IN PARTNERSHIP WITH:
Université Rennes 1

Activity Report 2015

Project-Team ALF

Amdahl's Law is Forever

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

Project-Team ALF

Creation of the Team: 2009 January 01, updated into Project-Team: 2011 January 01

Keywords:

Computer Science and Digital Science:

1.1. - Architectures
1.1.1. - Multicore
1.1.2. - Hardware accelerators (GPGPU, FPGA, etc.)
1.1.3. - Memory models
1.1.4. - High performance computing
1.6. - Green Computing
2.2. - Compilation
2.2.2. - Memory models
2.2.3. - Run-time systems
2.2.4. - Parallel architectures
2.2.5. - GPGPU, FPGA, etc.
2.2.6. - Adaptive compilation
2.3.1. - Embedded systems
2.3.3. - Real-time systems

Other Research Topics and Application Domains:

1. - Life sciences
2. - Health
3. - Environment and planet
4. - Energy
5. - Industry of the future
6. - IT and telecom
7. - Transport and logistics
8. - Smart Cities and Territories
9. - Society and Knowledge

1. Members

Research Scientists

André Seznec [Team leader, Inria, Senior Researcher, HdR]
Sylvain Collange [Inria, Researcher]
Pierre Michaud [Inria, Researcher]
Erven Rohou [Inria, Senior Researcher, HdR]

Faculty Members

Damien Hardy [Univ. Rennes I, Associate Professor]
Isabelle Puaut [Univ. Rennes I, Professor, HdR]

Engineers

Arthur Perais [Inria, from Oct 2015]
Thibault Person [Inria, until Oct 2015]

Emmanuel Riou [Inria, granted by Nano 2017]

PhD Students

Arif Ali Ana-Pparakkal [Inria, from Feb 2015,granted by Nano 2017]
Rabab Bouziane [Inria, from Nov 2015]
Nabil Hallou [Inria]
Sajith Kalathingal [Inria, granted by FP7 ERC DAL project]
Surya Khizakanchery Natarajan [Inria, until Jun 2015, granted by FP7 ERC DAL project]
Hanbing Li [Inria, until Oct 2015, granted by ANR W-SEPT project]
Andrea Mondelli [Inria, granted by FP7 ERC DAL project]
Bharath Narasimha Swamy [Inria, until Mar 2015, granted by FP7 ERC DAL project]
Viet Anh Nguyen [Univ. Rennes I, from Feb 2015]
Arthur Perais [Inria, until Sep 2015, granted by FP7 ERC DAL project]
Benjamin Rouxel [Univ. Rennes I, from Sept 2015]
Aswinkumar Sridharan [Inria, granted by FP7 ERC DAL project]
Arjun Suresh [Inria, granted by FP7 ERC DAL project]

Post-Doctoral Fellows

Fernando Endo [Inria, from Oct 2015]
Biswabandan Panda [Inria, from Nov 2015]
Tao Sun [Inria, until Aug 2015, granted by FP7 ERC DAL project]

Administrative Assistant

Virginie Desroches [Inria]

2. Overall Objectives

2.1. Panorama

Multicore processors have now become mainstream for both general-purpose and embedded computing. In the near future, every hardware platform will feature thread level parallelism. Therefore, the overall computer science research community, but also industry, is facing new challenges; parallel architectures will have to be exploited by every application from HPC computing, web and entreprise servers, but also PCs, smartphones and ubiquitous embedded systems.

Within a decade, it will become technologically feasible to implement 1000s of cores on a single chip. However, several challenges must be addressed to allow the end-user to benefit from these 1000's cores chips. At that time, most applications will not be fully parallelized, therefore the effective performance of most computer systems will strongly depend on their performance on sequential sections and sequential control threads: Amdahl's law is forever. Parallel applications will not become mainstream if they have to be adapted to each new platform, therefore a simple performance scalability/portability path is needed for these applications. In many application domains, particularly in real-time systems, the effective use of multicore chips will depend on the ability of the software and hardware vendors to accurately assess the performance of applications.

The ALF team regroups researchers in computer architecture, software/compiler optimization, and real-time systems. The long-term goal of the ALF project-team is to allow the end-user to benefit from the 2020's many-core platform. We address this issue through architecture, i.e. we try to influence the definition of the 2020's many-core architecture, compiler, i.e. we intend to provide new code generation techniques for efficient execution on many-core architectures and performance prediction/guarantee, i.e. we try to propose new software and architecture techniques to predict/guarantee the response time of many-core architectures.

High performance on single thread process and sequential code is a key issue for enabling overall high performance on a 1000's cores system. Therefore, we anticipate that future manycore architectures will implement heterogeneous design featuring many simple cores and a few complex cores. Hence the research in the ALF project focuses on refining the microarchitecture to achieve high performance on single thread process and/or sequential code sections. We focus our architecture research in two main directions 1) enhancing the microarchitecture of high-end superscalar processors, 2) exploiting/modifying heterogeneous multicore architecture on a single thread. We also tackle a technological/architecture issue, the temperature wall.

