RESEARCH CENTRE

Grenoble - Rhône-Alpes

IN PARTNERSHIP WITH:

Institut polytechnique de Grenoble,
Université de Grenoble Alpes

2021
ACTIVITY REPORT

Project-Team
CORSE

compiler optimization and run-time systems

IN COLLABORATION WITH: Laboratoire d'Informatique de Grenoble (LIG)

DOMAIN

Algorithmics, Programming, Software and Architecture

THEME

Architecture, Languages and Compilation

Contents

Project-Team CORSE

Creation of the Project-Team: 2016 July 01

Keywords

Computer sciences and digital sciences

A1.1.1. – Multicore, Manycore

A1.1.2. – Hardware accelerators (GPGPU, FPGA, etc.)

A1.1.3. – Memory models

A1.1.4. – High performance computing

A1.1.12. – Non-conventional architectures

A1.6. – Green Computing

A2.1.6. – Concurrent programming

A2.1.7. – Distributed programming

A2.1.8. – Aspect-oriented programming

A2.1.10. – Domain-specific languages

A2.2. – Compilation

A2.2.1. – Static analysis

A2.2.2. – Memory models

A2.2.3. – Memory management

A2.2.4. – Parallel architectures

A2.2.5. – Run-time systems

A2.2.6. – GPGPU, FPGA...

A2.2.8. – Code generation

A2.2.9. – Security by compilation

A2.3.2. – Cyber-physical systems

A4.4. – Security of equipment and software

A7.1. – Algorithms

Other research topics and application domains

B4.5. – Energy consumption

B5.3. – Nanotechnology

B6.1.2. – Software evolution, maintenance

B6.6. – Embedded systems

B6.7. – Computer Industry (harware, equipments...)

B9.1. – Education

1 Team members, visitors, external collaborators

Research Scientists

- Fabrice Rastello [Team leader, Inria, Senior Researcher, HDR]
- Guillaume Iooss [Inria, Starting Research Position]

Faculty Members

- Florent Bouchez - Tichadou [Univ Grenoble Alpes, Associate Professor]
- Francois Broquedis [Institut polytechnique de Grenoble, Associate Professor]
- Ylies Falcone [Univ Grenoble Alpes, Associate Professor]
- Manuel Selva [Institut polytechnique de Grenoble, Associate Professor]

Post-Doctoral Fellows

- Hugo Brunie [Inria, from Apr 2021]
- Victor Roussanaly [Inria, from Jul 2021]

PhD Students

- Theo Barollet [Inria]
- Theophile Bastian [Univ Grenoble Alpes, from Sep 2021]
- Nicolas Derumigny [Univ Grenoble Alpes]
- Nihel Kaboubi [Orange Labs, CIFRE]
- Marius Monnier [Univ Grenoble Alpes, from Oct 2021]
- Auguste Olivry [Univ Grenoble Alpes]
- Chukri Soueidi [Inria]
- Mathieu Stoffel [Bull, Jan 2021]
- Nicolas Tollenaere [Inria]

Technical Staff

- Theophile Bastian [Inria, Engineer, until Feb 2021]
- Christophe Guillon [Inria, Engineer, from Nov 2021]
- Stephane Pouget [Inria, Engineer, from Feb 2021 until Aug 2021]
- Mariana Vargas Vieyra [Inria, Engineer, from Oct 2021]

Interns and Apprentices

- Theophile Bastian [École Normale Supérieure de Paris, from Mar 2021 until Aug 2021]
- Theo Degioanni [École normale supérieure de Rennes, from May 2021 until Jul 2021]
- Florian Gallay [Inria, from Mar 2021 until Sep 2021]
- Marius Monnier [Inria, until Jul 2021]
- Ny Aina Pedersen [Univ Grenoble Alpes, from May 2021 until Jul 2021]
- Lev Pikman [Inria, from Apr 2021 until Jul 2021]
- Gabriel Soria-Ramos [Inria, from May 2021 until Jul 2021]

Administrative Assistant

- Maria Immaculada Presseguer [Inria]

2 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 applications*. 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 both on compilers and run-time systems. 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 and run-time systems must *evolve* and, in particular, interact, *taking into account the complexity of the target architecture*.

The overall objective of CORSE is to address this challenge by *combining static and dynamic compilation* techniques, with more interactive *embedding of programs and compiler environment in the run-time system*.

3 Research program

3.1 Scientific Foundations

One of the characteristics of CORSE is to base our researches on diverse advanced mathematical tools. Compiler optimization requires the usage of the several tools around discrete mathematics: combinatorial optimization, algorithmic, and graph theory. The aim of CORSE is to tackle optimization not only for general purpose but also for domain specific applications. We believe that new challenges in compiler technology design and in particular for split compilation should also take advantage of graph labeling techniques. In addition to run-time and compiler techniques for program instrumentation, hybrid analysis and compilation advances will be mainly based on polynomial and linear algebra.

The other specificity of CORSE is to address technical challenges related to compiler technology, run-time systems, and hardware characteristics. This implies mastering the details of each. This is especially important as any optimization is based on a reasonably accurate model. Compiler expertise will be used in modeling applications (e.g. through automatic analysis of memory and computational complexity); Run-time expertise will be used in modeling the concurrent activities and overhead due to contention (including memory management); Hardware expertise will be extensively used in modeling physical resources and hardware mechanisms (including synchronization, pipelines, etc.).

The core foundation of the team is related to the combination of static and dynamic techniques, of compilation, and run-time systems. We believe this to be essential in addressing high-performance and low energy challenges in the context of new important changes shown by current application, software, and architecture trends.

Our project is structured along two main directions. The first direction belongs to the area of run-time systems with the objective of developing strong relations with compilers. The second direction belongs to the area of compiler analysis and optimization with the objective of combining dynamic analysis and optimization with static techniques. The aim of CORSE is to ground those two research activities on the development of the end-to-end optimization of some specific domain applications.

4 Application domains

4.1 Transfer

The main industrial sector related to the research activities of CORSE 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 run-time technology. These applications are based over numerical kernels (linear algebra, FFT, convolution...) that can be adapted on a large spectrum of architectures. More specifically, an important activity concerns the optimization of machine learning applications for some high-performance accelerators. Members of CORSE already maintain fruitful and strong collaborations with several companies such as STMICROELECTRONICS, Atos/Bull, Kalray.

