IN PARTNERSHIP WITH:
Université Rennes 1

Activity Report 2019

Project-Team PACAP

Pushing Architecture and Compilation for Application Performance

IN COLLABORATION WITH: Institut de recherche en informatique et systèmes aléatoires (IRISA)

RESEARCH CENTER
Rennes - Bretagne-Atlantique

THEME
Architecture, Languages and Compilation

Table of contents

1. Team, Visitors, External Collaborators **2**
2. Overall Objectives **2**
2.1.1. Long-Term Goal 2
2.1.2. Approach 3
2.1.3. Latency-oriented Computing 3
2.1.4. Throughput-Oriented Computing 3
2.1.5. Real-Time Systems – WCET 4
2.1.6. Performance Assessment 4
2.1.7. Dealing with Attacks – Security 4
2.1.8. Green Computing – Power Concerns 5
3. Research Program **5**
3.1. Motivation 5
3.1.1. Technological constraints 5
3.1.2. Evolving community 5
3.1.3. Domain constraints 6
3.2. Research Objectives 6
3.2.1. Static Compilation 7
3.2.2. Software Adaptation 7
3.2.3. Research directions in uniprocessor micro-architecture 7
3.2.4. Towards heterogeneous single-ISA CPU-GPU architectures 9
3.2.5. Real-time systems 9
3.2.6. Power efficiency 10
3.2.7. Security 11
4. Application Domains **11**
5. Highlights of the Year **11**
6. New Software and Platforms **12**
6.1. ATMI 12
6.2. HEPTANE 12
6.3. tiptop 12
6.4. PADRONE 13
6.5. If-memo 13
6.6. Simty 14
6.7. Barra 14
6.8. Memoization 14
6.9. FiPlib 15
6.10. sigmask 15
7. New Results **15**
7.1. Compilation and Optimization 15
7.1.1. Optimization in the Presence of NVRAM 15
7.1.1.1. Checkpoint Placement based Worst-Case Energy Consumption 15
7.1.1.2. Dynamic Adaptive Checkpoint Placement 16
7.1.2. Dynamic Binary Optimization 16
7.1.2.1. Guided just-in-time specialization 16
7.1.2.2. Run-time parallelization and de-parallelization 16
7.1.3. Automatic and Parametrizable Memoization 17
7.1.4. Autotuning 17
7.1.5. Loop splitting 17
7.1.6. Hardware/Software JIT Compiler 17
7.1.7. Scalable program tracing 18

7.1.8. Compiler optimization for quantum architectures 18
7.2. Processor Architecture 18
7.2.1. Value prediction 18
7.2.2. Compressed caches 18
7.2.3. Deep microarchitecture 19
7.2.4. Dynamic thermal management 19
7.2.5. Thread convergence prediction for general-purpose SIMT architectures 19
7.2.6. Exploring the design space of GPU architectures 20
7.3. WCET estimation and optimization 20
7.3.1. WCET estimation for many core processors 20
7.3.1.1. Optimization of WCETs by considering the effects of local caches 20
7.3.1.2. Shared resource contentions and WCET estimation 21
7.3.1.3. Interference-sensitive run-time adaptation of time-triggered schedules 21
7.3.1.4. WCET-Aware Parallelization of Model-Based Applications for Multi-Cores 21
7.3.2. WCET estimation and optimizing compilers 21
7.3.3. WCET estimation and processor micro-architecture 22
7.4. Security 22
7.4.1. Attack detection co-processor for real-time systems 22
7.4.2. Multi-nop fault injection attack 23
7.4.3. Compiler-based automation of side-channel countermeasures 23
7.4.4. Platform for adaptive dynamic protection of programs 23
8. Bilateral Contracts and Grants with Industry **23**
9. Partnerships and Cooperations **23**
9.1. Regional Initiatives 23
9.2. National Initiatives 24
9.2.1. Zero Power Computing Systems (ZEP): Inria Project Lab (2017–2020) 24
9.2.2. NOPE 24
9.2.3. Hybrid SIMD architectures (2018–2019) 24
9.2.4. DGA/PEC ARMOUR (2018–2021) 24
9.2.5. ANR DYVE (31/03/2020 – 30/09/2023) 25
9.3. European Initiatives 25
9.3.1.1. ARGO 25
9.3.1.2. HiPEAC4 NoE 26
9.3.1.3. Eurolab-4-HPC 26
9.4. International Initiatives 27
9.4.1. ANR CHIST-ERA SECODE 2016–2019 27
9.4.2. Informal International Partners 27
10. Dissemination **27**
10.1. Promoting Scientific Activities 27
10.1.1. Scientific Events: Selection 27
10.1.1.1. Member of the Conference Program Committees 27
10.1.1.2. Reviewer 28
10.1.2. Journal 28
10.1.2.1. Member of the Editorial Boards 28
10.1.2.2. Reviewer - Reviewing Activities 28
10.1.3. Invited Talks 28
10.1.4. Leadership within the Scientific Community 28
10.1.5. Research Administration 29
10.2. Teaching - Supervision - Juries 29
10.2.1. Teaching 29
10.2.2. Supervision 29

10.2.3. Juries 30
10.3. Popularization 30
10.3.1. Internal or external Inria responsibilities 30
10.3.2. Articles and contents 31
10.3.3. Education 31
10.3.4. Interventions 31
10.3.5. Internal action 31
11. Bibliography **31**

Project-Team PACAP

Creation of the Project-Team: 2016 July 01

Keywords:

Computer Science and Digital Science:

A1.1. - Architectures
A1.1.1. - Multicore, Manycore
A1.1.2. - Hardware accelerators (GPGPU, FPGA, etc.)
A1.1.3. - Memory models
A1.1.4. - High performance computing
A1.1.5. - Exascale
A1.1.9. - Fault tolerant systems
A1.1.10. - Reconfigurable architectures
A1.1.11. - Quantum architectures
A1.6. - Green Computing
A2.2. - Compilation
A2.2.1. - Static analysis
A2.2.2. - Memory models
A2.2.4. - Parallel architectures
A2.2.5. - Run-time systems
A2.2.6. - GPGPU, FPGA...
A2.2.7. - Adaptive compilation
A2.2.8. - Code generation
A2.2.9. - Security by compilation
A2.3.1. - Embedded systems
A2.3.3. - Real-time systems
A4.2. - Correcting codes
A4.4. - Security of equipment and software
A8.9. - Performance evaluation
A8.10. - Computer arithmetic

Other Research Topics and Application Domains:

B1. - Life sciences
B2. - Health
B3. - Environment and planet
B4. - Energy
B5. - Industry of the future
B6. - IT and telecom
B7. - Transport and logistics
B8. - Smart Cities and Territories
B9. - Society and Knowledge

1. Team, Visitors, External Collaborators

Research Scientists

Erven Rohou [Team leader, Inria, Senior Researcher, HDR]
Caroline Collange [Inria, Researcher]
Byron Hawkins [Inria, Starting Research Position]
Pierre Michaud [Inria, Researcher]
André Seznec [Inria, Senior Researcher, HDR]

Faculty Members

Damien Hardy [Univ de Rennes I, Associate Professor]
Isabelle Puaut [Univ de Rennes I, Professor, HDR]

Post-Doctoral Fellows

Pierre-Yves Peneau [Inria]
Stefanos Skalistis [Univ de Rennes I, until Jun 2019]

PhD Students

Nicolas Bellec [Inria, from Sep 2019]
Arthur Blanleuil [Univ de Rennes I]
Niloofar Charmchi [Inria]
Kleovoulos Kalaitzidis [Inria]
Kévin Le Bon [Inria]
Anis Peysieux [Inria, from Oct 2019]
Daniel Rodrigues Carvalho [Inria]
Bahram Yarahmadi [Inria]

Technical staff

Alexandre Kouyoumdjian [Inria, Engineer]
Stefanos Skalistis [Inria, Engineer, from Jul 2019]
Loïc Besnard [CNRS]

Interns and Apprentices

Nicolas Bellec [École normale supérieure de Rennes, from Feb 2019 until Jun 2019]
Piéric Giraud [Inria, from May 2019 until Aug 2019]
Anis Peysieux [Inria, from Mar 2019 until Sep 2019]
Nicolas Poirier [None, Feb 2019]

Administrative Assistant

Virginie Desroches [Inria]

2. Overall Objectives

2.1. Overall Objectives

2.1.1. Long-Term Goal

In brief, the long-term goal of the PACAP project-team is about *performance*, that is: how fast programs run. We intend to contribute to the ongoing race for exponentially increasing performance and for performance guarantees.

Traditionally, the term “performance” is understood as “how much time is needed to complete execution”. *Latency*-oriented techniques focus on minimizing the average-case execution time (ACET). We are also interested in other definitions of performance. *Throughput*-oriented techniques are concerned with how many units of computations can be completed per unit of time. This is more relevant on manycores and GPUs where many computing nodes are available, and latency is less critical. Finally, we also study worst-case execution time (WCET), which is extremely important for critical real-time systems where designers must guarantee that deadlines are met, in any situation.

Given the complexity of current systems, simply assessing their performance has become a non-trivial task which we also plan to tackle.

We occasionally consider other metrics related to performance, such as power efficiency, total energy, overall complexity, and real-time response guarantee. Our ultimate goal is to propose solutions that make computing systems more efficient, taking into account current and envisioned applications, compilers, runtimes, operating systems, and micro-architectures. And since increased performance often comes at the expense of another metric, identifying the related trade-offs is of interest to PACAP.

The previous decade witnessed the end of the “magically” increasing clock frequency and the introduction of commodity multicore processors. PACAP is experiencing the end of Moore’s law [1], and the generalization of commodity heterogeneous manycore processors. This impacts how performance is increased and how it can be guaranteed. It is also a time where exogenous parameters should be promoted to first-class citizens:

1. the existence of faults, whose impact is becoming increasingly important when the photo-lithography feature size decreases;
2. the need for security at all levels of computing systems;
3. *green* computing, or the growing concern of power consumption.

2.1.2. Approach

We strive to address performance in a way as transparent as possible for users. For example, instead of proposing any new language, we consider existing applications (written for example in standard C), and we develop compiler optimizations that immediately benefit programmers; we propose microarchitectural features as opposed to changes in processor instruction sets; we analyze and re-optimize binary programs automatically, without any user intervention.

The perimeter of research directions of the PACAP project-team derives from the intersection of two axes: on the one hand, our high-level research objectives, derived from the overall panorama of computing systems, on the other hand the existing expertise and background of the team members on key technology (see illustration on Figure 1). Note that it does not imply that we will systematically explore all intersecting points of the figure, yet all correspond to a sensible research direction. These lists are neither exhaustive, nor final. Operating systems in particular constitute a promising operating point for several of the issues we plan to tackle. Other aspects will likely emerge during the lifespan of the project-team.

2.1.3. Latency-oriented Computing

Improving the ACET of general purpose systems has been the “core business” of PACAP’s ancestors (CAPS and ALF) for two decades. We plan to pursue this line of research, acting at all levels: compilation, dynamic optimizations, and micro-architecture.

2.1.4. Throughput-Oriented Computing

The goal is to maximize the performance-to-power ratio. We will leverage the execution model of throughput-oriented architectures (such as GPUs) and extend it towards general purpose systems. To address the memory wall issue, we will consider bandwidth saving techniques, such as cache and memory compression.

[1]Moore’s law states that the number of transistors in a circuit doubles (approximately) every two years.

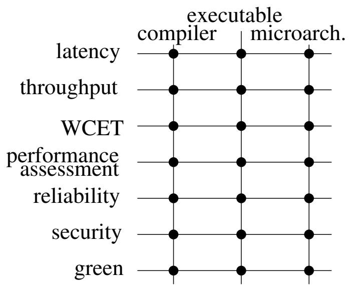

Figure 1. Perimeter of Research Objectives

2.1.5. Real-Time Systems – WCET

Designers of real-time systems must provide an upper bound of the worst-case execution time of the tasks within their systems. By definition this bound must be safe (i.e., greater than any possible execution time). To be useful, WCET estimates have to be as tight as possible. The process of obtaining a WCET bound consists in analyzing a binary executable, modeling the hardware, and then maximizing an objective function that takes into account all possible flows of execution and their respective execution times. Our research will consider the following directions:

1. better modeling of hardware to either improve tightness, or handle more complex hardware (e.g. multicores);
2. eliminate unfeasible paths from the analysis;
3. consider probabilistic approaches where WCET estimates are provided with a confidence level.

2.1.6. Performance Assessment

Moore's law drives the complexity of processor micro-architectures, which impacts all other layers: hypervisors, operating systems, compilers and applications follow similar trends. While a small category of experts is able to comprehend (parts of) the behavior of the system, the vast majority of users are only exposed to – and interested in – the bottom line: how fast their applications are actually running. In the presence of virtual machines and cloud computing, multi-programmed workloads add yet another degree of non-determinism to the measure of performance. We plan to research how application performance can be characterized and presented to a final user: behavior of the micro-architecture, relevant metrics, possibly visual rendering. Targeting our own community, we also research techniques appropriate for fast and accurate ways to simulate future architectures, including heterogeneous designs, such as latency/throughput platforms.

Once diagnosed, the way bottlenecks are addressed depends on the level of expertise of users. Experts can typically be left with a diagnostic as they probably know better how to fix the issue. Less knowledgeable users must be guided to a better solution. We plan to rely on iterative compilation to generate multiple versions of critical code regions, to be used in various runtime conditions. To avoid the code bloat resulting from multiversioning, we will leverage split-compilation to embed code generation "recipes" to be applied just-in-time, or even at rutime thanks to dynamic binary translation. Finally, we will explore the applicability of auto-tuning, where programmers expose which parameters of their code can be modified to generate alternate versions of the program (for example trading energy consumption for quality of service) and let a global orchestrator make decisions.

2.1.7. Dealing with Attacks – Security

Computer systems are under constant attack, from young hackers trying to show their skills, to "professional" criminals stealing credit card information, and even government agencies with virtually unlimited resources.

