Guided Hybrid Compilation, October 8 2013, advised by Fabrice Rastello
- PhD in progress: François Gindraud, Semantics and compilation for a data-flow model with a global address space and software cache coherency, January 1st 2013, advised by Fabrice Rastello and Albert Cohen.
- PhD in progress: Venmugil Elango, Dynamic Analysis for Characterization of Data Locality Potential, September 2010, advised by Fabrice Rastello and P. Sadayappan.

9.2.3. Juries

- Feng Li, Reviewer, 20th of Mai, Compiling for a multithreaded dataflow architecture: algorithms, tools, and experience, Doctorat de l'UPMC EDITE
- Kevin Pouget, Jury, 3rd of February, Programming-Model Centric Debugging for Multicore Embedded Systems, MSTII, Grenoble

10. Bibliography

Publications of the year

Articles in International Peer-Reviewed Journals

[1] V. Elango, N. Sedaghati, F. Rastello, L.-N. Pouchet, J. Ramanujam, r. Teodorescu, P. Sadayappan. *On Using the Roofline Model with Lower Bounds on Data Movement*, in "ACM Transactions on Architecture and Code Optimization (TACO) ", January 2015, vol. 11, n^o 4, pp. 67:1–67:23, https://hal.inria.fr/hal-01104765

International Conferences with Proceedings

[2] B. Dupont De Dinechin, D. van Amstel, M. Poulhies, G. Lager. *Time-critical computing on a single-chip massively parallel processor*, in "Conference on Design, Automation & Test in Europe", Dresden, Germany, European Design and Automation Association (editor), Proceedings of the Conference on Design, Automation & Test in Europe 2014, March 2014, pp. 97:1-97:6, https://hal.inria.fr/hal-01090449

[3] B. Dupont De Dinechin, D. Yves, D. van Amstel, A. Ghiti. *Guaranteed Services of the NoC of a Manycore Processor*, in "International Workshop on Network-on-Chips", Cambridge, United Kingdom, Proceedings of the International Workshop on Network-on-Chips 2014, December 2014, 6 p. , https://hal.inria.fr/hal-01102657

[4] V. Elango, F. Rastello, L.-N. Pouchet, J. Ramanujam, P. Sadayappan. *On Characterizing the Data Movement Complexity of Computational DAGs for Parallel Execution*, in "Symposium on Parallelism in Algorithms and Architectures (SPAA '14)", Prague, Poland, ACM, 2014, https://hal.inria.fr/hal-01016090

[5] V. Elango, F. Rastello, L.-N. Pouchet, J. Ramanujam, P. Sadayappan. *On Characterizing the Data Access Complexity of Programs*, in "42nd Annual ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL", Mumbai, India, ACM, January 2015, pp. 567-580, https://hal.inria.fr/hal-01104556

[6] K. Stock, M. Kong, T. Grosser, L.-N. Pouchet, F. Rastello, J. Ramanujam, P. Sadayappan. *A Framework for Enhancing Data Reuse via Associative Reordering*, in "PLDI '14 - 35th ACM SIGPLAN Conference on Programming Language Design and Implementation", Edinburgh, United Kingdom, ACM, June 2014, pp. 65-76 [*DOI :* 10.1145/2594291.2594342], https://hal.inria.fr/hal-01016093

[7] A. Tavares, F. Rastello, B. Boissinot, F. Pereira. *Parameterized Construction of Program Representations for Sparse Dataflow Analyses*, in "CC 2014 - 23rd International Conference on Compiler Construction", Grenoble, France, Springer, 2014, https://hal.inria.fr/hal-00921461

Research Reports

[8] L. Domagala, F. Rastello, S. Ponnuswany, D. Van Amstel. *A Tiling Perspective for Register Optimization*, May 2014, n^{o} RR-8541, 24 p. , https://hal.inria.fr/hal-00998915

[9] V. Elango, F. Rastello, L.-N. Pouchet, J. Ramanujam, P. Sadayappan. *On Characterizing the Data Movement Complexity of Computational DAGs for Parallel Execution*, April 2014, n^{o} RR-8522, 27 p. , https://hal.inria.fr/hal-00980580

[10] A. Tavares, B. Boissinot, F. Pereira, F. Rastello. *Parameterized Construction of Program Representations for Sparse Dataflow Analyses*, Inria, March 2014, n^{o} RR-8491, 27 p. , https://hal.inria.fr/hal-00963590