5 Social and environmental responsibility

5.1 Footprint of research activities

This year saw a huge drop with the number of travels. The main cause being the COVID pandemia and travel ban, we expect people to reduce travel activities in the future.

6 Highlights of the year

6.1 Awards

HiPEAC paper award for our PLDI paper IOOPT [10].

7 New software and platforms

This year saw the birth of **EasyTracker**, a library for building visual representations of the execution of a program. Concerning **GUS** (a performance debugging tool based on abstract simulation): We worked on the support for ARM in **pipedream** (a tool for measuring throughput based on micro-benchmarks); we improved the integration of **PALMED** (a tool for reverse engineering the throughput characteristics of superscalar CPUs) and uOP.info (an existing throughput characterization that uses hardware counters) in GUS; we improved the modeling of cache in GUS. We developed PALMED web demo. Concerning **IOLB** (a tool for automatic derivation of data movement complexity of affine programs) and **IOOpt** (a tool for automatic derivation of optimal rectangular tiling for affine programs) , we improved the interface and developed IOOPT web demo. **TTile** has been intergrated in TVM (a compiler framework for deep neural networks) along with several extensions (OpenOpt and Pariopt – search space exploration strategies for compiler optimization). Concerning **BISM**, we worked on augmenting the versatility of the tools by developing new use cases. A redesign and some refactoring also happened for improving usability.

7.1 New software

7.1.1 GUS

Keywords: CPU, Microarchitecture simulation, Performance analysis, Dynamic Analysis

Functional Description: GUS' goal is to detect performance bottlenecks at the very low level on monothread applications by the use of sensitivity analysis. It is coded as a QEMU plug-in in order to collect runtime information that are later treated by the generic CPU model.

URL: https://gitlab.inria.fr/nderumig/gus

Contact: Nicolas Derumigny

7.1.2 Pipedream

Name: Pipedream

Keywords: Performance analysis, CPU, Reverse engineering

Scientific Description: Pipedream reverse engineers the following performance characteristics: (1) Instruction latency – The number of cycles an instruction requires to execute. (2) Peak micro-op retirement rate – How many fused micro-ops the CPU can retire per cycle. (3) Micro-fusion – The number of fused micro-ops an instruction decomposes into. (4) Micro-op decomposition and micro-op port usage – The list of unfused micro-ops every instruction decomposes into and the list of execution ports every one of these micro-ops can execute on.

The first step of the reverse engineering process consists of generating a number of microbenchmarks. Pipedream then runs these benchmark, measuring their performance using hardware counters. The latency, throughput, and micro-fusion of different instructions can then be read directly from these measurements.

The process of finding port mappings, i.e. micro-op decompositions and micro-op port usage, however, is more involved. For this purpose, we have defined a variation of the maximum flow problem which we call the "instruction flow problem". We have developed a linear program (LP) formulation of the instruction flow problem which can be used to calculate the peak IPC and micro-operations per cycle (MPC) a benchmark kernel can theoretically achieve with a given port mapping. The actual port mapping of the underlying hardware is then determined by finding the mapping for which the throughput predicted by instruction flow best matches the actual measured IPC and MPC.

Functional Description: Pipedream is a tool for measuring specific performance characteristics of CPUs It is used to build the performance model of another tool called Gus (https://gitlab.inria.fr/nderumig/gus). Pipedream finds measured performance characteristics such as the throughput and latency of instructions by running a large set of automatically generated microbenchmarks. The tool can also find port mappings, a model of part of the CPU instruction scheduler, by analysing performance measurements of specially crafted microkernels using a LP solver. We have used it to produce a port mapping for the Intel Skylake CPU architecture. Pipedream is able to find the port mappings for some instructions for which existing approaches fall back to manual analysis.

URL: https://gitlab.inria.fr/fgruber/pipedream

Contact: Nicolas Derumigny

7.1.3 IOLB

Keywords: Complexity, Polyhedral compilation, Performance analysis

Functional Description: IOLB computes a symbolic lower bound on the I/O, or data movement, complexity of a computer program, that is the amount of data that needs to be moved between cache and main memory to perform its computation. The input is a C program, and the output is a mathematical formula that depends on program parameters (array sizes...) and cache size.

URL: https://gitlab.inria.fr/CORSE/iolb

Publications: hal-02421026, hal-02910961

Contact: Auguste Olivry

7.1.4 IOOpt

Keywords: I/O, Polyhedral compilation

Functional Description: IOOpt takes as input an abstract representation of a tileable program. The tool generates a tractable set of relevant permutations of the tiling loops, and a symbolic I/O cost expression for each of them. It then uses a non linear problem optimizer to find the best permutations and corresponding tile sizes for a given value of machine parameters (cache sizes and bandwidths at each level). IOOop can also be used to find an upper bound on the I/O cost of a program, for a given tiling scheme.

Publication: hal-03200539

Contact: Auguste Olivry

7.1.5 PALMED

Keywords: CPU, Performance measure, Performance analysis, Reverse engineering

Functional Description: PALMED computes a bipartite graph assembly instructions <-> abstract resources that may be used for performance prediction, targeting static analysis tools and compilers. Internally, PALMED uses PIPEDREAM as a framework for microbenchmarking code generation, and use gurobi to find a first small graph. Then, PALMED deduces from the found resources and the microbenchmarks that saturates them a mapping of every supported instruction.

URL: https://gitlab.inria.fr/nderumig/palmed

Contact: Nicolas Derumigny

7.1.6 BISM

Name: BISM: Bytecode-level Instrumentation for Software Monitoring

Keywords: Java, Bytecode, Instrumentation, Control Flow

Functional Description: BISM (Bytecode-level Instrumentation for Software Monitoring) is a lightweight Java bytecode instrumentation tool which features an expressive high-level control-flow-aware instrumentation language. The language follows the aspect-oriented programming paradigm by adopting the joinpoint model, advice inlining, and separate instrumentation mechanisms. BISM provides joinpoints ranging from bytecode instruction to method execution, access to comprehensive context information, and instrumentation methods. BISM runs in two modes: build-time and load-time.