A vast amount of techniques have been proposed in the literature to circumvent attacks. Many of them cause significant slowdowns due to additional checks and countermeasures. Thanks to our expertise in micro-architecture and compilation techniques, we will be able to significantly improve efficiency, robustness and coverage of security mechanisms, as well as to partner with field experts to design innovative solutions.

2.1.8. Green Computing – Power Concerns

Power consumption has become a major concern of computing systems, at all form factors, ranging from energy-scavenging sensors for IoT, to battery powered embedded systems and laptops, and up to supercomputers operating in the tens of megawatts. Execution time and energy are often related optimization goals. Optimizing for performance under a given power cap, however, introduces new challenges. It also turns out that technologists introduce new solutions (e.g. magnetic RAM) which, in turn, result in new trade-offs and optimization opportunities.

3. Research Program

3.1. Motivation

Our research program is naturally driven by the evolution of our ecosystem. Relevant recent changes can be classified in the following categories: technological constraints, evolving community, and domain constraints. We hereby summarize these evolutions.

3.1.1. Technological constraints

Until recently, binary compatibility guaranteed portability of programs, while increased clock frequency and improved micro-architecture provided increased performance. However, in the last decade, advances in technology and micro-architecture started translating into more parallelism instead. Technology roadmaps even predict the feasibility of thousands of cores on a chip by 2020. Hundreds are already commercially available. Since the vast majority of applications are still sequential, or contain significant sequential sections, such a trend put an end to the automatic performance improvement enjoyed by developers and users. Many research groups consequently focused on parallel architectures and compiling for parallelism.

Still, the performance of applications will ultimately be driven by the performance of the sequential part. Despite a number of advances (some of them contributed by members of the team), sequential tasks are still a major performance bottleneck. Addressing it is still on the agenda of the PACAP project-team.

In addition, due to power constraints, only part of the billions of transistors of a microprocessor can be operated at any given time (the *dark silicon* paradigm). A sensible approach consists in specializing parts of the silicon area to provide dedicated accelerators (not run simultaneously). This results in diverse and heterogeneous processor cores. Application and compiler designers are thus confronted with a moving target, challenging portability and jeopardizing performance.

Note on technology.
Technology also progresses at a fast pace. We do not propose to pursue any research on technology *per se*. Recently proposed paradigms (non-Silicon, brain-inspired) have received lots of attention from the research community. We do *not* intend to invest in those paradigms, but we will continue to investigate compilation and architecture for more conventional programming paradigms. Still, several technological shifts may have consequences for us, and we will closely monitor their developments. They include for example non-volatile memory (impacts security, makes writes longer than loads), 3D-stacking (impacts bandwidth), and photonics (impacts latencies and connection network), quantum computing (impacts the entire software stack).

3.1.2. Evolving community

The PACAP project-team tackles performance-related issues, for conventional programming paradigms. In fact, programming complex environments is no longer the exclusive domain of experts in compilation and architecture. A large community now develops applications for a wide range of targets, including mobile "apps", cloud, multicore or heterogeneous processors.

This also includes domain scientists (in biology, medicine, but also social sciences) who started relying heavily on computational resources, gathering huge amounts of data, and requiring a considerable amount of processing to analyze them. Our research is motivated by the growing discrepancy between on the one hand, the complexity of the workloads and the computing systems, and on the other hand, the expanding community of developers at large, with limited expertise to optimize and to map efficiently computations to compute nodes.

3.1.3. Domain constraints

Mobile, embedded systems have become ubiquitous. Many of them have real-time constraints. For this class of systems, correctness implies not only producing the correct result, but also doing so within specified deadlines. In the presence of heterogeneous, complex and highly dynamic systems, producing *tight* (i.e., useful) upper bound to the worst-case execution time has become extremely challenging. Our research will aim at improving the tightness as well as enlarging the set of features that can be safely analyzed.

The ever growing dependence of our economy on computing systems also implies that security has become of utmost importance. Many systems are under constant attacks from intruders. Protection has a cost also in terms of performance. We plan to leverage our background to contribute solutions that minimize this impact.

Note on Applications Domains.
PACAP works on fundamental technologies for computer science: processor architecture, performance-oriented compilation and guaranteed response time for real-time. The research results may have impact on any application domain that requires high performance execution (telecommunication, multimedia, biology, health, engineering, environment...), but also on many embedded applications that exhibit other constraints such as power consumption, code size and guaranteed response time.

We strive to extract from active domains the fundamental characteristics that are relevant to our research. For example, *big data* is of interest to PACAP because it relates to the study of hardware/software mechanisms to efficiently transfer huge amounts of data to the computing nodes. Similarly, the *Internet of Things* is of interest because it has implications in terms of ultra low-power consumption.

3.2. Research Objectives

Processor micro-architecture and compilation have been at the core of the research carried by the members of the project teams for two decades, with undeniable contributions. They continue to be the foundation of PACAP.

Heterogeneity and diversity of processor architectures now require new techniques to guarantee that the hardware is satisfactorily exploited by the software. One of our goals is to devise new static compilation techniques (cf. Section 3.2.1), but also build upon iterative [1] and split [40] compilation to continuously adapt software to its environment (Section 3.2.2). Dynamic binary optimization will also play a key role in delivering adapting software and increased performance.

The end of Moore's law and Dennard's scaling [2] offer an exciting window of opportunity, where performance improvements will no longer derive from additional transistor budget or increased clock frequency, but rather come from breakthroughs in micro-architecture (Section 3.2.3). Reconciling CPU and GPU designs (Section 3.2.4) is one of our objectives.

Heterogeneity and multicores are also major obstacles to determining tight worst-case execution times of real-time systems (Section 3.2.5), which we plan to tackle.

Finally, we also describe how we plan to address transversal aspects such as power efficiency (Section 3.2.6), and security (Section 3.2.7).

[2] According to Dennard scaling, as transistors get smaller the power density remains constant, and the consumed power remains proportional to the area.

3.2.1. Static Compilation

Static compilation techniques continue to be relevant in addressing the characteristics of emerging hardware technologies, such as non-volatile memories, 3D-stacking, or novel communication technologies. These techniques expose new characteristics to the software layers. As an example, non-volatile memories typically have asymmetric read-write latencies (writes are much longer than reads) and different power consumption profiles. PACAP studies new optimization opportunities and develops tailored compilation techniques for upcoming compute nodes. New technologies may also be coupled with traditional solutions to offer new trade-offs. We study how programs can adequately exploit the specific features of the proposed heterogeneous compute nodes.

We propose to build upon iterative compilation [1] to explore how applications perform on different configurations. When possible, Pareto points are related to application characteristics. The best configuration, however, may actually depend on runtime information, such as input data, dynamic events, or properties that are available only at runtime. Unfortunately a runtime system has little time and means to determine the best configuration. For these reasons, we also leverage split-compilation [40]: the idea consists in pre-computing alternatives, and embedding in the program enough information to assist and drive a runtime system towards to the best solution.

3.2.2. Software Adaptation

More than ever, software needs to adapt to its environment. In most cases, this environment remains unknown until runtime. This is already the case when one deploys an application to a cloud, or an "app" to mobile devices. The dilemma is the following: for maximum portability, developers should target the most general device; but for performance they would like to exploit the most recent and advanced hardware features. JIT compilers can handle the situation to some extent, but binary deployment requires dynamic binary rewriting. Our work has shown how SIMD instructions can be upgraded from SSE to AVX transparently [2]. Many more opportunities will appear with diverse and heterogeneous processors, featuring various kinds of accelerators.

On shared hardware, the environment is also defined by other applications competing for the same computational resources. It becomes increasingly important to adapt to changing runtime conditions, such as the contention of the cache memories, available bandwidth, or hardware faults. Fortunately, optimizing at runtime is also an opportunity, because this is the first time the program is visible as a whole: executable and libraries (including library versions). Optimizers may also rely on dynamic information, such as actual input data, parameter values, etc. We have already developed a software platform [46] to analyze and optimize programs at runtime, and we started working on automatic dynamic parallelization of sequential code, and dynamic specialization.

We started addressing some of these challenges in ongoing projects such as Nano2017 PSAIC Collaborative research program with STMicroelectronics, as well as within the Inria Project Lab MULTICORE. The H2020 FET HPC project ANTAREX also addresses these challenges from the energy perspective. We further leverage our platform and initial results to address other adaptation opportunities. Efficient software adaptation requires expertise from all domains tackled by PACAP, and strong interaction between all team members is expected.

3.2.3. Research directions in uniprocessor micro-architecture

Achieving high single-thread performance remains a major challenge even in the multicore era (Amdahl's law). The members of the PACAP project-team have been conducting research in uniprocessor micro-architecture research for about 20 years covering major topics including caches, instruction front-end, branch prediction, out-of-order core pipeline, and value prediction. In particular, in recent years they have been recognized as world leaders in branch prediction [50] [44] and in cache prefetching [6] and they have revived the forgotten concept of value prediction [9][8]. This research was supported by the ERC Advanced grant DAL (2011-2016) and also by Intel. We pursue research on achieving ultimate unicore performance. Below are several non-orthogonal directions that we have identified for mid-term research:

1. management of the memory hierarchy (particularly the hardware prefetching);
2. practical design of very wide issue execution cores;

3. speculative execution.

Memory design issues:
Performance of many applications is highly impacted by the memory hierarchy behavior. The interactions between the different components in the memory hierarchy and the out-of-order execution engine have high impact on performance.

The last *Data Prefetching Contest* held with ISCA 2015 has illustrated that achieving high prefetching efficiency is still a challenge for wide-issue superscalar processors, particularly those featuring a very large instruction window. The large instruction window enables an implicit data prefetcher. The interaction between this implicit hardware prefetcher and the explicit hardware prefetcher is still relatively mysterious as illustrated by Pierre Michaud's BO prefetcher (winner of DPC2) [6]. The first research objective is to better understand how the implicit prefetching enabled by the large instruction window interacts with the L2 prefetcher and then to understand how explicit prefetching on the L1 also interacts with the L2 prefetcher.

The second research objective is related to the interaction of prefetching and virtual/physical memory. On real hardware, prefetching is stopped by page frontiers. The interaction between TLB prefetching (and on which level) and cache prefetching must be analyzed.

The prefetcher is not the only actor in the hierarchy that must be carefully controlled. Significant benefits can also be achieved through careful management of memory access bandwidth, particularly the management of spatial locality on memory accesses, both for reads and writes. The exploitation of this locality is traditionally handled in the memory controller. However, it could be better handled if larger temporal granularity was available. Finally, we also intend to continue to explore the promising avenue of compressed caches. In particular we recently proposed the skewed compressed cache [11]. It offers new possibilities for efficient compression schemes.

Ultra wide-issue superscalar.
To effectively leverage memory level parallelism, one requires huge out-of-order execution structures as well as very wide issue superscalar processors. For the two past decades, implementing ever wider issue superscalar processors has been challenging. The objective of our research on the execution core is to explore (and revisit) directions that allow the design of a very wide-issue (8-to-16 way) out-of-order execution core while mastering its complexity (silicon area, hardware logic complexity, power/energy consumption).

The first direction that we are exploring is the use of clustered architectures [7]. Symmetric clustered organization allows to benefit from a simpler bypass network, but induce large complexity on the issue queue. One remarkable finding of our study [7] is that, when considering two large clusters (e.g. 8-wide), steering large groups of consecutive instructions (e.g. 64 μops) to the same cluster is quite efficient. This opens opportunities to limit the complexity of the issue queues (monitoring fewer buses) and register files (fewer ports and physical registers) in the clusters, since not all results have to be forwarded to the other cluster.

The second direction that we are exploring is associated with the approach that we developed with Sembrant et al. [47]. It reduces the number of instructions waiting in the instruction queues for the applications benefiting from very large instruction windows. Instructions are dynamically classified as ready (independent from any long latency instruction) or non-ready, and as urgent (part of a dependency chain leading to a long latency instruction) or non-urgent. Non-ready non-urgent instructions can be delayed until the long latency instruction has been executed; this allows to reduce the pressure on the issue queue. This proposition opens the opportunity to consider an asymmetric micro-architecture with a cluster dedicated to the execution of urgent instructions and a second cluster executing the non-urgent instructions. The micro-architecture of this second cluster could be optimized to reduce complexity and power consumption (smaller instruction queue, less aggressive scheduling...)

Speculative execution.
Out-of-order (OoO) execution relies on speculative execution that requires predictions of all sorts: branch, memory dependency, value...

The PACAP members have been major actors of branch prediction research for the last 20 years; and their proposals have influenced the design of most of the hardware branch predictors in current microprocessors. We will continue to steadily explore new branch predictor designs, as for instance [48].

In speculative execution, we have recently revisited value prediction (VP) which was a hot research topic between 1996 and 2002. However it was considered until recently that value prediction would lead to a huge increase in complexity and power consumption in every stage of the pipeline. Fortunately, we have recently shown that complexity usually introduced by value prediction in the OoO engine can be overcome [9][8] [50] [44]. First, very high accuracy can be enforced at reasonable cost in coverage and minimal complexity [9]. Thus, both prediction validation and recovery by squashing can be done outside the out-of-order engine, at commit time. Furthermore, we propose a new pipeline organization, EOLE ({Early | Out-of-order | Late} Execution), that leverages VP with validation at commit to execute many instructions outside the OoO core, in-order [8]. With EOLE, the issue-width in OoO core can be reduced without sacrificing performance, thus benefiting the performance of VP without a significant cost in silicon area and/or energy. In the near future, we will explore new avenues related to value prediction. These directions include register equality prediction and compatibility of value prediction with weak memory models in multiprocessors.