Compilers are keystone solutions for any approach that deals with high performance on 100+ core systems. But general-purpose compilers try to embrace so many domains and try to serve so many constraints that they frequently fail to achieve very high performance. They need to be deeply revisited. We identify four main compiler/software related issues that must be addressed in order to allow efficient use of multi- and many-cores: 1) programming 2) resource management 3) application deployment 4) portable performance. Addressing these challenges requires to revisit parallel programming and code generation extensively.

While compiler and architecture research efforts often focus on maximizing average case performance, applications with real-time constraints do not only need high performance but also performance guarantees in all situations, including the worst-case situation. Worst-Case Execution Time estimates (WCET) need to be upper bounds of any possible execution time. The amount of safety required depends on the criticality of applications. Within the ALF team, our objective is to study performance guarantees for both (i) sequential codes running on complex cores ; (ii) parallel codes running on multicores.

Our research is partially supported by industry (Intel,STmicroelectronics), the ANR W-SEPT, ANR Continuum and ANR CHIST-ERA SECODE projects, the "projet d'investissement d'avenir" Usine Nouvelle (the project Capacités) and the European Union (NoE HiPEAC3, ERC grant DAL, Antarex, Eurolab-4-HPC, ARGO and COST action TACLe).

3. Research Program

3.1. Motivations

Multicores have become mainstream in general-purpose as well as embedded computing in the last few years. The integration technology trend allows to anticipate that a 1000-core chip will become feasible before 2020. On the other hand, while traditional parallel application domains, e.g. supercomputing and transaction servers, are benefiting from the introduction of multicores, there are very few new parallel applications that have emerged during the last few years.

In order to allow the end-user to benefit from the technological breakthrough, new architectures have to be defined for the 2020's many-cores, new compiler and code generation techniques as well as new performance prediction/guarantee techniques have to be proposed .

3.2. The context

3.2.1. Technological context: The advent of multi- and many- core architecture

For almost 30 years since the introduction of the first microprocessor, the processor industry was driven by the Moore's law till 2002, delivering performance that doubled every 18-24 months on a uniprocessor. However since 2002 , and despite new progress in integration technology, the efforts to design very aggressive and very complex wide issue superscalar processors have essentially been stopped due to poor performance returns, as well as power consumption and temperature walls.

Since 2002-2003, the microprocessor industry has followed a new path for performance: the so-called multicore approach, i.e., integrating several processors on a single chip. This direction has been followed by the whole processor industry. At the same time, most of the computer architecture research community has taken the same path, focusing on issues such as scalability in multicores, power consumption, temperature management and new execution models, e.g. hardware transactional memory.

In terms of integration technology, the current trend will allow to continue to integrate more and more processors on a single die. Doubling the number of cores every two years will soon lead to up to a thousand processor cores on a single chip. The computer architecture community has coined these future processor chips as many-cores.

3.2.2. The application context: multicores, but few parallel applications

For the past five years, small scale parallel processor chips (hyperthreading, dual and quad-core) have become mainstream in general-purpose systems. They are also entering the high-end embedded system market. At the same time, very few (scalable) mainstream parallel applications have been developed. Such development of scalable parallel applications is still limited to niche market segments (scientific applications, transaction servers).

3.2.3. The overall picture

Till now, the end-user of multicores is experiencing improved usage comfort because he/she is able to run several applications at the same time. Eventually, in the near future with the 8-core or the 16-core generation, the end-user will realize that he/she is not experiencing any functionality improvement or performance improvement on current applications. The end-user will then realize that he/she needs more effective performance rather than more cores. The end-user will then ask either for parallel applications or for more effective performance on sequential applications.

3.3. Technology induced challenges

3.3.1. The power and temperatures walls

The power and the temperature walls largely contributed to the emergence of the small-scale multicores. For the past five years, mainstream general-purpose multicores have been built by assembling identical superscalar cores on a chip (e.g. IBM Power series). No new complex power hungry mechanisms were introduced in the core architectures, while power saving techniques such as power gating, dynamic voltage and frequency scaling were introduced. Therefore, since 2002, the designers have been able to keep the power consumption budget and the temperature of the chip within reasonable envelopes while scaling the number of cores with the technology.

Unfortunately, simple and efficient power saving techniques have already caught most of the low hanging fruits on energy consumption. Complex power and thermal management mechanisms are now becoming mainstream; e.g. the Intel Montecito (IA64) featured an adjunct (simple) core whose unique mission is to manage the power and temperature on two cores. Processor industry will require more and more heroic efforts on this power and temperature management policy to maintain its current performance scaling path. Hence the power and temperature walls might slow the race towards 100's and 1000's cores unless the processor industry takes a new paradigm shift from the current "replicating complex cores" (e.g. Intel Nehalem) towards many simple cores (e.g. Intel Larrabee) or heterogeneous manycores (e.g. new GPUs, IBM Cell).