URL: https://gitlab.inria.fr/bism/bism-public

Publication: hal-03081265

Contact: Ylies Falcone

Participants: Chukri Soueidi, Ylies Falcone, Ali Kassem

7.1.7 TTiLE

Keywords: Deep learning, Optimization, HPC

Functional Description: TTiLE is a code generation tool for tensor operators present in CNNs such as tensor contraction and convolution. It takes as input: 1. a kernel specification, that is, a fonctionnal description of the operator (iteration space, tensors, single assignment description of the computation), 2. an optimization scheme that describes the layered structure of the generated code. The scheme "language" allows to express loop permutation, loop tiling, vectorization, unrolling, packing (which consists in using temporary buffers making cache access better behaved) and branching. Indeed, as opposed to existing schemes that rely on partial tiles, TTile can create non-perfectly nested loops to combine several "micro-kernels" (micro-kernel stands for the innermost part of the loop nest that is fully unrolled, register promoted and vectorized here). Using a specialized search strategy that combines operational research on analytical performance model and native execution time measurement, TTile outperforms current highly optimized libraries such as Intel oneDNN and Intel MKL.

URL: https://gitlab.inria.fr/CORSE/ttile

Publication: hal-03149553

Contact: Nicolas Tollenaere

7.1.8 EasyTracker

Keywords: Monitoring, Debug, Visualization, Teaching of programming

Scientific Description: Learning to program involves building a mental representation of how a machine executes instructions and stores information in memory. To help students, teachers often use visual representations to illustrate executions of programs or particular concepts in their lectures. EasyTracker is a library that assists teachers of programming courses in building tools that generate representations tuned to their needs from actual programs. At its core, EasyTracker provides ways of controlling the execution and inspecting the state of programs. The control and inspection are driven and customized through a Python interface. The controlled program itself can be written either in Python or in any GDB supported language like C.

Functional Description: EasyTracker is intended for computer science teaching. It encapsulates the execution control and monitoring of another program written in Python or any compiled language supported by GDB. It can pause execution at interest points described in a high level language (variable modified or return of recursive functions for example).

URL: https://gitlab.inria.fr/CORSE/easytracker/

Contact: Theo Barollet

Participants: Theo Barollet, Manuel Selva, François Broquedis, Florent Bouchez, Fabrice Rastello

8 New results

8.1 Performance Debugging and Compiler Optimization

Participants: Fabrice Rastello, Guillaume Iooss, Nicolas Derumigny, Théophile Bastian, Auguste Olivry, Nicolas Tollenaere, Fabian Gruber *(ARM)*, Christophe Guillon *(STMIcroelectronics)*, Albert Cohen *(Google, France)*, P. Sadayappan *(OSU, USA)*, Christophe Alias, Louis-Noël Pouchet *(CSU, USA)*, Sanjay Rajopadhye *(CSU, USA)*, Atanas Rountev *(OSU, USA)*.

Our current efforts with regard to code optimization follows two directions.

1. The first consists in improving compiler optimization techniques by considering pattern specific applications such as those that fit into the polyhedral framework or more restrictively those related to machine learning. In this context we developed: 1. a tool for computing data movement complexity (both lower and upper bounds [10]) for affine programs (Section 8.1.1; 2. a new compiler scheme for optimizing computational kernels of DNNs (Section 8.1.2).

2. The second consists in generating dynamic analysis based performance debugging tools. For that purpose we: 1. developed a tool for automatically characterizing CPU resources [5] and, 2. extended a binary translator (QEMU) to perform an abstract simulation based sensitivity analysis (Section 8.1.3).

8.1.1 Automated derivation of parametric data movement complexity for affine programs

Researchers and practitioners have for long worked on improving the computational complexity of algorithms, focusing on reducing the number of operations needed to perform a computation. However the hardware trend nowadays clearly shows a higher performance and energy cost for data movements than computations: quality algorithms have to minimize data movements as much as possible.

The theoretical operational complexity of an algorithm is a function of the total number of operations that must be executed, regardless of the order in which they will actually be executed. But theoretical data movement (or, I/O) complexity is fundamentally different: one must consider all possible legal schedules of the operations to determine the minimal number of data movements achievable, a major theoretical challenge. I/O complexity has been studied via complex manual proofs, e.g., refined from $\Omega(n^3/\sqrt{S})$ for matrix-multiply on a cache size S by Hong & Kung to $2n^3/\sqrt{S}$ by Smith et al. While asymptotic complexity may be sufficient to compare I/O potential between broadly different algorithms, the accuracy of the reasoning depends on the tightness of these I/O lower bounds. Precisely, exposing constants is essential to enable precise comparison between different algorithms: for example the $2n^3/\sqrt{S}$ lower bound allows to demonstrate the optimality of panel-panel tiling for matrix-multiplication.

We developed the first static analysis to automatically derive non-asymptotic parametric expressions of data movement lower bounds with scaling constants, for arbitrary affine computations. Our approach is fully automatic, assisting algorithm designers to reason about I/O complexity and make educated decisions about algorithmic alternatives. The tool allowed us to compute the lower bound for all the kernels of the Polybench suite. We extended this work by: 1. designing the first algorithm for computing a symbolic over-approximation of the data-movement for a parametric (multi-dimensional) tiled version of an affine code 2. designing the first fully automated scheme for expressing as an operational research problem the minimization of this data-movement expression; 3. integrating those techniques into a tool that computes for a class of affine computations a parametric expressions for I/O upper bounds.

A paper that has been presented at PLDI 2021 [10].

8.1.2 Hybrid Performance Modeling and Schedule Optimization for Tensor Computations

Tensor computation such as Sparse Matrix Multi-vector multiplication, Sampled Dense Dense Matrix Multiplication, Dense Matrix Multiplication, Tensor Contraction, Convolution are important kernels used in many domains like Fluid Dynamics, Data Analytics, Economic Modelling, and Machine Learning. Developing highly optimized code for such kernels requires the combination of highly tuned register/instruction level micro-kernels and appropriate multi-level tiling. In this context we developed: an hybrid (analytical/statistical) performance-based optimization scheme along with a code generator for DNNs.