3.2.4. Towards heterogeneous single-ISA CPU-GPU architectures

Heterogeneous single-ISA architectures have been proposed in the literature during the 2000's [43] and are now widely used in the industry (Arm big.LITTLE, NVIDIA 4+1...) as a way to improve power-efficiency in mobile processors. These architectures include multiple cores whose respective micro-architectures offer different trade-offs between performance and energy efficiency, or between latency and throughput, while offering the same interface to software. Dynamic task migration policies leverage the heterogeneity of the platform by using the most suitable core for each application, or even each phase of processing. However, these works only tune cores by changing their complexity. Energy-optimized cores are either identical cores implemented in a low-power process technology, or simplified in-order superscalar cores, which are far from state-of-the-art throughput-oriented architectures such as GPUs.

We investigate the convergence of CPU and GPU at both architecture and compiler levels.

Architecture.
The architecture convergence between Single Instruction Multiple Threads (SIMT) GPUs and multicore processors that we have been pursuing [42] opens the way for heterogeneous architectures including latency-optimized superscalar cores and throughput-optimized GPU-style cores, which all share the same instruction set. Using SIMT cores in place of superscalar cores will enable the highest energy efficiency on regular sections of applications. As with existing single-ISA heterogeneous architectures, task migration will not necessitate any software rewrite and will accelerate existing applications.

Compilers for emerging heterogeneous architectures.
Single-ISA CPU+GPU architectures will provide the necessary substrate to enable efficient heterogeneous processing. However, it will also introduce substantial challenges at the software and firmware level. Task placement and migration will require advanced policies that leverage both static information at compile time and dynamic information at run-time. We are tackling the heterogeneous task scheduling problem at the compiler level.

3.2.5. Real-time systems

Safety-critical systems (e.g. avionics, medical devices, automotive...) have so far used simple unicore hardware systems as a way to control their predictability, in order to meet timing constraints. Still, many critical embedded systems have increasing demand in computing power, and simple unicore processors are not sufficient anymore. General-purpose multicore processors are not suitable for safety-critical real-time systems, because they include complex micro-architectural elements (cache hierarchies, branch, stride and value predictors) meant to improve average-case performance, and for which worst-case performance is difficult to predict. The prerequisite for calculating tight WCET is a deterministic hardware system that avoids dynamic, time-unpredictable calculations at run-time.

Even for multi and manycore systems designed with time-predictability in mind (Kalray MPPA manycore architecture [3], or the Recore manycore hardware [4]) calculating WCETs is still challenging. The following two challenges will be addressed in the mid-term:

1. definition of methods to estimate WCETs tightly on manycores, that smartly analyze and/or control shared resources such as buses, NoCs or caches;
2. methods to improve the programmability of real-time applications through automatic parallelization and optimizations from model-based designs.

3.2.6. Power efficiency

PACAP addresses power-efficiency at several levels. First, we design static and split compilation techniques to contribute to the race for Exascale computing (the general goal is to reach 10^{18} FLOP/s at less than 20 MW). Second, we focus on high-performance low-power embedded compute nodes. Within the ANR project Continuum, in collaboration with architecture and technology experts from LIRMM and the SME Cortus, we research new static and dynamic compilation techniques that fully exploit emerging memory and NoC technologies. Finally, in collaboration with the CAIRN project-team, we investigate the synergy of reconfigurable computing and dynamic code generation.

Green and heterogeneous high-performance computing.
Concerning HPC systems, our approach consists in mapping, runtime managing and autotuning applications for green and heterogeneous High-Performance Computing systems up to the Exascale level. One key innovation of the proposed approach consists of introducing a separation of concerns (where self-adaptivity and energy efficient strategies are specified aside to application functionalities) promoted by the definition of a Domain Specific Language (DSL) inspired by aspect-oriented programming concepts for heterogeneous systems. The new DSL will be introduced for expressing adaptivity/energy/performance strategies and to enforce at runtime application autotuning and resource and power management. The goal is to support the parallelism, scalability and adaptability of a dynamic workload by exploiting the full system capabilities (including energy management) for emerging large-scale and extreme-scale systems, while reducing the Total Cost of Ownership (TCO) for companies and public organizations.

High-performance low-power embedded compute nodes.
We will address the design of next generation energy-efficient high-performance embedded compute nodes. It focuses at the same time on software, architecture and emerging memory and communication technologies in order to synergistically exploit their corresponding features. The approach of the project is organized around three complementary topics: 1) compilation techniques; 2) multicore architectures; 3) emerging memory and communication technologies. PACAP will focus on the compilation aspects, taking as input the software-visible characteristics of the proposed emerging technology, and making the best possible use of the new features (non-volatility, density, endurance, low-power).

Hardware Accelerated JIT Compilation.
Reconfigurable hardware offers the opportunity to limit power consumption by dynamically adjusting the number of available resources to the requirements of the running software. In particular, VLIW processors can adjust the number of available issue lanes. Unfortunately, changing the processor width often requires recompiling the application, and VLIW processors are highly dependent of the quality of the compilation, mainly because of the instruction scheduling phase performed by the compiler. Another challenge lies in the high constraints of the embedded system: the energy and execution time overhead due to the JIT compilation must be carefully kept under control.

We started exploring ways to reduce the cost of JIT compilation targeting VLIW-based heterogeneous manycore systems. Our approach relies on a hardware/software JIT compiler framework. While basic optimizations and JIT management are performed in software, the compilation back-end is implemented by means of specialized hardware. This back-end involves both instruction scheduling and register allocation, which are known to be the most time-consuming stages of such a compiler.

[3] http://www.kalrayinc.com
[4] http://www.recoresystems.com/

3.2.7. Security

Security is a mandatory concern of any modern computing system. Various threat models have led to a multitude of protection solutions. Members of PACAP already contributed in the past, thanks to the HAVEGE [49] random number generator, and code obfuscating techniques (the obfuscating just-in-time compiler [41], or thread-based control flow mangling [45]). Still, security is not core competence of PACAP members.

Our strategy consists in partnering with security experts who can provide intuition, know-how and expertise, in particular in defining threat models, and assessing the quality of the solutions. Our expertise in compilation and architecture helps design more efficient and less expensive protection mechanisms.

Examples of collaborations so far include the following:

Compilation: We partnered with experts in security and codes to prototype a platform that demonstrates resilient software. They designed and proposed advanced masking techniques to hide sensitive data in application memory. PACAP's expertise is key to select and tune the protection mechanisms developed within the project, and to propose safe, yet cost-effective solutions from an implementation point of view.

Dynamic Binary Rewriting: Our expertise in dynamic binary rewriting combines well with the expertise of the CIDRE team in protecting application. Security has a high cost in terms of performance, and static insertion of counter measures cannot take into account the current threat level. In collaboration with CIDRE, we propose an adaptive insertion/removal of countermeasures in a running application based of dynamic assessment of the threat level.

WCET Analysis: Designing real-time systems requires computing an upper bound of the worst-case execution time. Knowledge of this timing information opens an opportunity to detect attacks on the control flow of programs. In collaboration with CIDRE, we are developing a technique to detect such attacks thanks to a hardware monitor that makes sure that statically computed time information is preserved (CAIRN is also involved in the definition of the hardware component).

4. Application Domains

4.1. Domains

The PACAP team is working on the fundamental technologies for computer science: processor architecture, performance-oriented compilation and guaranteed response time for real-time. The research results may have impact on any application domain that requires high performance execution (telecommunication, multimedia, biology, health, engineering, environment...), but also on many embedded applications that exhibit other constraints such as power consumption, code size and guaranteed response time. Our research activity implies the development of software prototypes.

5. Highlights of the Year

5.1. Highlights of the Year

5.1.1. Awards

Benjamin Rouxel, Stefanos Skalistis, Steven Derrien and Isabelle Puaut received an Outstanding paper award for their paper entitled “Hiding Communication Delays in Contention-Free Execution for SPM-based Multi-Core Architectures” at the Euromicro conference on real time systems .

BEST PAPER AWARD:

[28]

B. ROUXEL, S. SKALISTIS, S. DERRIEN, I. PUAUT. *Hiding Communication Delays in Contention-Free Execution for SPM-Based Multi-Core Architectures*, in "ECRTS 2019 - 31st Euromicro Conference on Real-Time Systems", Stuttgart, Germany, July 2019, pp. 1-24 [*DOI :* 10.4230/LIPIcs.ECRTS.2019.25], https://hal.archives-ouvertes.fr/hal-02190271

6. New Software and Platforms

6.1. ATMI

KEYWORDS: Analytic model - Chip design - Temperature

SCIENTIFIC DESCRIPTION: Research on temperature-aware computer architecture requires a chip temperature model. General purpose models based on classical numerical methods like finite differences or finite elements are not appropriate for such research, because they are generally too slow for modeling the time-varying thermal behavior of a processing chip.

ATMI (Analytical model of Temperature in MIcroprocessors) is an ad hoc temperature model for studying thermal behaviors over a time scale ranging from microseconds to several minutes. ATMI is based on an explicit solution to the heat equation and on the principle of superposition. ATMI can model any power density map that can be described as a superposition of rectangle sources, which is appropriate for modeling the microarchitectural units of a microprocessor.

FUNCTIONAL DESCRIPTION: ATMI is a library for modelling steady-state and time-varying temperature in microprocessors. ATMI uses a simplified representation of microprocessor packaging.

- Participant: Pierre Michaud
- Contact: Pierre Michaud
- URL: https://team.inria.fr/pacap/software/atmi/

6.2. HEPTANE

KEYWORDS: IPET - WCET - Performance - Real time - Static analysis - Worst Case Execution Time

SCIENTIFIC DESCRIPTION: WCET estimation

The aim of Heptane is to produce upper bounds of the execution times of applications. It is targeted at applications with hard real-time requirements (automotive, railway, aerospace domains). Heptane computes WCETs using static analysis at the binary code level. It includes static analyses of microarchitectural elements such as caches and cache hierarchies.

FUNCTIONAL DESCRIPTION: In a hard real-time system, it is essential to comply with timing constraints, and Worst Case Execution Time (WCET) in particular. Timing analysis is performed at two levels: analysis of the WCET for each task in isolation taking account of the hardware architecture, and schedulability analysis of all the tasks in the system. Heptane is a static WCET analyser designed to address the first issue.

- Participants: Benjamin Lesage, Loïc Besnard, Damien Hardy, François Joulaud, Isabelle Puaut and Thomas Piquet
- Partner: Université de Rennes 1
- Contact: Isabelle Puaut
- URL: https://team.inria.fr/pacap/software/heptane/

6.3. tiptop

KEYWORDS: Instructions - Cycles - Cache - CPU - Performance - HPC - Branch predictor

SCIENTIFIC DESCRIPTION: Tiptop is a new simple and flexible user-level tool that collects hardware counter data on Linux platforms (version 2.6.31+) and displays them in a way simple to the Linux "top" utility. The goal is to make the collection of performance and bottleneck data as simple as possible, including simple installation and usage. No privilege is required, any user can run tiptop.

Tiptop is written in C. It can take advantage of libncurses when available for pseudo-graphic display. Installation is only a matter of compiling the source code. No patching of the Linux kernel is needed, and no special-purpose module needs to be loaded.

Current version is 2.3.1, released October 2017. Tiptop has been integrated in major Linux distributions, such as Fedora, Debian, Ubuntu, CentOS.

FUNCTIONAL DESCRIPTION: Today's microprocessors have become extremely complex. To better understand the multitude of internal events, manufacturers have integrated many monitoring counters. Tiptop can be used to collect and display the values from these performance counters very easily. Tiptop may be of interest to anyone who wants to optimise the performance of their HPC applications.

- Participant: Erven Rohou
- Contact: Erven Rohou
- URL: http://tiptop.gforge.inria.fr

6.4. PADRONE

KEYWORDS: Legacy code - Optimization - Performance analysis - Dynamic Optimization

FUNCTIONAL DESCRIPTION: Padrone is new platform for dynamic binary analysis and optimization. It provides an API to help clients design and develop analysis and optimization tools for binary executables. Padrone attaches to running applications, only needing the executable binary in memory. No source code or debug information is needed. No application restart is needed either. This is especially interesting for legacy or commercial applications, but also in the context of cloud deployment, where actual hardware is unknown, and other applications competing for hardware resources can vary. The profiling overhead is minimum.

- Participants: Emmanuel Riou and Erven Rohou
- Contact: Erven Rohou
- URL: https://team.inria.fr/pacap/software/padrone

6.5. If-memo

KEYWORD: Performance

SCIENTIFIC DESCRIPTION: We propose a linker based technique for enabling software memorizing of any dynamically linked pure function by function interception and we illustrate our framework using a set of computationally expensive pure functions - the transcendental functions.

FUNCTIONAL DESCRIPTION: If-memo is a linker-based technique for enabling software memorizing of any dynamically linked pure function by function interception. Typically, this framework is useful to intercept the computationally expensive pure functions - the transcendental functions from the math library. Our technique does not need the availability of source code and thus can even be applied to commercial applications as well as applications with legacy codes. As far as users are concerned, enabling memoization is as simple as setting an environment variable. Our framework does not make any specific assumptions about the underlying architecture or compiler too-chains, and can work with a variety of current architectures.

- Participants: Arjun Suresh and Erven Rohou
- Contact: Erven Rohou
- URL: https://team.inria.fr/pacap/software/if-memo/

6.6. Simty

KEYWORDS: GPU - Softcore - FPGA - SIMT - Multi-threading - RISC-V

FUNCTIONAL DESCRIPTION: Simty is a massively multi-threaded processor core that dynamically assembles SIMD instructions from scalar multi-thread code. It runs the RISC-V (RV32-I) instruction set. Unlike existing SIMD or SIMT processors like GPUs, Simty takes binaries compiled for general-purpose processors without any instruction set extension or compiler changes. Simty is described in synthesizable VHDL.

- Author: Caroline Collange
- Contact: Caroline Collange
- URL: https://gforge.inria.fr/projects/simty

6.7. Barra

KEYWORDS: GPU - GPGPU - Tesla ISA - Debug - Computer architecture - Performance - Profiling - Simulator - HPC - CUDA

SCIENTIFIC DESCRIPTION: Research on throughput-oriented architectures demands accurate and representative models of GPU architectures in order to be able to evaluate new architectural ideas, explore design spaces and characterize applications. The Barra project is a simulator of the NVIDIA Tesla GPU architecture.

Barra builds upon knowledge acquired through micro-benchmarking, in order to provide a baseline model representative of industry practice. The simulator provides detailed statistics to identify optimization opportunities and is fully customizable to experiment ideas of architectural modifications. Barra incorporates both a functional model and a cycle-level performance model.

FUNCTIONAL DESCRIPTION: Barra is a Graphics Processing Unit (GPU) architecture simulator. It simulates NVIDIA CUDA programs at the assembly language level. Barra is a tool for research on computer architecture, and can also be used to debug, profile and optimize CUDA programs at the lowest level.

RELEASE FUNCTIONAL DESCRIPTION: Version 0.5.10 introduces: Timing model, Tesla-like architecture model, Fermi-like architecture model, New per-PC control-flow divergence management, Support for Simultaneous branch and warp interweaving, Support for Affine vector cache.

- Participants: Alexandre Kouyoumdjian, David Defour, Fabrice Mouhartem and Caroline Collange
- Partners: ENS Lyon - UPVD
- Contact: Caroline Collange
- URL: http://barra.gforge.inria.fr/

6.8. Memoization

KEYWORDS: Optimization - Pure function - Memoization

FUNCTIONAL DESCRIPTION: Memoization is a technique used at runtime that consists in caching results of pure functions and retrieving them instead of computing them when the arguments repeat. It can be applied to C and C++ programs. To be memoized, the interface of a pure function (or a method) must verify the following properties: (1) the function/method has at most four arguments of same type T, (2) the function/method returns a data of type T, (3) T is either 'double', 'float', or 'int'.

The memoization operation of a function/method is controlled by several parameters: the size of the internal table (number of entries), the replacement policy to be used in case of index conflict (whether the value of the table must be replaced or not), an approximation threshold that allows to not distinguish very close values). It is also possible to initialize the table with the content of a file, and to save the content of the table to a file at the end of the execution (the data may be used as input for a future execution).

- Participants: Loïc Besnard, Imane Lasri and Erven Rohou
- Contact: Loïc Besnard

6.9. FiPlib

KEYWORDS: Compilation - Approximate computing - Fixed-point representation

FUNCTIONAL DESCRIPTION: FiPlib is a C++ library that provides type definition and conversion operations for computations in fixed-point representation. Basic arithmetic as well as logical operations are transparently supported thanks to operator overloading. FiPlib also provides optimized implementations of the transcendental math functions of libm. For convenient integration, FiPlib is released as C++ header files only. Optionally, FiPlib can detect overflows and compute errors compared to floating point representation.

- Participants: Pierre Le Meur, Imane Lasri and Erven Rohou
- Contact: Erven Rohou

6.10. sigmask

KEYWORDS: Compilation - Side-channel - Masking - Security - Embedded systems

SCIENTIFIC DESCRIPTION: Sigmask is a compiler plugin based on the LLVM infrastructure that automatically protects secret information in programs, such as encryption keys, against side-channel attacks. The programmer annotates their source code to highlight variables containing sensitive data. The compiler automatically analyzes the program and computes all memory locations potentially derived from the secret. It then applies a masking scheme to avoid information leakage. Sigmask provides several schemes: OSDM (Orthogonal Direct Sum Masking), IP (Inner Product) Masking, and simple random bit masking. The programmer may also provide their own masking scheme through a well-defined API.

FUNCTIONAL DESCRIPTION: Sigmask is a compiler plugin based on the LLVM infrastructure that automatically protects secret information in programs, such as encryption keys, against side-channel attacks. The programmer annotates their source code to highlight variables containing sensitive data. The compiler automatically analyzes the program and computes all memory locations potentially derived from the secret. It then applies a masking scheme to avoid information leakage. Sigmask provides several schemes: ODSM (Orthogonal Direct Sum Masking), IP (Inner Product) Masking, and simple random bit masking. The programmer may also provide their own masking scheme through a well-defined API.

- Participants: Nicolas Kiss, Damien Hardy and Erven Rohou
- Contact: Erven Rohou

7. New Results

7.1. Compilation and Optimization

Participants: Loïc Besnard, Caroline Collange, Byron Hawkins, Erven Rohou, Bahram Yarahmadi.

7.1.1. Optimization in the Presence of NVRAM

Participants: Erven Rohou, Bahram Yarahmadi.

A large and increasing number of Internet-of-Things devices are not equipped with batteries and harvest energy from their environment. Many of them cannot be physically accessed once they are deployed (embedded in civil engineering structures, sent in the atmosphere or deep in the oceans). When they run out of energy, they stop executing and wait until the energy level reaches a threshold. Programming such devices is challenging in terms of ensuring memory consistency and guaranteeing forward progress.

7.1.1.1. Checkpoint Placement based Worst-Case Energy Consumption

Previous work has proposed to insert checkpoints in the program so that execution can resume from well-defined locations. We propose to define these checkpoint locations based on worst-case energy consumption of code sections, with limited additional effort for programmers. As our method is based upon worst-case energy consumption, we can guarantee memory consistency and forward progress.

This work has been presented at the Compas 2019 conference.

7.1.1.2. Dynamic Adaptive Checkpoint Placement

Previous work has proposed to back-up the volatile states which are necessary for resuming the program execution after power failures. They either do it at compile time by placing checkpoints into the control flow of the program or at runtime by leveraging voltage monitoring facilities and interrupts, so that execution can resume from well-defined locations after power failures. We propose for the first time a dynamic checkpoint placement strategy which delays checkpoint placement and specialization to the runtime and takes decisions based on the past power failures and execution paths that are taken. We evaluate our work on a TI MSP430 device, with different types of benchmarks as well as different uninterrupted intervals, and we measure the execution time. We show that our work can outperform compiler-based state-of-the-art with memory footprint kept under the control.

This research is done within the context of the project IPL ZEP.

7.1.2. Dynamic Binary Optimization

Participant: Erven Rohou.

7.1.2.1. Guided just-in-time specialization

JavaScript's portability across a vast ecosystem of browsers makes it today a core building block of the web. Yet, building efficient systems in JavaScript is still challenging. Because this language is so dynamic, JavaScript programs provide little information that just-in-time compilers can use to carry out safe optimizations. Motivated by this observation, we propose to guide the JIT compiler in the task of code specialization. To this end, we have augmented [17] the language with an annotation that indicates which function call sites are likely to benefit from specialization. To support the automatic annotation of programs, we have introduced a novel static analysis that identifies profitable specialization points. We have implemented our ideas in JavaScriptCore, the built-in JavaScript engine for WebKit. The addition of guided specialization to this engine required us to change it in several non-trivial ways. Such changes let us observe speedups of up to 1.7$\times$ on programs present in synthetic benchmarks.

7.1.2.2. Run-time parallelization and de-parallelization

Runtime compilation has opportunities to parallelize code which are generally not available using static parallelization approaches. However, the parallelized code can possibly slowdown the performance due to unforeseen parallel overheads such as synchronization and speculation support pertaining to the chosen parallelization strategy and the underlying parallel platform. Moreover, with the wide usage of heterogeneous architectures, such choice options become more pronounced. We consider [22] an adaptive form of the parallelization operation, for the first time. We propose a method for performing on-stack de-parallelization for a parallelized binary loop at runtime, thereby allowing for rapid loop replacement with a more optimized one. We consider a loop parallelization strategy and propose a corresponding de-parallelization method. The method relies on stopping the execution at safe points, gathering threads' states, producing a corresponding serial code, and continuing execution serially. The decision to de-parallelize or not is taken based on the anticipated speedup. To assess the extent of our approach, we have conducted an initial study on a small set of programs with various parallelization overheads. Results show up to 4$\times$ performance improvement for a synchronization intense program on a 4-core Intel processor.

With the multicore trend, the need for automatic parallelization is more pronounced, especially for legacy and proprietary code where no source code is available and/or the code is already running and restarting is not an option. We engineer [21] a mechanism for transforming at runtime a frequent for-loop with no data dependencies in a binary program into a parallel loop, using on-stack replacement. With our mechanism, there is no need for source code, debugging information or restarting the program. Also, the mechanism needs no static instrumentation or information. The mechanism is implemented using the Padrone binary modification system and `pthreads`, where the remaining iterations of the loop are executed in parallel. The mechanism keeps the running program state by extracting the targeted loop into a separate function and copying the current stack frame into the corresponding frames of the created threads. Initial study is conducted on a set of kernels from the Polybench workload. Experimental results show from 2× to 3.5× speedup from sequential to parallelized code on four cores, which is similar to source code level parallelization.

This research was partially done within the context of the project PHC IMHOTEP.

7.1.3. Automatic and Parametrizable Memoization

Participants: Loïc Besnard, Erven Rohou.

Improving execution time and energy efficiency is needed for many applications and usually requires sophisticated code transformations and compiler optimizations. One of the optimization techniques is memoization, which saves the results of computations so that future computations with the same inputs can be avoided. We propose [16] a framework that automatically applies memoization techniques to C/C++ applications. The framework is based on automatic code transformations using a source-to-source compiler and on a memoization library. With the framework users can select functions to memoize as long as they obey to certain restrictions imposed by our current memoization library. We show the use of the framework and associated memoization technique and the impact on reducing the execution time and energy consumption of four representative benchmarks. The support library is available at https://gforge.inria.fr/projects/memoization (registered with APP under number IDDN.FR.001.250029.000.S.P.2018.000.10800).

7.1.4. Autotuning

Participants: Loïc Besnard, Erven Rohou.

The ANTAREX FET HPC project relies on a Domain Specific Language (DSL) based on Aspect Oriented Programming (AOP) concepts to allow applications to enforce extra functional properties such as energy-efficiency and performance and to optimize Quality of Service (QoS) in an adaptive way. The DSL approach allows the definition of energy-efficiency, performance, and adaptivity strategies as well as their enforcement at runtime through application autotuning and resource and power management. We present [20] an overview of the key outcome of the project, the ANTAREX DSL, and some of its capabilities through a number of examples, including how the DSL is applied in the context of the project use cases. We demonstrated [30] tools and techniques in two domains: computational drug discovery, and online vehicle navigation.

7.1.5. Loop splitting

The loop splitting technique takes advantage of long running loops to explore the impact of several optimization sequences at once, thus reducing the number of necessary runs. We rely on a variant of loop peeling which splits a loop into into several loops, with the same body, but a subset of the iteration space. New loops execute consecutive chunks of the original loop. We then apply different optimization sequences on each loop independently. Timers around each chunk observe the performance of each fragment. This technique may be generalized to combine compiler options and different implementations of a function called in a loop. It is useful when, for example, the profiling of the application shows that a function is critical in term of time of execution. In this case, the user must try to find the best implementation of their algorithm.

This research was partially done within the context of the ANTAREX FET HPC collaborative project, collaboration is currently ongoing with University of Porto, Portugal.

7.1.6. Hardware/Software JIT Compiler

Participant: Erven Rohou.

Single-ISA heterogeneous systems (such as ARM big.LITTLE) are an attractive solution for embedded platforms as they expose performance/energy trade-offs directly to the operating system. Recent works have demonstrated the ability to increase their efficiency by using VLIW cores, supported through Dynamic Binary Translation (DBT) to maintain the illusion of a single-ISA system. However, VLIW cores cannot rival with Out-of-Order (OoO) cores when it comes to performance, mainly because they do not use speculative execution. We study [27] how it is possible to use memory dependency speculation during the DBT process. Our approach enables fine-grained speculation optimizations thanks to a combination of hardware and software. Our results show that our approach leads to a geo-mean speed-up of 10 % at the price of a 7 % area overhead.

Our previous work on Hybrid-DBT was also presented at the RISC-V workshop in Zürich, Switzerland [38].

This work is a collaboration with the CAIRN team.

7.1.7. Scalable program tracing

Participants: Byron Hawkins, Erven Rohou.

The initial goal of scalable tracing is to record long executions at under 5× overhead (ideally 2×), but it is equally important for analysis of the compressed trace to be efficient. This requires careful organization of the recorded data structures so that essential factors can be accessed without decompressing the trace or comprehensively iterating its paths. Precise context sensitivity is especially important for both optimization and security applications of trace-based program analysis, but scalability becomes challenging for frequently invoked functions that have a high degree of internal complexity. To avoid state space explosion in the context graph, such a function can be represented as a singleton while its complexity is preserved orthogonally. The current efforts focus mainly on developing an integration strategy to simplify program analysis over these two orthogonal dimensions of the trace.

7.1.8. Compiler optimization for quantum architectures

Participant: Caroline Collange.

In 2016, the first quantum processors have been made available to the general public. The possibility of programming an actual quantum device has elicited much enthusiasm [34]. Yet, such possibility also brought challenges. One challenge is the so called Qubit Allocation problem: the mapping of a virtual quantum circuit into an actual quantum architecture. There exist solutions to this problem; however, in our opinion, they fail to capitalize on decades of improvements on graph theory.

In collaboration with the Federal University of Minas Gerais, Brazil, we show how to model qubit allocation as the combination of Subgraph Isomorphism and Token Swapping [31]. This idea has been made possible by the publication of an approximative solution to the latter problem in 2016. We have compared our algorithm against five other qubit allocators, all independently designed in the last two years, including the winner of the IBM Challenge. When evaluated in “Tokyo”, a quantum architecture with 20 qubits, our technique outperforms these state-of-the-art approaches in terms of the quality of the solutions that it finds and the amount of memory that it uses, while showing practical runtime.

7.2. Processor Architecture

Participants: Arthur Blanleuil, Niloofar Charmchi, Caroline Collange, Kleovoulos Kalaitzidis, Pierre Michaud, Anis Peysieux, Daniel Rodrigues Carvalho, André Seznec.

7.2.1. Value prediction

Participants: Kleovoulos Kalaitzidis, André Seznec.