Addressing the problem of automatic generation of optimized operators raises two challenges: The first is associated to the design of a domain specific code generation framework able to output high-quality binary code. The second is to carefully bound the search space and choose an optimizing objective function that neither leads to yet another combinatorial optimizing problem, nor leads to a too approximate performance objective. This work tackles those two challenges by: 1. revisiting the usual belief that packing should enable stride-1 accesses at every level allowing to make packing optional; 2. highlighting the importance of considering the packing decision and shape as being part of

the optimization problem; 3. revisiting the usual belief that register spilling should be avoided if possible allowing to consider other (more packing friendly) micro-kernels as good candidates; 4. revisiting the misleading intuition that convolution dimensions should be brought at the innermost level allowing more freedom for memory reuse at outer-dimensions; 5. showing that the optimization problem can be decoupled into: finding a small set of good micro-kernels candidates using an exhaustive search; finding a good schedule (loop tiling/permutation) and associated packing using operational research; finding the best tiles sizes using auto-tuning; 6. designing a single-pass micro-kernel generation algorithm, to emit code for any choice of register blocking dimensions, unrolling factor, and packing decisions; 7. designing a lowering scheme for abstract iterators, compatible with diverse packing and tiling strategies thrifty with integer arithmetic and loop control usage; 8. designing a packing algorithm compatible with various choices of transposition and subviews; 9. implementing a code generator based on these algorithms, driven by a simple and modular configuration language.

This work has been done in the context of the IOcomplexity associate team (see Section 8.1.1) and the PBI project ES3CAP (see Section 9.2.1).

8.1.3 Automatic Resource Characterization and Performance Feedback

Performance modeling is a critical component for program optimizations, assisting compilers as well as developers in predicting the performance of code variations ahead of time. Performance models can be obtained through different approaches that span from precise and complex simulation of a hardware description (Zesto, GEM5, PTLSim) to application level analytical formulations. An interesting approach for modeling the CPU of modern pipelined, super-scalar, out-of-order processors trades simulation time with accuracy by separately characterizing both latency and throughput of instructions. This approach is suitable both for optimizing compilers, but also for hand-tuning critical kernels written in assembler (see Section 8.1.2). It is used by performance-analysis tools such as CQA, Intel IACA, OSACA, MIAMI or llvm-mca. Cycle-approximate simulators such as ZSim or MCsimA can also take advantage of such an instruction characterization. In this context, we developed two tools: PALMED and GUS (see Section 7).

PALMED: Throughput Characterization for Any Architecture PALMED is a tool that automatically builds a resource mapping, which can be used as a performance model for pipelined, super-scalar, out-of-order CPU architectures. Resource mappings describe the execution of a program by assigning instructions in the program to abstract resources. They can be used to predict the throughput of basic blocks or as a machine model for the backend of an optimizing compiler.

PALMED does not require hardware performance counters, and relies solely on runtime measurements to construct resource mappings. This allows it to model not only execution port usage, but also other limiting resources, such as the frontend or the reorder buffer. Also, thanks to a dual representation of resource mappings, our algorithm for constructing mappings scales to large instruction sets, like that of x86.

GUS: Practical Performance Debugging For Out-of-Order Processors GUS is the first framework for performance debugging of complex, realistic pipelined out-of-order processors executing arbitrary programs, via abstract simulation for sensitivity analysis. Abstract simulation aims at providing significant speed improvement versus cycle-accurate simulation. This allows to perform quickly multiple runs with several value of latency, throughput and count of the modeled resources: this is sensitivity analysis. To determine if a resource is a bottleneck, its capacity is artificially increased to determine whether it affects the (simulated) program execution time accordingly. Such approach is not feasible with cycle-accurate simulator, yet to be realistic it requires an accurate modeling of the execution time.

This work has been done in the context of the European project CPS4EU (see Section 10.3.1).

8.2 Runtime Monitoring, Verification, and Enforcement

Participants: Yliès Falcone, Hosein Nazarpour, Saddek Bensalem, Marius Bozga, Gwen Salaün, Florian Gallay, Chukri Soueidi, Victor Roussanaly.

During the period, our new results and contributions on monitoring can be categorized as follows.

8.2.1 Runtime Enforcement with Reordering, Healing, and Suppression

Runtime enforcement analyses an execution trace, detects when this execution deviates from its expected behaviour with respect to a given property, and corrects the trace to make it satisfy the property. In this work [7], we present new enforcement techniques that reorder actions when necessary, inject actions to the application to ensure progress of the property, and discard actions to avoid storing too many unnecessary actions. At any step of the enforcement, we provide a verdict, called enforcement trend in this work, which takes its value in a 4-valued truth domain. Our approach has been implemented in a tool and validated on several application examples. Experimental results show that our techniques better preserve the application actions, hence ensuring better service continuity.

8.2.2 Semi-automated Modelling of Optimized BPMN Processes

In this work [8], we propose a semi-automated approach for helping non-experts in Business Process Model and Notation (BPMN). We focus on a timed version of BPMN where tasks are associated with a range indicating the minimum and maximum duration needed to execute that task. In a first step, the user defines the tasks involved in the process and possibly gives a partial order between tasks. A first algorithm then generates an abstract graph, which serves as a simplified version of the process being specified. Given such an abstract graph, a second algorithm computes the minimum and maximum time for executing the whole graph. The user can rely on this information for refining the graph. For each version of the graph, these minimum/maximum execution times are computed. Once the user is satisfied with a specific abstract graph, we propose a third algorithm to synthesize the BPMN process from that graph.

8.2.3 Monitoring Distributed Component-Based Systems

In this work [6], we monitor asynchronous distributed component-based systems with multi-party interactions. We consider independent components whose interactions are managed by several distributed schedulers. In this context, neither a global state nor the total ordering of the executions of the system is available at runtime. We instrument the system to retrieve local events from the local traces of the schedulers. Local events are sent to a global observer which reconstructs on-the-fly the set of global traces that are compatible with the local traces, in a concurrency-preserving fashion. The set of compatible global traces is represented in the form of an original lattice over partial states, such that each path of the lattice corresponds to a possible execution of the system.