Modern context-based value predictors tightly associate recurring values with instructions and contexts by building confidence upon them [9]. However, when execution monotony exists in the form of intervals, the potential prediction coverage is limited, since prediction confidence is reset at the beginning of each new interval. In [25], we address this challenge by introducing the notion of Equality Prediction (EP), which represents the binary facet of value prediction. Following a two fold decision scheme (similar to branch prediction), EP makes use of control-flow history to determine equality between the last committed result read at fetch time, and the result of the fetched occurrence. When equality is predicted with high confidence, the read value is used. Our experiments show that this technique obtains the same level of performance as previously proposed state-of-the-art context-based predictors. However, by virtue of better exploiting patterns of interval equality, our design complements the established way that value prediction is performed, and when combined with contemporary prediction models, improves the delivered speedup by 19 % on average.

7.2.2. Compressed caches

Participants: Daniel Rodrigues Carvalho, Niloofar Charmchi, Caroline Collange, André Seznec.

The speed gap between CPU and memory is impairing performance. Cache compression and hardware prefetching are two techniques that could confront this bottleneck by decreasing last level cache misses. However, compression and prefetching have positive interactions, as prefetching benefits from higher cache capacity and compression increases the effective cache size. We propose Compressed cache Layout Aware Prefetching (CLAP) to leverage the recently proposed sector-based compressed cache layouts such as SCC or YACC to create a synergy between compressed cache and prefetching. The idea of this approach is to prefetch contiguous blocks that can be compressed and co-allocated together with the requested block on a miss access [33]. Prefetched blocks that share storage with existing blocks do not need to evict a valid existing entry; therefore, CLAP avoids cache pollution. In order to decide the co-allocatable blocks to prefetch, we propose a compression predictor. Based on our experimental evaluations, CLAP reduces the number of cache misses by 12 % and improves performance by 4 % on average, comparing to a compressed cache [23].

7.2.3. *Deep microarchitecture*

Participants: Anis Peysieux, André Seznec.

The design of an efficient out-of-order execution core is particularly challenging. When the issue-width increases, the cost of the extra logic required by out-of-core execution increases dramatically. The silicon area occupied by this OoO core tends to grow quasi-quadratically with the issue-width (e.g. issue logic, register file and result bypass). At the same time, the power requirement and the energy consumption of the out-of–order core grow super-linearly with issue width. On wide-issue out-of-order execution cores, issue logic response time, register file access time, as well as result bypass delays represent potential critical paths that might impair cycle time or might necessitate further deepening of the execution pipeline. The objective of the PhD thesis of Anis Peysieux will be to reduce the number of instructions that enter the OoO core, and therefore to master the hardware complexity while still achieving the performance promises of a very wide issue processor.

7.2.4. *Dynamic thermal management*

Participant: Pierre Michaud.

As power dissipation and circuit temperature constrain their performance, modern processors feature turbo control mechanisms to adjust the voltage and clock frequency dynamically so that circuit temperature stays below a certain limit. In particular, turbo control exploits the fact that, after a long period of low processor activity, the thermal capacity of the chip, its package and the heatsink can absorb heat at a relatively fast rate during a certain time, before the temperature limit constrains that rate. Hence power dissipation can be temporarily boosted above the average sustainable value. The turbo control must monitor circuit temperature continuously to maximize the clock frequency. Temperature can be monitored by reading the integrated thermal sensors. However, making the clock frequency depend on thermal sensor readings implies that processor performance depends on ambient temperature. Yet this form of performance non-determinism is a problem for certain processor makers. A possible solution is to determine the clock frequency not from the true temperature but from a thermal model based on the nominal ambient temperature. Such model should be as accurate as possible in order to prevent sensor-based protection from triggering but sporadically, without hurting performance by overestimating temperature too much. The model should also be simple enough to provide calculated temperature in real time. We propose a thermal model possessing these qualities, and a new turbo control algorithm based on that model [37].

7.2.5. *Thread convergence prediction for general-purpose SIMT architectures*

Participants: Arthur Blanleuil, Caroline Collange.

GPUs group threads of SPMD programs in warps and synchronize them to execute the same instruction at the same time. This execution model, referred to as Single-Instruction, Multiple-Thread (SIMT), enables the use of energy-efficient SIMD execution units by factoring out control logic such as instruction fetch and decode pipeline stages for a whole warp. SIMT execution is the key enabler for the energy efficiency of GPUs. We seek to generalize the SIMT execution model to general-purpose superscalar cores.

As threads within a warp may follow different directions through conditional branches in the program, the warp must follow each taken path in turn, while disabling individual threads that do not participate. Following divergence, current GPU architectures attempt to restore convergence at the earliest program point following static annotations in the binary. However, this policy has been shown to be suboptimal in many cases, in which later convergence improves performance. In fact, optimal convergence points depend on dynamic program behavior, so static decisions are unable to capture them.

The goal of the thesis of Arthur Blanleuil is to design predictors that enable the microarchitecture to infer dynamic code behavior and place convergence points appropriately. Convergence predictors have analogies with branch predictors and control independence predictors studied in superscalar processor architecture, but they present one additional challenge: the thread runaway problem. Although a branch misprediction will be identified and repaired locally, a wrong thread scheduling decision may go unnoticed and delay convergence by thousands of instructions. To address the thread runaway problem, we plan to explore promise-based speculation and recovery strategies. When no information is available, we follow the traditional conservative earliest-convergence scheduling policy. Once the predictor has enough information to make a more aggressive prediction, it generates assumptions about the prediction. The microarchitecture then keeps checking dynamically whether the assumptions actually hold true in the near future. If assumptions turn out to be wrong, the prediction will be reconsidered by changing back priorities to conservative. Such promise-based speculation policies can address the thread runaway problem by fixing a bound on the worst-case performance degradation of an aggressive scheduling policy against the conservative baseline.

Accurate thread convergence policies will enable dynamic vectorization to adapt to application characteristics dynamically. They will both improve performance and simplify programming of many-core architectures by alleviating the need for advanced code tuning by expert programmers.

7.2.6. Exploring the design space of GPU architectures

Participants: Alexandre Kouyoumdjian, Caroline Collange.

We study tradeoffs in the internal organization of GPUs in the context of general-purpose parallel processing [35]. In particular, we analyze the performance impact of having a few wide streaming multiprocessors compared to many narrow ones. Although we find narrow configurations usually give higher performance for an equal number of execution units, they require more hardware resources and energy. On the other hand, our evaluation show that the optimal streaming multiprocessor width varies across applications. This study motivates adaptive GPU architectures that would support configurable internal organization.

7.3. WCET estimation and optimization

Participants: Loïc Besnard, Damien Hardy, Isabelle Puaut, Stefanos Skalistis.

7.3.1. WCET estimation for many core processors

Participants: Damien Hardy, Isabelle Puaut, Stefanos Skalistis.

7.3.1.1. Optimization of WCETs by considering the effects of local caches

The overall goal of this research is to define WCET estimation methods for parallel applications running on many-core architectures, such as the Kalray MPPA machine. Some approaches to reach this goal have been proposed, but they assume the mapping of parallel applications on cores is already done. Unfortunately, on architectures with caches, task mapping requires a priori known WCETs for tasks, which in turn requires knowing task mapping (i.e., co-located tasks, co-running tasks) to have tight WCET bounds. Therefore, scheduling parallel applications and estimating their WCET introduce a chicken-and-egg situation.

We addressed this issue by developing both optimal and heuristic techniques for solving the scheduling problem, whose objective is to minimize the WCET of a parallel application. Our proposed static partitioned non-preemptive mapping strategies address the effect of local caches to tighten the estimated WCET of the parallel application. Experimental results obtained on real and synthetic parallel applications show that co-locating tasks that reuse code and data improves the WCET by 11 % on average for the optimal method and by 9 % on average for the heuristic method. An implementation on the Kalray MPPA machine allowed to identify implementation-related overheads. All results are described in [18].

7.3.1.2. Shared resource contentions and WCET estimation

Accurate WCET analysis for multi-cores is known to be challenging, because of concurrent accesses to shared resources, such as communication through busses or Networks on Chips (NoC). Since it is impossible in general to guarantee the absence of resource conflicts during execution, current WCET techniques either produce pessimistic WCET estimates or constrain the execution to enforce the absence of conflicts, at the price of a significant hardware under-utilization. In addition, the large majority of existing works consider that the platform workload consists of independent tasks. As parallel programming is the most promising solution to improve performance, we envision that within only a few years from now, real-time workloads will evolve toward parallel programs. The WCET behavior of such programs is challenging to analyze because they consist of *dependent* tasks interacting through complex synchronization/communication mechanisms.

In [28], we propose a scheduling technique that jointly selects Scratchpad Memory (SPM) contents off-line, in such a way that the cost of SPM loading/unloading is hidden. Communications are fragmented to augment hiding possibilities. Experimental results show the effectiveness of the proposed technique on streaming applications and synthetic task-graphs. The overlapping of communications with computations allows the length of generated schedules to be reduced by 4 % on average on streaming applications, with a maximum of 16 %, and by 8 % on average for synthetic task graphs. We further show on a case study that generated schedules can be implemented with low overhead on a predictable multi-core architecture (Kalray MPPA).

7.3.1.3. Interference-sensitive run-time adaptation of time-triggered schedules

In time-critical systems, run-time adaptation is required to improve the performance of time-triggered execution, derived based on Worst-Case Execution Time (WCET) of tasks. By improving performance, the systems can provide higher Quality-of-Service, in safety-critical systems, or execute other best-effort applications, in mixed-critical systems. To achieve this goal, we propose in [32] a parallel interference-sensitive run-time adaptation mechanism that enables a fine-grained synchronisation among cores. Since the run-time adaptation of offline solutions can potentially violate the timing guarantees, we present the Response-Time Analysis (RTA) of the proposed mechanism showing that the system execution is free of timing-anomalies. The RTA takes into account the timing behavior of the proposed mechanism and its associated WCET. To support our contribution, we evaluate the behavior and the scalability of the proposed approach for different application types and execution configurations on the 8-core Texas Instruments TMS320C6678 platform. The obtained results show significant performance improvement compared to state-of-the-art centralized approaches.

7.3.1.4. WCET-Aware Parallelization of Model-Based Applications for Multi-Cores

Parallel architectures are nowadays not only confined to the domain of high performance computing, they are also increasingly used in embedded time-critical systems.

The Argo H2020 project provides a programming paradigm and associated tool flow to exploit the full potential of architectures in terms of development productivity, time-to-market, exploitation of the platform computing power and guaranteed real-time performance. The Argo toolchain operates on Scilab and XCoS inputs, and targets ScratchPad Memory (SPM)-based multi-cores. Data-layout and loop transformations play a key role in this flow as they improve SPM efficiency and reduce the number of accesses to shared main memory.

In [19] we present the overall results of the project, a compiler tool-flow for automated parallelization of model-based real-time software, which addresses the shortcomings of multi-core architectures in real-time systems. The flow is demonstrated using a model-based Terrain Awareness and Warning Systems (TAWS) and an edge detection algorithm from the image-processing domain. Model-based applications are first transformed into real-time C code and from there into a well-predictable parallel C program. Tight bounds for the Worst-Case Execution Time (WCET) of the parallelized program can be determined using an integrated multi-core WCET analysis. Thanks to the use of an architecture description language, the general approach is applicable to a wider range of target platforms. An experimental evaluation for a research architecture with network-on-chip (NoC) interconnect shows that the parallel WCET of the TAWS application can be improved by factor 1.77 using the presented compiler tools.

7.3.2. WCET estimation and optimizing compilers

Participants: Isabelle Puaut, Stefanos Skalistis.

Static Worst-Case Execution Time (WCET) estimation techniques operate upon the binary code of a program in order to provide the necessary input for schedulability analysis techniques. Compilers used to generate this binary code include tens of optimizations, that can radically change the flow information of the program. Such information is hard to be maintained across optimization passes and may render automatic extraction of important flow information, such as loop bounds, impossible. Thus, compiler optimizations, especially the sophisticated optimizations of mainstream compilers, are typically avoided. We explore [24] for the first time iterative-compilation techniques that reconcile compiler optimizations and static WCET estimation. We propose a novel learning technique that selects sequences of optimizations that minimize the WCET estimate of a given program. We experimentally evaluate the proposed technique using an industrial WCET estimation tool (AbsInt aiT) over a set of 46 benchmarks from four different benchmarks suites, including reference WCET benchmark applications, image processing kernels and telecommunication applications. Experimental results show that WCET estimates are reduced on average by 20.3 % using the proposed technique, as compared to the best compiler optimization level applicable.

7.3.3. WCET estimation and processor micro-architecture

Participant: Isabelle Puaut.

Cache memories in modern embedded processors are known to improve average memory access performance. Unfortunately, they are also known to represent a major source of unpredictability for hard real-time workload. One of the main limitations of typical caches is that content selection and replacement is entirely performed in hardware. As such, it is hard to control the cache behavior in software to favor caching of blocks that are known to have an impact on an application's worst-case execution time (WCET). In [26], we consider a cache replacement policy, namely DM-LRU, that allows system designers to prioritize caching of memory blocks that are known to have an important impact on an application's WCET. Considering a single-core, single-level cache hierarchy, we describe an abstract interpretation-based timing analysis for DM-LRU. We implement the proposed analysis in a self-contained toolkit and study its qualitative properties on a set of representative benchmarks. Apart from being useful to compute the WCET when DM-LRU or similar policies are used, the proposed analysis can allow designers to perform WCET impact-aware selection of content to be retained in cache.

Long pipelines need good branch predictors to keep the pipeline running. Current branch predictors are optimized for the average case, which might not be a good fit for real-time systems and worst-case execution time analysis. We present [29] a time-predictable branch predictor co-designed with the associated worst-case execution time analysis. Thee branch predictor uses a fully-associative cache to track branch outcomes and destination addresses. The fully-associative cache avoids any false sharing of entries between branches. Therefore, we can analyze program scopes that contain a number of branches lower than or equal to the number of branches in the prediction table. Experimental results show that the worst-case execution time bounds of programs using the proposed predictor are lower than using static branch predictors at a moderate hardware cost.

7.4. Security

Participants: Nicolas Bellec, Damien Hardy, Kévin Le Bon, Isabelle Puaut, Erven Rohou.

7.4.1. Attack detection co-processor for real-time systems

Participants: Nicolas Bellec, Isabelle Puaut.

Real-time embedded systems (RTES) are required to interact more and more with their environment, thereby increasing their attack surface. Recent security breaches on car brakes and other critical components, have already proven the feasibility of attacks on RTES. Such attacks may change the control-flow of the programs, which may lead to violations of the timing constraints of the system. In this ongoing work, we design a technique to detect attacks in RTES based on timing information. Our technique is based on a monitor, implemented in hardware to preserve the predictability of instrumented programs. The monitor uses timing information (Worst-Case Execution Time – WCET – of code regions) to detect attacks. An algorithm for the region selection, optimal when the monitoring memory is not limited is presented and provides guarantees on attack detection latency. An implementation of the hardware monitor and its simulation demonstrates the practicality of our approach. An experimental study evaluates the maximum attack detection latency for different monitor memory budgets.

This work is done in collaboration with the CIDRE and CAIRN teams.

7.4.2. Multi-nop fault injection attack

Participants: Damien Hardy, Erven Rohou.

The CIDRE team has developed a platform named Traitor that allows to perform multiple fault injection attack by replacing instructions by nops during the execution of a program. In this context, we are defining a program model where each instruction can be replaced by a nop at runtime. On this model we plan to apply compilation techniques on the binary to automatically determine where nops have to be inserted at runtime to perform sophisticated attacks such as dump of memory, modification of the memory, memory protection deactivation, execution of code in RAM.

This work is done in collaboration with the CIDRE team.

7.4.3. Compiler-based automation of side-channel countermeasures

Participants: Damien Hardy, Erven Rohou.

Masking is a popular protection against side-channel analysis exploiting the power consumption or electro-magnetic radiations. Besides the many schemes based on simple Boolean encoding, some alternative schemes such as Orthogonal Direct Sum Masking (ODSM) or Inner Product Masking (IP) aim to provide more security, reduce the entropy or combine masking with fault detection. The practical implementation of those schemes is done manually at assembly or source-code level, some of them even stay purely theoretical. We proposed a compiler extension to automatically apply different masking schemes for block cipher algorithms. We introduced a generic approach to describe the schemes and we inserted three of them at compile-time on an AES implementation. Currently, a practical side-channel analysis is performed in collaboration with TAMIS to assess the correctness and the performance of the code inserted.

This work is done in collaboration with the TAMIS team.

7.4.4. Platform for adaptive dynamic protection of programs

Participants: Kévin Le Bon, Erven Rohou.

Memory corruption attacks are a serious threat for system integrity. Many techniques have been developed in order to protect systems from these attacks. However, the deployment of heavy protections often degrades the performance of programs. We propose [36] a dynamic approach that adapts the protection level of the target process during its execution depending on the observed behavior.

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Grants with Industry

8.1.1. Intel research grant INTEL2016-11174

Participants: Niloofar Charmchi, Kleovoulos Kalaitzidis, Anis Peysieux, André Seznec.

Intel is supporting the research of the PACAP project-team on “Design tradeoffs for extreme cores”.

9. Partnerships and Cooperations

9.1. Regional Initiatives

The Brittany Region is partially funding the PhD fellowship for Niloofar Charmchi on the topic “Hardware prefetching and related issues” and Nicolas Bellec on the topic “Security in real-time embedded systems”.

9.2. National Initiatives

9.2.1. Zero Power Computing Systems (ZEP): Inria Project Lab (2017–2020)

Participants: Erven Rohou, Bahram Yarahmadi.

This proposal addresses the issue of designing tiny wireless, batteryless, computing objects, harvesting energy in the environment. The energy level harvested being very low, very frequent energy shortages are expected. In order for the new system to maintain a consistent state, it will be based on a new architecture embedding non-volatile RAM (NVRAM). In order to benefit from the hardware innovations related to energy harvesting and NVRAM, software mechanisms will be designed. On the one hand, a compilation pass will compute a worst-case energy consumption. On the other hand, dedicated runtime mechanisms will allow:

1. to manage efficiently and correctly the NVRAM-based hardware architecture;
2. to use energy intelligently, by computing the worst-case energy consumption.

The ZEP project gathers four Inria teams that have a scientific background in architecture, compilation, operating systems together with the CEA Lialp and Lisan laboratories of CEA LETI & LIST [39]. The main application target is Internet of Things (IoT).

9.2.2. NOPE

Participants: Piéric Giraud, Erven Rohou, Bahram Yarahmadi.

NOPE is a one-year exploratory action funded by the Labex Cominlabs. This project aimed at being a first step, and served to elaborate more ambitious future works. Through this project, the consortium was able to grow its knowledge on a topical research theme and lay the foundations of an innovative hardware-software approach. The short term goals were:

- building and sharing across the consortium a strong expertise in state-of-the art results and tools on transient computing, and identifying challenges that should be focused on;
- initiating collaborations between the participants in order to identify opportunities at the hardware-software interface;
- building the foundations of a shared experimental platform for transient computing.

An intern, Piéric Giraud, was hired thanks to NOPE. He ported our WCET infrastructure Heptane to the MSP430 instruction set.

The NOPE project gathers teams PACAP, IETR Syscom and LS2N STR.

9.2.3. Hybrid SIMD architectures (2018–2019)

Participants: Caroline Collange, Alexandre Kouyoumdjian, Erven Rohou.

The project objective is to define new parallel computer architectures that offer high parallel performance on high-regularity workloads while keeping the flexibility to run more irregular parallel workloads. inspired by both GPU and SIMD or vector architectures.

This project is funded by the French Ministry of Armed Forces (*Ministère des Armées*).

9.2.4. DGA/PEC ARMOUR (2018–2021)

Participants: Kévin Le Bon, Erven Rohou.

ARMOUR (dynAmic binaRy optiMizatiOn cyber-secURity) aims at improving the security of computing systems at the software level. Our contribution will be twofold: (1) identify vulnerabilities in existing software, and (2) develop adaptive countermeasure mechanisms against attacks. We will rely on dynamic binary rewriting (DBR) which consists in observing a program and modifying its binary representation in memory while it runs. DBR does not require the source code of the programs it manipulates, making it convenient for commercial and legacy applications. We will study the feasibility of an adaptive security agent that monitors target applications and deploys (or removes) countermeasures based on dynamic conditions. Lightweight monitoring is appropriate when the threat condition is low, heavy countermeasures will be dynamically woven into the code when an attack is detected. Vulnerability analysis will be based on advanced fuzzing. DBR makes it possible to monitor and modify deeply embedded variables, inaccessible to traditional monitoring systems, and also to detect unexpected/suspicious values taken by variables and act before the application crashes.

ARMOUR is funded by DGA (*Direction Générale de l'Armement*) and PEC (*Pôle d'Excellence Cyber*).

9.2.5. ANR DYVE (31/03/2020 – 30/09/2023)

Participants: Arthur Blanleuil, Caroline Collange, Pierre-Yves Peneau.

Most of today's computer systems have CPU cores and GPU cores on the same chip. Though both are general-purpose, CPUs and GPUs still have fundamentally different software stacks and programming models, starting from the instruction set architecture. Indeed, GPUs rely on static vectorization of parallel applications, which demands vector instruction sets instead of CPU scalar instruction sets. In the DYVE project, we advocate a disruptive change in both CPU and GPU architecture by introducing Dynamic Vectorization at the hardware level.

Dynamic Vectorization will combine the efficiency of GPUs with the programmability and compatibility of CPUs by bringing them together into heterogeneous general-purpose multicores. It will enable processor architectures of the next decades to provide (1) high performance on sequential program sections thanks to latency-optimized cores, (2) energy-efficiency on parallel sections thanks to throughput-optimized cores, (3) programmability, binary compatibility and portability.

DYVE is funded by the ANR through the JCJC funding instrument.

9.3. European Initiatives

9.3.1. FP7 & H2020 Projects

9.3.1.1. ARGO

Participants: Damien Hardy, Isabelle Puaut, Stefanos Skalistis.

Title: Argo: WCET-Aware Parallelization of Model-Based Applications for Heterogeneous Parallel Systems

Program: H2020

Type: RIA

Duration: Jan 2016 – Mar 2019

Coordinator: Karlsruhe Institut für Technologie (Germany)

Université de Rennes 1 contact: Steven Derrien

Partners:

Karlsruher Institut für Technologie (Germany)

SCILAB enterprises SAS (France)

Université de Rennes 1 (France)

Technologiko Ekpaideftiko Idryma (TEI) Dytikis Elladas (Greece)

Absint GmbH (Germany)

Deutsches Zentrum für Luft- und Raumfahrt EV (Germany)

Fraunhofer (Germany)

Increasing performance and reducing costs, while maintaining safety levels and programmability are the key demands for embedded and cyber-physical systems in European domains, e.g. aerospace, automation, and automotive. For many applications, the necessary performance with low energy consumption can only be provided by customized computing platforms based on heterogeneous many-core architectures. However, their parallel programming with time-critical embedded applications suffers from a complex toolchain and programming process. Argo (WCET-Aware PaRallelization of Model-Based Applications for HeteroGeneOus Parallel Systems) will address this challenge with a holistic approach for programming heterogeneous multi- and many-core architectures using automatic parallelization of model-based real-time applications. Argo will enhance WCET-aware automatic parallelization by a crosslayer programming approach combining automatic tool-based and user-guided parallelization to reduce the need for expertise in programming parallel heterogeneous architectures. The Argo approach will be assessed and demonstrated by prototyping comprehensive time-critical applications from both aerospace and industrial automation domains on customized heterogeneous many-core platforms.

Argo also involves Steven Derrien and Angeliki Kritikakou from the CAIRN team.

9.3.1.2. HiPEAC4 NoE

Participants: Pierre Michaud, Erven Rohou, André Seznec, Isabelle Puaut.

P. Michaud, A. Seznec and E. Rohou are members of the European Network of Excellence HiPEAC4.

HiPEAC4 addresses the design and implementation of high-performance commodity computing devices in the 10+ year horizon, covering both the processor design, the optimizing compiler infrastructure, and the evaluation of upcoming applications made possible by the increased computing power of future devices.

9.3.1.3. Eurolab-4-HPC

Participant: Erven Rohou.

Title: EuroLab-4-HPC: Foundations of a European Research Center of Excellence in High Performance Computing Systems

Program: H2020

Duration: September 2018 – September 2020

Coordinator: Chalmers Tekniska Hoegskola AB (Sweden)

Partners:

Barcelona Supercomputing Center - Centro Nacional de Supercomputacion (Spain)

Chalmers Tekniska Hoegskola (Sweden)

Foundation for Research and Technology Hellas (Greece)

Universität Stuttgart (Germany)

The University of Manchester (United Kingdom)

Inria (France)

Universität Augsburg (Germany)

ETH Zürich (Switzerland)

École Polytechnique Federale de Lausanne (Switzerland)

Technion - Israel Institute of Technology (Israel)

The University of Edinburgh (United Kingdom)

Rheinisch-Westfaelische Technische Hochschule Aachen (Germany)

Universiteit Gent (Belgium)

Inria contact: Albert Cohen (Inria Paris)

Europe has built momentum in becoming a leader in large parts of the HPC ecosystem. It has brought together technical and business stakeholders from application developers via system software to exascale systems. Despite such gains, excellence in high performance computing systems is often fragmented and opportunities for synergy missed. To compete internationally, Europe must bring together the best research groups to tackle the long-term challenges for HPC. These typically cut across layers, e.g., performance, energy efficiency and dependability, so excellence in research must target all the layers in the system stack. The EuroLab-4-HPC project's bold overall goal is to build connected and sustainable leadership in high-performance computing systems by bringing together the different and leading performance oriented communities in Europe, working across all layers of the system stack and, at the same time, fueling new industries in HPC.

9.4. International Initiatives

9.4.1. ANR CHIST-ERA SECODE 2016–2019

Participants: Damien Hardy, Erven Rohou.

Title: SECODE – Secure Codes to Thwart Cyber-Physical Attacks

CHIST-ERA - RTCPS

Duration: January 2016 – December 2019 (one year extension)

Coordinator: Télécom Paris Tech (France)

Partners:

Télécom Paris Tech (France)

Inria (France)

Université Paris 8 (France)

Sabancı Üniversitesi (Turkey)

Université Catholique de Louvain (Belgium)

Inria contact: Erven Rohou

In this project, we specify and design error correction codes suitable for an efficient protection of sensitive information in the context of Internet of Things (IoT) and connected objects. Such codes mitigate passive attacks, like memory disclosure, and active attacks, like stack smashing. The innovation of this project is to leverage these codes for protecting against both cyber and physical attacks. The main advantage is a full coverage of attacks of the connected embedded systems, which is considered as a smart connected device and also a physical device. The outcome of the project is first a method to generate and execute cyber-resilient software, and second to protect data and its manipulation from physical threats like side-channel attacks.

9.4.2. Informal International Partners

Caroline Collange has collaborated with Marcos Yukio Siraichi, Vinicius Fernandes dos Santos and Fernando Magno Quintão Pereira from UFMG, Brazil [31].

Isabelle Puaut has collaborated with Renato Mancuso (University of Boston, USA) and Heechul Yun (University of Kansas, USA) on predictable memory hierarchies [26]. She has collaborated with Martin Schoeberl (Technical University of Denmark) on predictable branch predictors [29].

Erven Rohou has been collaborating with Prof. Ahmed El-Mahdy (Egypt-Japan University of Science and Technology, Alexandria, Egypt) and his group [21], [22].

Erven Rohou and Loïc Besnard have been collaborating with Prof. João Cardoso (University of Porto, Porto, Portugal) and his group [16].