8.2.4 Decentralized LTL Enforcement

In this work [9], we consider the runtime enforcement of Linear-time Temporal Logic formulas on decentralized systems with no central observation point nor authority. A so-called enforcer is attached to each system component and observes its local trace. Should the global trace violate the specification, the enforcers coordinate to correct their local traces. We formalize the decentralized runtime enforcement problem and define the expected properties of enforcers, namely soundness, transparency and optimality. We present two enforcement algorithms. In the first one, the enforcers explore all possible local modifications to find the best global correction. Although this guarantees an optimal correction, it forces the system to synchronize and is more costly, computation and communication wise. In the second one, each enforcer makes a local correction before communicating. The reduced cost of this version comes at the price of the optimality of the enforcer corrections.

8.2.5 Formal Methods in Outer Space

In this work, we led the edition of an honorary volume dedicated to Klaus Havelund on the occasion of his 66th birthday. The volume features contributions by leading researchers in the domain and a broad range of academic research and engineering successes in programming language design.

8.2.6 Byte-code level Instrumentation for Software Monitoring

In this work, we aim to an efficient and expressive tool for the instrumentation of Java programs at the bytecode-level. BISM (Bytecode-Level Instrumentation for Software Monitoring) is a light-weight Java bytecode instrumentation tool that features an expressive high-level control-flow-aware instrumentation language. The language is inspired by the aspect-oriented programming paradigm in modularizing instrumentation into separate transformers, that encapsulate joinpoint selection and advice inlining. BISM allows capturing joinpoints ranging from bytecode instructions to methods execution and provides comprehensive static and dynamic context information. It runs in two instrumentation modes: build-time and load-time.

8.3 Teaching of Algorithms, Programming, Debugging, and Automata

Participants: Théo Barollet, Florent Bouchez Tichadou, Manuel Selva, François Broquedis, Fabrice Rastello.

This domain is a new axis of the CORSE team. Our goal here is to combine our expertise in compilation and teaching to help teachers and learners in computer science fields such as programming, algorithms, data structures, automata, or more generally computing litteracy. This axis is derived into two projects. The first is the automated generation and recommendation of exercises using artificial intelligence. The second focuses on tools to help learning through visualization (data structures, debugger, automata) or gamification.

8.3.1 Do Common Educational Datasets Contain Static Information? A Statistical Study?

In Intelligent Tutoring Systems (ITS), methods to choose the next exercise for a student are inspired from generic recommender systems, used, for instance, in online shopping or multimedia recommendation. As such, collaborative filtering, especially matrix factorization, is often included as apart of recommendation algorithms in ITS. One notable difference in ITS is the rapid evolution of users, who improve their performance, as opposed to multimedia recommendation where preferences are more static. This raises the following question: how reliably can we use matrix factorization, a tool tried and tested in a static environment, in a context where timelines seem to be of importance.In this work we tried to quantify empirically how much information can be extracted statically from datasets in education versus datasets in multimedia, as the quality of such information is critical to be able to accurately make predictions and recommendations. We found that educational datasets contain less static information compared to multimedia datasets, to the extent that vectors of higher dimensions only marginally increase the precision of the matrix factorization compared to a 1-dimensional characterization. These results show that educational datasets must be used with time information, and warn against the dangers of directly trying to use existing algorithms developed for static datasets.

This work has been done in the context of the AI4HI Inria exploratory project and has been presented at EDM 2021 conference [4].

8.3.2 A generic library for controlling and inspecting program execution and state

Learning to program involves building a mental representation of how a machine executes instructions and stores information in memory. To help students, teachers often use visual representations to illustrate executions of programs or particular concepts in their lectures. As a famous example, references/pointers are very often represented with arrows pointing to objects or memory locations. While these visual representations are most of the time hand-drawn, they nowadays tend to be supplemented by tool-generated ones. These tools have the advantage of being usable by learners, empowering them with the ability of validating their own understanding of the concept the tool aims at representing. However, building such a tool from scratch requires a lot of effort and a high level of technical expertize, and the ones that already exist are difficult to adapt to different contexts. In this work we developped EasyTracker, a library that assists teachers of programming courses in building tools that generate representations tuned to their needs from actual programs. At its core, EasyTracker provides ways of controlling the

execution and inspecting the state of programs. The control and inspection are driven and customized through a Python interface. The controlled program itself can be written either in Python or in any GDB supported language like C. This work showcases two tools built on EasyTracker which are used in a teaching context to explain the notions of stack and heap, and to visualize recursion as well as a game prototype to help students learn debugging.

This work has been submitted for publication at ACM ITICSE 2022.

8.4 COVID related activities

Participants: Fabrice Rastello, Nicolas Terzi *(CHU Grenoble)*, Christophe Déhan *(MinMaxMedical)*, Florian Sigaud *(CHU Grenoble)*, Guillaume Rigault *(CHU Grenoble)*, Stan Borkowski, Cyril Fromentin *(FineHeart)*, Adrien Farrugia *(SteadXP)*, Claude Guérin *(HCL Lyon)*.

8.4.1 Mechanized Assisted Ventilation

To address the issue of ventilator shortages, our group (eSpiro Network) developed a freely replicable, open-source hardware ventilator. **Design**: We performed a bench study. Setting: Dedicated research room as part of an ICU affiliated to a university hospital. **Subjects**: We set the lung model with three conditions of resistance and linear compliance for mimicking different respiratory mechanics of representative intensive care unit (ICU) patients. **Interventions**: The performance of the device was tested using the ASL5000 lung model. **Measurements and Main Results**: Twenty-seven conditions were tested. All the measurements fell within the 10% limits for the tidal volume (VT). The volume error was influenced by the mechanical condition ($p = 5.9 \times 10 - 15$) and the PEEP level ($P = 1.1 \times 10 - 12$) but the clinical significance of this finding is likely meaningless (maximum -34 mL in the error). The PEEP error was not influenced by the mechanical condition ($p = 0.25$). Our experimental results demonstrate that the eSpiro ventilator is reliable to deliver VT and PEEP accurately in various respiratory mechanics conditions. Conclusions: We report a low-cost, easy-to-build ventilator, which is reliable to deliver VT and PEEP in passive invasive mechanical ventilation.

This work [2] has been done in the context of the Recovid project supported by Inria.

9 Bilateral contracts and grants with industry

Participants: François Broquedis, Frédéric Desprez, Mathieu Stoffel, Fabrice Rastello, Nicolas Tolenaere, Guillaume Iooss, Stephane Pouget, Hugo Brunie.

9.1 Bilateral contracts with industry

9.1.1 Atos/Bull

- Title: Static and dynamic approaches for the optimization of the energy consumption associated with applications of the High Performance Computing (HPC) field
- CORSE participants: François Broquedis, Frédéric Desprez, Mathieu Stoffel
- Partner: Atos/Bull
- Duration: February 2018 - February 2021
- Abstract: The purpose of this project is to dynamically improve the energy consumption of HPC applications on large-scale platforms. It relies on an adaptation of the CPU frequency at runtime,

based on the analysis of hardware-related metrics to determine an *application profile*. This profile is then split into different *phases*, each of which being associated to a best CPU frequency, depending on its nature (CPU bound, memory bound, ...). This project is funding the PhD of Mathieu Stoffel, and the corresponding development is to be integrated into *Bull Dynamic Power Optimizer*, a software suite developed by Atos/Bull.

9.2 Bilateral grants with industry

9.2.1 ES3CAP

- Title: Embedded Smart Safe Secure Computing Autonomous Platform
- CORSE participants: Fabrice Rastello, Nicolas Tolenaere
- Duration: July 2018 - February 2022
- INRIA Partners: AOSTE, PARKAS, CHROMA
- Other Partners: Renault-Nissan, EasyMile, Safran E&D, MBDA, ANSYS/ESterel Technologies, Kronno-Safe, Prove & Run, Kalray, Prophesee, CEA
- Abstract: The objective of ES3CAP is to develop a tool-chain that targets multi- and many-core architectures for critical systems. In particular it should address the different challenges related to making existing critical systems solutions (heterogeneous, decentralized, single-core, single-task) match the industrial constraints targeted by Kalray's MPPA (MPPA, high-performance, real-time, safety, security). Considered applications are autonomous driving, drones, avionics, and defense. CORSE is involved in the optimization of machine learning algorithms for many-core architectures.

10 Partnerships and cooperations

Participants: Ylies Falcone.

10.1 International initiatives

10.1.1 Inria associate team not involved in an IIL or an international program

RV4IoT

Title: Runtime Verification for the Internet of Things

Duration: 2020 ->

Coordinator: Sylvain Hallé (shalle@acm.org)

Partners:

- Université du Québec à Chicoutimi

Inria contact: Ylies Falcone

Summary: The research of the team focuses on the runtime verification of systems in the so-called domain of internet of things. The team objective is to develop methods and tools to guarantee the safety and security of such systems by using the complementary expertise from the teams.

10.2 International research visitors

10.2.1 Visits of international scientists

Other international visits to the team

Sylvain Hallé

Status (professor)

Institution of origin: University of québec at Chicoutimi

Country: Canada

Dates: December 2021

Context of the visit: RV4IoT associate team

10.2.2 Visits to international teams

Research stays abroad

Nicolas Derumigny

Visited institution: Colorado State University (CSU)

Country: USA

Dates: Sept 2021 – Now

Mobility program/type of mobility: Cotutelle

10.3 European initiatives

10.3.1 FP7 & H2020 projects

CPS4EU

Title: Cyber Physical Systems for Europe

Duration: July 2019 – June 2022

Coordinator: Philippe Gougeon, VALEO VISION SAS

Partners:

- ABENGOA INNOVACION SOCIEDAD ANONIMA (Spain)
- ANSYS FRANCE SAS (France)
- BUDAPESTI MUSZAKI ES GAZDASAGTUDOMANYI EGYETEM (Hungary)
- CENTRE NATIONAL DE LA RECHERCHE SCIENTIFIQUE CNRS (France)
- COMMISSARIAT A L ENERGIE ATOMIQUE ET AUX ENERGIES ALTERNATIVES (France)
- EMMTRIX TECHNOLOGIES GMBH (Germany)
- ETH LAB SRL (Italy)
- EUROTECH SPA (Italy)
- FUNDACION CENTRO DE TECNOLOGIAS DE INTERACCION VISUAL Y COMUNICACIONES VICOMTECH (Spain)
- GREENWAVES TECHNOLOGIES (France)
- INSTITUTO TECNOLOGICO DE INFORMATICA (Spain)
- KALRAY SA (France)

- LEONARDO - SOCIETA PER AZIONI (Italy)
- M3 SYSTEMS SAS (France)
- PROVE & RUN (France)
- SCHNEIDER ELECTRIC FRANCE SAS (France)
- SEQUANS COMMUNICATIONS SA (France)
- SHERPA ENGINEERING SA (France)
- SPINSPLIT MUSZAKI KUTATO FEJLESZTOKFT (Hungary)
- TECHNISCHE UNIVERSITAT CLAUSTHAL (Germany)
- TECNOLOGIAS SERVICIOS TELEMATICOS YSISTEMAS SA (Spain)
- THALES (France)
- UNIVERSITAET AUGSBURG (Germany)
- UNIVERSITE DE LORRAINE (France)
- UNIVERSITE GRENOBLE ALPES (France)
- VALEO COMFORT AND DRIVING ASSISTANCE (France)
- VALEO VISION SAS (France)

Inria contact: Fabrice Rastello

Summary: Cyber Physical Systems (CPS) represent key drivers for the innovation capacity of European industries, large and small, generating sustainable economic growth and supporting meaningful jobs for citizens. The ultimate objective of CPS4EU is to strengthen the CPS value chain by creating world class European SMEs and by providing CPS technologies that in turn will sustain the leadership of the large European groups in key economy sectors and, in this way will stimulate innovative products to support the massive digitization increasingly integrated into our everyday environment.

10.4 National initiatives

ANR SEVERITAS

Title: Secure and Verifiable Test and Assessment System (SEVERITAS)

Duration: May 2021 – April 2025

Coordinator: Ylies Falcone

Partners:
- Laboratoire d'Informatique de Grenoble (LIG)
- Laboratoire d'Informatique, de Modélisation et d'Optimisation des Systèmes (LIMOS)
- University of Luxembourg / Interdisciplianary Center for Security, Reliability and Trust (SnT/UL)
- Laboratoire lorrain de recherche en informatique et ses applications (LORIA)

CORSE contact: Ylies Falcone

Summary: SEVERITAS advances information socio-technical security for Electronic Test and Assessment Systems (e-TAS). These systems measure skills and performances in education and training. They improve management, reduce time-to-assessment, reach larger audiences, but they do not always provide security by design. This project recognizes that the security aspects for e-TAS are still mostly unexplored. We fill these gaps by studying current and other to-be-defined security properties. We develop automated tools to advance the formal verification of security and show how to validating e- TAS security. rigorously. We also develop new secure, transparent, verifiable and lawful e-TAS procedures and protocols. We also deploy novel run-time monitoring strategies to reduce frauds and study the user experience about processes to foster e-TAS usable security. And thanks to connections with players in the business of e-TAS, such as OASYS, this project will contribute to the development of secure e-TAS.