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific Events: Selection

10.1.1.1. Member of the Conference Program Committees

- Caroline Collange is a member of the program committee of the DATE 2020 conference
- Pierre Michaud was a member of the program committee of the International Conference on Computer Design (ICCD 2019)

- Pierre Michaud was a member of the program committee of the Third Data Prefetching Championship (DPC3)
- Isabelle Puaut was a member of the program committee of the Euromicro Conference on Real Time Systems (ECRTS) 2019 and 2020, IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS) 2020, Real Time Systems Symposium (RTSS) in 2019.
- Isabelle Puaut was a member of the program committee of the “Real-Time and (Networked) Embedded Systems” track of IEEE ETFA 2019.
- Isabelle Puaut was a member of the program committee of the International Conference on Real-Time Networks and Systems (RTNS) 2019 and 2020.
- Isabelle Puaut was a member of the program committee of the 18th Workshop on Worst-Case Execution Time Analysis (WCET 2019).
- Erven Rohou and Caroline Collange were members of the program committee of the French *Conférence francophone en informatique autour des thématiques du parallélisme, de l'architecture et des systèmes* (Compas 2019).
- Erven Rohou was a member of the program committee of the Special Session on Compiler Architecture, Design and Optimization (CADO) of HPCS 2019.
- André Seznec was a member of program committee of ACM/IEEE Micro 2019 conference
- André Seznec is a member of program committee of ACM/IEEE ISCA 2020 conference
- André Seznec is a member of program committee of the IPDPS 2020 conference

10.1.1.2. Reviewer

Members of PACAP routinely review submissions to numerous international conferences and events.

10.1.2. Journal

10.1.2.1. Member of the Editorial Boards

- Isabelle Puaut is Associate Editor of IEEE Transactions on Computers (IEEE TC) and Springer International Journal of Time-Critical Computing Systems.
- André Seznec is a member of the editorial boards of ACM Transactions on Architecture and Compiler Optimization.

10.1.2.2. Reviewer - Reviewing Activities

Members of PACAP routinely review submissions to numerous international journals.

10.1.3. Invited Talks

Erven Rohou and Byron Hawkins gave invited talks at the 13th *rencontres de la communauté française de compilation*.

Erven Rohou gave an invited talk at the GDR SOC2 day in Nantes.

Erven Rohou gave a talk at the Compas 2019 conference.

André Seznec was an invited speaker at the RISC V workshop in Paris.

10.1.4. Leadership within the Scientific Community

Caroline Collange is a member of the steering committee of the *Conférence francophone en informatique autour des thématiques du parallélisme, de l'architecture et des systèmes* (Compas 2019).

Isabelle Puaut is member of the steering committee of RTNS (Real-Time Networks and Systems).

Isabelle Puaut is member of the steering committee of the Worst Case Execution Time (WCET) workshop, held in conjunction with the Euromicro Conference on Real Time Systems (ECRTS).

Isabelle Puaut is member of the steering committee of the Euromicro Conference on Real Time Systems (ECRTS).

10.1.5. Research Administration

Isabelle Puaut is member of the Research Council (*Commission Recherche*) of the Université de Rennes 1. She is member of the working group “Habilitation à Diriger des Recherches”.

Isabelle Puaut is member of the board of directors (*Conseil d’Administration*) of ISTIC (computer science and electrical engineering departement of Université de Rennes 1).

Erven Rohou is “correspondant scientifique des relations internationales” for Inria Rennes Bretagne Atlantique. As such he is a member of the Inria COST GTRI (Groupe de Travail “Relations Internationales”).

Erven Rohou is a member of the steering committee of the high security research laboratory (LHS).

Erven Rohou is a member of the “Comité de Centre” of the Inria Rennes Research Center.

André Seznec is an elected member of the Administration Council of Inria.

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

- Licence: Nicolas Bellec, Système, 14 hours, L3, Université de Rennes 1, France
- License: Nicolas Bellec, Informatique, 14 hours, L1, Lycée Chateaubriand (Rennes), France
- Arthur Blanleuil, SYS1 (Systèmes mono-utilisateur), 18 hours, M1, Université Rennes 1, France
- Arthur Blanleuil, NOY (Systèmes d’exploitation – implémentation de noyaux de systèmes), 22 hours, M1, Université Rennes 1, France
- Arthur Blanleuil, SE (Système d’Exploitation), 24 hours, M1, Université Rennes 1, France
- Master: C. Collange, GPU programming, 20 hours, M1, Université de Rennes 1, France
- Licence: D. Hardy, Real-time systems, 68 hours, L3, Université de Rennes 1, France
- Master: D. Hardy, Operating systems, 59 hours, M1, Université de Rennes 1, France
- Master: Kévin Le Bon, Architecture à Objet Canonique, 20 hours, M2, Université de Rennes 1, France
- Licence: Kévin Le Bon, SI2, 40, L1, Université de Rennes 1, France
- Master: I. Puaut, Operating systems: concepts and system programming under Linux (SEL), 77 hours, M1, Université de Rennes 1, France
- Master: I. Puaut, Operating systems kernels (NOY), 54 hours, M1, Université de Rennes 1, France
- Master: I. Puaut, Real-time systems, 40 hours, M1, Université de Rennes 1, France
- Master: I. Puaut, Optimizing and Parallelizing Compilers (OPC), 9 hours, M2, Université de Rennes 1, France
- Master: I. Puaut, Writing of scientific publications, 9 hours, M2 and PhD students, Université de Rennes 1, France
- Master: A. Seznec, Advanced Design and Architectures, 12 hours, M2 SIF, Université de Rennes 1.

10.2.2. Supervision

PhD: Arif Ali Ana-Pparakkal, *Performance Centric Dynamic Function Level Binary Transformation* [15], Université de Rennes 1, 9 Dec 2019, advisor E. Rohou

PhD in progress : Nicolas Bellec, *Security in real-time embedded systems*, started Dec 2019, advisors I. Puaut (50 %), G. Hiet from CIDRE (25 %), F. Tronel from CIDRE (25 %)

PhD in progress: Arthur Blanleuil, *Thread convergence prediction for SIMT architectures*, Université de Rennes 1, started Oct 2018, advisor C. Collange and A. Seznec

PhD in progress: Niloofar Charmchi, *Hardware prefetching and related issue*, Université de Rennes 1, started Jan 2017, advisors A. Seznec and C. Collange

PhD in progress: Kleovoulos Kalaitzidis, *Ultrawide Issue Superscalar Processors*, Université de Rennes 1, started Dec 2016, advisor A. Seznec

PhD in progress : Kévin Le Bon, *Dynamic Binary Analysis and Optimization for Cyber-Security*, started Dec 2018, advisors E. Rohou (30 %), G. Hiet from CIDRE (35 %), F. Tronel from CIDRE (35 %)

PhD in progress: Daniel Rodrigues Carvalho, *Towards a compressed memory hierarchy*, Université de Rennes 1, started Oct 2017, advisor A. Seznec

PhD in progress : Bahram Yarahmadi, *Compiler Optimizations and Worst-Case Energy Consumption*, started Feb 2018, advisor E. Rohou

10.2.3. Juries

Isabelle Puaut was a member of the following PhD thesis and habilitation thesis committees:

- Muhammad Refaat Sedky Soliman (PhD thesis), *Automated Compilation Framework for Scratchpad-based Real-Time Systems*, University of Waterloo, Canada, Juin 2019 (reviewer)
- Amine Naji (PhD thesis), *Timing Analysis for Time-Predictable Architectures*, Sorbonne Université, June 2019 (reviewer)
- Luca Santinelli (habilitation), *Mixed Criticality Modeling and Analysis Paradigms for Real Time Embedded Systems*, Habilitation à diriger des recherches, Institut National Polytechnique de Toulouse, May 2019 (reviewer)
- Roberto Medina (PhD thesis), *Deployment of Mixed-Criticality and Data-Driven Systems on Multi-core Architectures*, Université de Paris Saclay, January 2019

Isabelle Puaut was a member of the following hiring committees of assistant professors/professors:

- Professor, Université de Bretagne Occidentale, “methods and tools for the design of embedded systems, Systems on Chips, embedded systems and architectures”
- Professor, ENS Lyon
- Assistant professor, Université de Strasbourg, “performance, modeling and simulation”
- Assistant professor, ISAE-ENSMA Poitiers, “embedded real-time systems, data and model engineering”

Erven Rohou was a member of the following hiring committee (comités de sélection):

- Assistant professor position at Université de Grenoble on cloud and edge computing

Erven Rohou was a member of the following PhD thesis committees:

- Julien Proy, *Sécurisation systématique d’applications embarquées contre les attaques physiques*, June 2019 (reviewer).
- Nicolas Belleville, *Compilation pour l’application de contre-mesures contre les attaques par canal auxiliaire*, Université Grenoble Alpes, Nov 2019 (reviewer).
- Patryk Kiepas, *Performance analyses and code transformations for MATLAB applications*, Dec 2019 (reviewer).
- Paul Godard, *Parallélisation et passage à l’échelle durable d’une chaîne de traitement graphique pour l’impression professionnelle*, Université de Strasbourg, Dec 2019 (reviewer).

10.3. Popularization

10.3.1. Internal or external Inria responsibilities

Caroline Collange is a member of the committee of the Gilles Kahn PhD prize of Société Informatique de France.

10.3.2. Articles and contents

- Erven Rohou gave an interview in the *Émergences* magazine on (volume 56, Mar 22nd 2019) on automatically masking sensitive information to protect against side-channel attacks.

10.3.3. Education

Nicolas Bellec and Isabelle Puaut taught *Basics of computer architecture*, a training of high school teachers as part of the opening of the new computer science option in the two final years before Baccalauréat, 10 hours.

Erven Rohou was invited to present the life of a researcher in computer science to middle school students (*Collège de Cesson-Sévigné)*

10.3.4. Interventions

Erven Rohou was present at the International Cybersecurity Forum in Lille (FIC https://www.forum-fic.com/en/home.htm). He presented a demo on compiler-generated countermeasures against side-channel attacks, developed within the context of the CHIST-ERA SECODE project.

10.3.5. Internal action

Erven Rohou presented the activities of PACAP at the SecDays event organized at Inria Rennes and IRISA.

11. Bibliography

Major publications by the team in recent years

[1] F. Bodin, T. Kisuki, P. M. W. Knijnenburg, M. F. P. O'Boyle, E. Rohou. *Iterative Compilation in a Non-Linear Optimisation Space*, in "Workshop on Profile and Feedback-Directed Compilation (FDO-1), in conjunction with PACT '98", October 1998

[2] N. Hallou, E. Rohou, P. Clauss, A. Ketterlin. *Dynamic Re-Vectorization of Binary Code*, in "SAMOS", July 2015, https://hal.inria.fr/hal-01155207

[3] D. Hardy, I. Puaut. *Static probabilistic Worst Case Execution Time Estimation for architectures with Faulty Instruction Caches*, in "21st International Conference on Real-Time Networks and Systems", Sophia Antipolis, France, October 2013 [*DOI :* 10.1145/2516821.2516842], https://hal.inria.fr/hal-00862604

[4] D. Hardy, I. Sideris, N. Ladas, Y. Sazeides. *The performance vulnerability of architectural and non-architectural arrays to permanent faults*, in "MICRO 45", Vancouver, Canada, December 2012, https://hal.inria.fr/hal-00747488

[5] S. Kalathingal, S. Collange, B. Swamy, A. Seznec. *DITVA: Dynamic Inter-Thread Vectorization Architecture*, in "Journal of Parallel and Distributed Computing", October 2018, pp. 1-32 [*DOI :* 10.1016/j.jpdc.2017.11.006], https://hal.archives-ouvertes.fr/hal-01655904

[6] P. Michaud. *Best-Offset Hardware Prefetching*, in "International Symposium on High-Performance Computer Architecture", Barcelona, Spain, March 2016 [*DOI :* 10.1109/HPCA.2016.7446087], https://hal.inria.fr/hal-01254863

[7] P. Michaud, A. Mondelli, A. Seznec. *Revisiting Clustered Microarchitecture for Future Superscalar Cores: A Case for Wide Issue Clusters*, in "ACM Transactions on Architecture and Code Optimization (TACO)", August 2015, vol. 13, n^{o} 3, 22 p. [*DOI :* 10.1145/2800787], https://hal.inria.fr/hal-01193178

[8] A. PERAIS, A. SEZNEC. *EOLE: Paving the Way for an Effective Implementation of Value Prediction*, in "International Symposium on Computer Architecture", Minneapolis, MN, United States, ACM/IEEE, June 2014, vol. 42, pp. 481-492 [*DOI :* 10.1109/ISCA.2014.6853205], https://hal.inria.fr/hal-01088130

[9] A. PERAIS, A. SEZNEC. *Practical data value speculation for future high-end processors*, in "International Symposium on High Performance Computer Architecture", Orlando, FL, United States, IEEE, February 2014, pp. 428-439 [*DOI :* 10.1109/HPCA.2014.6835952], https://hal.inria.fr/hal-01088116

[10] E. ROHOU, B. NARASIMHA SWAMY, A. SEZNEC. *Branch Prediction and the Performance of Interpreters - Don't Trust Folklore*, in "International Symposium on Code Generation and Optimization", Burlingame, United States, February 2015, https://hal.inria.fr/hal-01100647

[11] S. SARDASHTI, A. SEZNEC, D. A. WOOD. *Skewed Compressed Caches*, in "47th Annual IEEE/ACM International Symposium on Microarchitecture, 2014", Minneapolis, United States, December 2014, https://hal.inria.fr/hal-01088050

[12] S. SARDASHTI, A. SEZNEC, D. A. WOOD. *Yet Another Compressed Cache: a Low Cost Yet Effective Compressed Cache*, in "ACM Transactions on Architecture and Code Optimization", September 2016, 25 p. , https://hal.inria.fr/hal-01354248