11 Dissemination

Participants: Florent Bouchez-Tichadou, Yliès Falcone, Guillaume Iooss, Fabrice Rastello, Manuel Selva.

11.1 Promoting scientific activities

11.1.1 Scientific events: selection

Member of the Conference Steering Committee

- Fabrice Rastello: Steering Committee Chair of ACM/IEEE CGO
- Yliès Falcone: Member of the Steering Committee of the Runtime Verification conference.
- Yliès Falcone: Member of the Steering Committee of Software Verification and Testing track at the Symposium on Applied Computing.

Member of the conference program committees

- Fabrice Rastello: ACM/IEEE CGO 2022

Reviewer

- Manuel Selva: COMPAS
- Yliès Falcone: RV 2021, SAC-SVT 2022

11.1.2 Journal

Reviewer - reviewing activities

- Fabrice Rastello: IEEE TPDS, ACM TOPLAS
- Yliès Falcone: Formal Methods in System Design, Software Tools for Technology Transfer
- Guillaume Iooss: ACM TOPLAS

11.1.3 Invited talks

- Ylies Falcone gave an invited keynote at VECOS 2021.

11.1.4 Leadership within the scientific community

- Fabrice Rastello: scientific council of Inria Grenoble Rhône-Alpes (COS)

11.2 Teaching - Supervision - Juries

11.2.1 Teaching

- License 3: François Broquedis, Imperative programming using python, 60 hours, Grenoble Institute of Technology (Ensimag)
- License 3: François Broquedis, Introduction to UNIX, 20 hours, Grenoble Institute of Technology (Ensimag)
- License 3: François Broquedis, C programming, 100 hours, Grenoble Institute of Technology (Ensimag)

- Master 1: François Broquedis, Object-Oriented Programming, 20 hours, Grenoble Institute of Technology (Ensimag)
- François Broquedis is in charge of the first year study at Ensimag
- Master: Florent Bouchez Tichadou, Algorithmic Problem Solving, 41 hours, M1 MoSIG.
- Licence: Florent Bouchez Tichadou, Algorithms languages and programming, 103 hours, L2 UGA.
- Master: Florent Bouchez Tichadou, remise à niveau d'agents de la SNCF en reconversion, 60 hours, UGA.
- DIU EIL, Florent Bouchez Tichadou, formation des enseignants du secondaire suite à la réforme du Baccalauréat. Bloc Algorithmique. 30 hours, UGA.
- Master 1: Yliès Falcone, Programming Language Semantics and Compiler Design, MoSIG and Master informatique, 45 hours
- License: Yliès Falcone, Languages and Automata, Univ. Grenoble Alpes, 45 hours
- Master: Yliès Falcone, is responsible for the two above courses.
- License 3: Manuel Selva, Imperative programming using python, 124 hours, Grenoble Institute of Technology (Ensimag)
- License 3: Manuel Selva is responsible for the above course.
- License 3: Manuel Selva, Introduction to UNIX, 20 hours, Grenoble Institute of Technology (Ensimag)
- License 3: Guillaume Iooss, Imperative programming using python (TP), 33 hours, Grenoble Institute of Technology (Ensimag)
- Licence 2: Marius Monnier, INF302: Languages and automata, 24 hours, DLST (UGA)
- Master 1: Auguste Olivry, Algorithmic Problem Solving, 41 hours, UGA UFR IM2AG / Grenoble INP Ensimag
- License 2: Auguste Olivry, Algorithms languages and imperative programming (TP), 18 hours, DLST, UGA UFR IM2AG
- Licence 2, Nicolas Tollenaere, Algorithms languages and imperative programming (TD/TP), 36 hours, DLST, UGA UFR IM2AG
- Licence 2, Nicolas Derumigny, Algorithms languages and imperative programming (TD/TP), 42 hours, DLST, UGA UFR IM2AG
- License 2: Theo Barollet, Algorithms languages and imperative programming (TD-TP), 33 hours, UGA
- Master 1: Theo Barollet, Compilation Project, 18h, UGA

11.2.2 Supervision

- PhD in progress: Marius Monnier. Who guards the guardians? Towards trustable and robust enforcement monitoring, advised by Ylies Falcone.
- PhD in progress: Théophile Bastian, Performance study: identifying bottlenecks by means of sensitivity analysis, September 2021, advised by Fabrice Rastello.
- PhD: Mathieu Stoffel, Static and dynamic approaches for the optimization of the energy consumption associated with applications of the High Performance Computing (HPC) field, Jan 2021, advised by François Broquedis, Frédéric Desprez, Abdelhafid Mazouz (Atos/Bull) and Philippe Rols (Atos/Bull)

- PhD in progress: Auguste Olivry, Data Locality and Parallelism Optimization for Linear and Multilinear Algebra, September 2019, advised by Fabrice Rastello.
- PhD in progress: Nicolas Tollenaere, Optimizing ML algorithms for MPPA Asics, April 2019, advised by Fabrice Rastello and Guillaume Iooss.
- PhD in progress: Nicolas Derumigny, Automatic generation of performance models for heterogeneous architectures, September 2019, advised by Fabrice Rastello.
- PhD in progress: Théo Barollet, Problem-based learning: automatic generation and recommendation of programming exercises, September 2019, advised by Florent Bouchez Tichadou and Fabrice Rastello.
- PhD in progress: Chukri Soueidi, Instrumentation, Runtime Verification and Enforcement for Multithreaded Programs, October 2020, advised by Yliès Falcone.

11.2.3 Juries

Fabrice Rastello

- Bahram Yarahmadi– Rennes, Examinateur/Rapporteur, July 1 2021 "Static ansd dynamic compiler support for intermittently powered computer systems"
- Idriss Daoudi– Bordeaux, Examinateur, Sept 21 2021 "Modélisation de performance et simulation d'applications OpenMP"

11.3 Popularization

11.3.1 Internal or external Inria responsibilities

- Fabrice Rastello; scientific council CEA-EDF-Inria summer schools

11.3.2 Education

- Florent Bouchez Tichadou, DIU EIL, Formation des enseignants des lycées suite à la réforme du Bac et l'introduction de l'option informatique en 1ère et Terminale (NSI). Apprentissage Par Problèmes. Algorithmique, programmation, debug. Académie de Grenoble.

12 Scientific production

12.1 Publications of the year

International journals

[1] G. Iooss, C. Alias and S. Rajopadhye. 'Monoparametric Tiling of Polyhedral Programs'. In: *International Journal of Parallel Programming* 49 (18th Mar. 2021), pp. 376–409. DOI: 10.1007/s10766-021-00694-2. URL: https://hal.inria.fr/hal-02493164.

[2] N. Terzi, F. Rastello, C. Déhan, M. Roux, F. Sigaud, G. Rigault, C. Fromentin, A. Farrugia and C. Guérin. 'The eSpiro Ventilator: An Open-Source Response to a Worldwide Pandemic'. In: *Journal of Clinical Medicine* 10.11 (June 2021), p. 2336. DOI: 10.3390/jcm10112336. URL: https://hal.inria.fr/hal-03536759.

International peer-reviewed conferences

[3] M. Á. Abella-González, P. Carollo-Fernández, L.-N. Pouchet, F. Rastello and G. Rodríguez. 'PolyBench/Python: benchmarking Python environments with polyhedral optimizations'. In: CC 2021 - 30th ACM SIGPLAN International Conference on Compiler Construction. Seoul, South Korea: ACM, 2nd Mar. 2021, pp. 59–70. DOI: 10.1145/3446804.3446842. URL: https://hal.inria.fr/hal-03153351.

[4] T. Barollet, F. Bouchez-Tichadou and F. Rastello. 'Do Common Educational Datasets Contain Static Information? A Statistical Study'. In: EDM 2021 - Conference on Educational Data Mining. Paris / Virtual, France, 29th June 2021, pp. 1–7. URL: https://hal.inria.fr/hal-03526276.

[5] N. Derumigny, T. Bastian, F. Gruber, G. Iooss, C. Guillon, L.-N. Pouchet and F. Rastello. 'PALMED: Throughput Characterization for Superscalar Architectures'. In: International Symposium on Code Generation and Optimization (CGO). Seoul, South Korea, 2nd Apr. 2022. URL: https://hal.inria.fr/hal-03531740.

[6] Y. Falcone, H. Nazarpour, S. Bensalem and M. Bozga. 'Monitoring Distributed Component-Based Systems'. In: FACS 2021 - 17th edition of the International Conference on Formal Aspects of Component Software. Grenoble, France, 28th Oct. 2021, pp. 1–19. URL: https://hal.inria.fr/hal-03525762.

[7] Y. Falcone and G. Salaün. 'Runtime Enforcement with Reordering, Healing, and Suppression'. In: SEFM 2021 - 19th IEEE International Conference on Software Engineering and Formal Methods. Virtual, United Kingdom: IEEE, 6th Dec. 2021, pp. 1–20. URL: https://hal.inria.fr/hal-03484045.

[8] Y. Falcone, G. Salaün and A. Zuo. 'Semi-automated Modelling of Optimized BPMN Processes'. In: SCC 2021 - IEEE International Conference on Services Computing. CHICAGO / Virtual, United States: IEEE, 5th Sept. 2021, pp. 1–6. URL: https://hal.inria.fr/hal-03330330.

[9] F. Gallay and Y. Falcone. 'Decentralized LTL Enforcement'. In: GandALF 2021 - 12th International Symposium on Games, Automata, Logics, and Formal Verification. Padua, France, 20th Sept. 2021, pp. 1–18. URL: https://hal.inria.fr/hal-03525845.

[10] A. Olivry, G. Iooss, N. Tollenaere, A. Rountev, P. Sadayappan and F. Rastello. 'IOOpt: Automatic Derivation of I/O Complexity Bounds for Affine Programs'. In: PLDI 2021 - 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation. Proceedings of the 42nd ACM SIGPLAN International Conference on Programming Language Design and Implementation. Virtual, Canada, 18th June 2021. DOI: 10.1145/3453483. URL: https://hal.inria.fr/hal-03200539.

[11] M. Stoffel, F. Broquedis, F. Desprez and A. Mazouz. 'Phase-TA: Periodicity Detection and Characterization for HPC Applications'. In: HPCS 2020 - 18th IEEE International Conference on High Performance Computing and Simulation. Proceedings of the IEEE 18th International Conference on High Performance Computing and Simulation (HPCS 2020). Barcelone / Virtual, Spain: IEEE, 22nd Mar. 2021, pp. 1–12. URL: https://hal.archives-ouvertes.fr/hal-03185251.

Scientific books

[12] E. Bartocci, Y. Falcone and M. Leucker. *Formal Methods in Outer Space*. Vol. 13065. Lecture Notes in Computer Science. Springer International Publishing, 2021, pp. 1–193. DOI: 10.1007/978-3-030-87348-6. URL: https://hal.inria.fr/hal-03533127.

Reports & preprints

[13] N. Derumigny, T. Bastian, F. Gruber, G. Iooss, C. Guillon, L.-N. Pouchet and F. Rastello. *PALMED: Throughput Characterization for Superscalar Architectures - Extended Version*. 18th Jan. 2022. URL: https://hal.inria.fr/hal-03114933.

[14] C. Soueidi, M. Monnier, A. Kassem and Y. Falcone. *Efficient and Expressive Bytecode-Level Instrumentation for Java Programs*. 2nd June 2021. URL: https://hal.inria.fr/hal-03533152.

[15] N. Tollenaere, A. Olivry, G. Iooss, H. Brunie, A. Cohen, P. Sadayappan and F. Rastello. *Efficient convolution optimisation by composing micro-kernels*. 14th Oct. 2021. URL: https://hal.archives-ouvertes.fr/hal-03149553.