[13] A. SEZNEC, P. MICHAUD. *A case for (partially)-tagged geometric history length branch prediction*, in "Journal of Instruction Level Parallelism", February 2006, http://www.jilp.org/vol8

[14] D. D. C. TEIXEIRA, S. COLLANGE, F. M. QUINTÃO PEREIRA. *Fusion of calling sites*, in "International Symposium on Computer Architecture and High-Performance Computing (SBAC-PAD)", Florianópolis, Santa Catarina, Brazil, October 2015 [*DOI :* 10.1109/SBAC-PAD.2015.16], https://hal.archives-ouvertes.fr/hal-01410221

Publications of the year

Doctoral Dissertations and Habilitation Theses

[15] A. A. ANAPPARAKKAL. *Performance Centric Dynamic Function Level Binary Transformation*, Université de Rennes 1 [UR1], December 2019, https://hal.inria.fr/tel-02394383

Articles in International Peer-Reviewed Journals

[16] L. BESNARD, P. PINTO, I. LASRI, J. BISPO, E. ROHOU, J. M. P. CARDOSO. *A framework for automatic and parameterizable memoization*, in "SoftwareX", July 2019, vol. 10, 100322 p. [*DOI :* 10.1016/J.SOFTX.2019.100322], https://hal.inria.fr/hal-02305415

[17] C. LIMA, J. CEZAR, G. VIEIRA LEOBAS, E. ROHOU, F. M. QUINTÃO PEREIRA. *Guided just-in-time specialization*, in "Science of Computer Programming", November 2019, vol. 185, 41 p. [*DOI :* 10.1016/J.SCICO.2019.102318], https://hal.inria.fr/hal-02314442

[18] V. A. NGUYEN, D. HARDY, I. PUAUT. *Cache-conscious Off-Line Real-Time Scheduling for Multi-Core Platforms: Algorithms and Implementation*, in "Real-Time Systems", 2019, pp. 1-37, forthcoming [*DOI :* 10.4230/LIPICS.ECRTS.2017.14], https://hal.inria.fr/hal-02044110

[19] S. REDER, F. KEMPF, H. BUCHER, J. BECKER, P. ALEFRAGIS, N. S. VOROS, S. SKALISTIS, S. DERRIEN, I. PUAUT, O. OEY, T. STRIPF, C. FERDINAND, C. DAVID, P. ULBIG, D. MUELLER, U. DURAK. *Worst-Case Execution-Time-Aware Parallelization of Model-Based Avionics Applications*, in "Journal of Aerospace Information Systems", November 2019, vol. 16, n^o 11, pp. 521-533 [*DOI :* 10.2514/1.I010749], https://hal.archives-ouvertes.fr/hal-02383381

[20] C. SILVANO, G. AGOSTA, A. BARTOLINI, A. R. BECCARI, L. BENINI, L. BESNARD, J. BISPO, R. CMAR, J. M. P. CARDOSO, C. CAVAZZONI, D. CESARINI, S. CHERUBIN, F. FICARELLI, D. GADIOLI, M. GOLASOWSKI, A. LIBRI, J. MARTINOVIČ, G. PALERMO, P. PINTO, E. ROHOU, K. SLANINOVÁ, E. VITALI. *The ANTAREX domain specific language for high performance computing*, in "Microprocessors and Microsystems: Embedded Hardware Design (MICPRO)", July 2019, vol. 68, pp. 58-73 [*DOI :* 10.1016/J.MICPRO.2019.05.005], https://hal.inria.fr/hal-02189586

[21] M. YUSUF, A. EL-MAHDY, E. ROHOU. *Runtime On-Stack Parallelization of Dependence-Free For-Loops in Binary Programs*, in "IEEE Letters of the Computer Society", March 2019, vol. 2, n^o 1, pp. 1-4 [*DOI :* 10.1109/LOCS.2019.2896559], https://hal.inria.fr/hal-02061340

[22] M. YUSUF, A. EL-MAHDY, E. ROHOU. *Towards Automatic Binary Runtime Loop De-Parallelization using On-Stack Replacement*, in "Information Processing Letters", May 2019, vol. 145, pp. 53-57 [*DOI :* 10.1016/J.IPL.2019.01.009], https://hal.inria.fr/hal-02002812

International Conferences with Proceedings

[23] N. CHARMCHI, C. COLLANGE, A. SEZNEC. *Compressed cache layout aware prefetching*, in "SBAC-PAD 2019 - International Symposium on Computer Architecture and High Performance Computing", Campo Grande, MS, Brazil, October 2019, pp. 1-4, https://hal.inria.fr/hal-02316773

[24] M. DARDAILLON, S. SKALISTIS, I. PUAUT, S. DERRIEN. *Reconciling Compiler Optimizations and WCET Estimation Using Iterative Compilation*, in "RTSS 2019 - 40th IEEE Real-Time Systems Symposium", Hong Kong, China, IEEE, December 2019, pp. 1-13, https://hal.archives-ouvertes.fr/hal-02286164

[25] K. KALAITZIDIS, A. SEZNEC. *Value Speculation through Equality Prediction*, in "ICCD 2019 - 37th IEEE International Conference on Computer Design", Abu Dhabi, United Arab Emirates, IEEE, November 2019, pp. 1-4, https://hal.archives-ouvertes.fr/hal-02383480

[26] R. MANCUSO, H. YUN, I. PUAUT. *Impact of DM-LRU on WCET: A Static Analysis Approach*, in "ECRTS 2019 - 31st Euromicro Conference on Real-Time Systems", Stuttgart, Germany, July 2019, pp. 1-25 [*DOI :* 10.4230/LIPICS.ECRTS.2019.17], https://hal.archives-ouvertes.fr/hal-02190255

[27] S. ROKICKI, E. ROHOU, S. DERRIEN. *Aggressive Memory Speculation in HW/SW Co-Designed Machines*, in "DATE 2019 - 22nd IEEE/ACM Design, Automation and Test in Europe", Florence, Italy, IEEE, March 2019, pp. 332-335 [*DOI :* 10.23919/DATE.2019.8715010], https://hal.archives-ouvertes.fr/hal-01941876

[28] *Best Paper*
B. ROUXEL, S. SKALISTIS, S. DERRIEN, I. PUAUT. *Hiding Communication Delays in Contention-Free Execution for SPM-Based Multi-Core Architectures*, in "ECRTS 2019 - 31st Euromicro Conference on Real-Time Systems", Stuttgart, Germany, July 2019, pp. 1-24 [*DOI :* 10.4230/LIPICS.ECRTS.2019.25], https://hal.archives-ouvertes.fr/hal-02190271.

[29] M. SCHOEBERL, B. ROUXEL, I. PUAUT. *A Time-predictable Branch Predictor*, in "SAC 2019 - 34th ACM/SIGAPP Symposium on Applied Computing", Limassol, Cyprus, April 2019, pp. 1-10 [*DOI :* 10.1145/3297280.3297337], https://hal.inria.fr/hal-01976187

[30] C. SILVANO, G. AGOSTA, A. BARTOLINI, A. R. BECCARI, L. BENINI, L. BESNARD, J. BISPO, R. CMAR, J. M. P. CARDOSO, C. CAVAZZONI, D. CESARINI, S. CHERUBIN, F. FICARELLI, D. GADIOLI, M. GOLASOWSKI, I. LASRI, A. LIBRI, J. MARTINOVIČ, G. PALERMO, P. PINTO, E. ROHOU, N. SANNA, K. SLANINOVÁ, E. VITALI. *Adaptive Optimization and Enforcement of Extra-Functional Properties in High Performance Computing: The ANTAREX Project*, in "PDP 2019 - 27th Euromicro International Conference on Parallel, Distributed and Network-Based Processing", Pavia, Italy, IEEE, February 2019, pp. 116-123 [*DOI :* 10.1109/EMPDP.2019.8671584], https://hal.inria.fr/hal-02197811

[31] M. Y. SIRAICHI, V. F. D. SANTOS, C. COLLANGE, F. M. QUINTÃO PEREIRA. *Qubit allocation as a combination of subgraph isomorphism and token swapping*, in "OOPSLA", Athens, Greece, October 2019, vol. 3, pp. 1-29 [*DOI :* 10.1145/3360546], https://hal.inria.fr/hal-02316820

[32] S. SKALISTIS, A. KRITIKAKOU. *Timely Fine-grained Interference-sensitive Run-time Adaptation of Time-triggered Schedules*, in "RTSS 2019 - 40th IEEE Real-Time Systems Symposium", Hong Kong, China, IEEE, December 2019, pp. 1-13, https://hal.archives-ouvertes.fr/hal-02316392

Conferences without Proceedings

[33] N. CHARMCHI, C. COLLANGE. *Toward compression-aware prefetching*, in "COMPAS 2019 - Conférence d'informatique en Parallélisme, Architecture et Système", Anglet, France, June 2019, pp. 1-9, https://hal.inria.fr/hal-02351461

[34] C. COLLANGE. *Ordinateurs quantiques : ouvrons la boîte*, in "COMPAS 2019 - Conférence d'informatique en Parallélisme, Architecture et Système", Anglet, France, June 2019, pp. 1-9, https://hal.inria.fr/hal-02318324

[35] A. KOUYOUMDJIAN, C. COLLANGE, E. ROHOU. *Vers la reconfiguration adaptative de GPU pour chaque application*, in "COMPAS 2019 - Conférence d'informatique en Parallélisme, Architecture et Système", Anglet, France, June 2019, pp. 1-6, https://hal.inria.fr/hal-02390821

[36] K. LE BON, B. HAWKINS, E. ROHOU, G. HIET, F. TRONEL. *Plateforme de protection de binaires configurable et dynamiquement adaptative*, in "RESSI 2019 - Rendez-Vous de la Recherche et de l'Enseignement de la Sécurité des Systèmes d'Information", Erquy, France, May 2019, pp. 1-3, https://hal.inria.fr/hal-02385216

Research Reports

[37] P. MICHAUD. *A Simple Model of Processor Temperature for Deterministic Turbo Clock Frequency*, Inria Rennes, December 2019, n^{o} RR-9308, https://hal.inria.fr/hal-02391970

Other Publications

[38] S. ROKICKI, E. ROHOU, S. DERRIEN. *Hybrid-DBT: Hardware Accelerated Dynamic Binary Translation*, June 2019, 1 p. , RISC-V 2019 - Workshop Zurich, Poster, https://hal.archives-ouvertes.fr/hal-02155019

References in notes

[39] G. Berthou, A. Carer, H.-P. Charles, S. Derrien, K. Marquet, I. Miro-Panades, D. Pala, I. Puaut, F. Rastello, T. Risset, E. Rohou, G. Salagnac, O. Sentieys, B. Yarahmadi. *The Inria ZEP project: NVRAM and Harvesting for Zero Power Computations*, March 2018, 1 p. , NVMW 2018 - 10th Annual Non-Volatile Memories Workshop, Poster, https://hal.inria.fr/hal-01941766

[40] A. Cohen, E. Rohou. *Processor Virtualization and Split Compilation for Heterogeneous Multicore Embedded Systems*, in "DAC", June 2010, pp. 102–107

[41] M. Hataba, A. El-Mahdy, E. Rohou. *OJIT: A Novel Obfuscation Approach Using Standard Just-In-Time Compiler Transformations*, in "International Workshop on Dynamic Compilation Everywhere", January 2015

[42] S. Kalathingal, S. Collange, B. Narasimha Swamy, A. Seznec. *Dynamic Inter-Thread Vectorization Architecture: extracting DLP from TLP*, in "International Symposium on Computer Architecture and High-Performance Computing (SBAC-PAD)", Los Angeles, United States, October 2016, https://hal.inria.fr/hal-01356202

[43] R. Kumar, D. M. Tullsen, N. P. Jouppi, P. Ranganathan. *Heterogeneous chip multiprocessors*, in "IEEE Computer", nov. 2005, vol. 38, n^{o} 11, pp. 32–38

[44] P. Michaud, A. Seznec. *Pushing the branch predictability limits with the multi-poTAGE+SC predictor : **Champion in the unlimited category***, in "4th JILP Workshop on Computer Architecture Competitions (JWAC-4): Championship Branch Prediction (CBP-4)", Minneapolis, United States, June 2014, https://hal.archives-ouvertes.fr/hal-01087719

[45] R. Omar, A. El-Mahdy, E. Rohou. *Arbitrary control-flow embedding into multiple threads for obfuscation: a preliminary complexity and performance analysis*, in "Proceedings of the 2nd international workshop on Security in cloud computing", ACM, 2014, pp. 51–58

[46] E. Riou, E. Rohou, P. Clauss, N. Hallou, A. Ketterlin. *PADRONE: a Platform for Online Profiling, Analysis, and Optimization*, in "Dynamic Compilation Everywhere", Vienna, Austria, January 2014

[47] A. Sembrant, T. Carlson, E. Hagersten, D. Black-Shaffer, A. Perais, A. Seznec, P. Michaud. *Long Term Parking (LTP): Criticality-aware Resource Allocation in OOO Processors*, in "International Symposium on Microarchitecture, Micro 2015", Honolulu, United States, Proceeding of the International Symposium on Microarchitecture, Micro 2015, ACM, December 2015, https://hal.inria.fr/hal-01225019

[48] A. Seznec, J. San Miguel, J. Albericio. *The Inner Most Loop Iteration counter: a new dimension in branch history* , in "48th International Symposium On Microarchitecture", Honolulu, United States, ACM, December 2015, 11 p. , https://hal.inria.fr/hal-01208347

[49] A. Seznec, N. Sendrier. *HAVEGE: A user-level software heuristic for generating empirically strong random numbers*, in "ACM Transactions on Modeling and Computer Simulation (TOMACS)", 2003, vol. 13, n^{o} 4, pp. 334–346

[50] A. Seznec. *TAGE-SC-L Branch Predictors: **Champion in 32Kbits and 256 Kbits category***, in "JILP - Championship Branch Prediction", Minneapolis, United States, June 2014, https://hal.inria.fr/hal-01086920