3.3.2. The memory wall

For the past 20 years, the memory access time has been one of the main bottlenecks for performance in computer systems. This was already true for uniprocessors. Complex memory hierarchies have been defined and implemented in order to limit the visible memory access time as well as the memory traffic demands. Up to three cache levels are implemented for uniprocessors. For multi- and many-cores the problems are even worse. The memory hierarchy must be replicated for each core, memory bandwidth must be shared among the distinct cores, data coherency must be maintained. Maintaining cache coherency for up to 8 cores can be handled through relatively simple bus protocols. Unfortunately, these protocols do not scale for large numbers of cores, and there is no consensus on coherency mechanism for manycore systems. Moreover there is no consensus on core organization (flat ring? flat grid? hierarchical ring or grid?).

Therefore, organizing and dimensioning the memory hierarchy will be a major challenge for the computer architects. The successful architecture will also be determined by the abilitty of the applications (i.e., the programmers or the compilers or the run-time) to efficiently place data in the memory hierarchy and achieve high performance.

Finally new technology opportunities may demand to revisit the memory hierarchy. As an example, 3D memory stacking enables a huge last-level cache (maybe several gigabytes) with huge bandwidth (several Kbits/ processor cycle). This dwarfs the main memory bandwidth and may lead to other architectural tradeoffs.

3.4. Need for efficient execution of parallel applications

Achieving high performance on future multicores will require the development of parallel applications, but also an efficient compiler/runtime tool chain to adapt codes to the execution platform.

3.4.1. The diversity of parallelisms

Many potential execution parallelism patterns may coexist in an application. For instance, one can express some parallelism with different tasks achieving different functionalities. Within a task, one can expose different granularities of parallelism; for instance a first layer message passing parallelism (processes executing the same functionality on different parts of the data set), then a shared memory thread level parallelism and fine grain loop parallelism (a.k.a vector parallelism).

Current multicores already feature hardware mechanisms to address these different parallelisms: physically distributed memory — e.g. the new Intel Nehalem already features 6 different memory channels — to address task parallelism, thread level parallelism — e.g. on conventional multicores, but also on GPUs or on Cell-based machines —, vector/SIMD parallelism — e.g. multimedia instructions. Moreover they also attack finer instruction level parallelism and memory latency issues. Compilers have to efficiently discover and manage all these forms to achieve effective performance.

3.4.2. Portability is the new challenge

Up to now, most parallel applications were developed for specific application domains in high end computing. They were used on a limited set of very expensive hardware platforms by a limited number of expert users. Moreover, they were executed in batch mode.

In contrast, the expectation of most end-users of the future mainstream parallel applications running on multicores will be very different. The mainstream applications will be used by thousands, maybe millions of non-expert users. These users consider functional portability of codes as granted. They will expect their codes to run faster on new platforms featuring more cores. They will not be able to tune the application environment to optimize performance. Finally, multiple parallel applications may have to be executed concurrently.

The variety of possible hardware platforms, the lack of expertise of the end-users and the varying run-time execution environments will represent major difficulties for applications in the multicore era.

First of all, the end user considers functional portability without recompilation as granted, this is a major challenge on parallel machines. Performance portability/scaling is even more challenging. It will become inconceivable to rewrite/retune each application for each new parallel hardware platform generation to exploit them. Therefore, apart from the initial development of parallel applications, the major challenge for the next decade will be to *efficiently* run parallel applications on hardware architectures radically different from their original hardware target.

3.4.3. The need for performance on sequential code sections

3.4.3.1. Most software will exhibit substantial sequential code sections

For the foreseeable future, the majority of applications will feature important sequential code sections.

First, many legacy codes were developed for uniprocessors. Most of these codes will not be completely redeveloped as parallel applications, but will evolve to applications using parallel sections for the most compute-intensive parts. Second, the overwhelming majority of the programmers have been educated to program in a sequential programming style. Parallel programming is much more difficult, time consuming and error prone than sequential programming. Debugging and maintaining a parallel code is a major issue. Investing in the development of a parallel application will not be cost-effective for the vast majority of software developments. Therefore, sequential programming style will continue to be dominant in the foreseeable future. Most developers will rely on the compiler to parallelize their application and/or use some software components from parallel libraries.

3.4.3.2. Future parallel applications will require high performance sequential processing on 1000's cores chip

With the advent of universal parallel hardware in multicores, large diffusion parallel applications will have to run on a broad spectrum of parallel hardware platforms. They will be used by non-expert users who will not be able to tune the application environment to optimize performance. They will be executed concurrently with other processes which may be interactive.

The variety of possible hardware platforms, the lack of expertise of the end-user and the varying run-time execution environments are major difficulties for parallel applications. This tends to constrain the programming style and therefore reinforces the sequential structure of the control of the application.

Therefore, *most future parallel applications will rely on a single main thread or a few main threads in charge of distinct functionalities of the application. Each main thread will have a general sequential control and can initiate and control the parallel execution of parallel tasks.*

In 1967, Amdahl [50] pointed out that, if only a portion of an application is accelerated, the execution time cannot be reduced below the execution time of the residual part of the application. Unfortunately, even highly parallelized applications exhibit some residual sequential part. For parallel applications, this indicates that the effective performance of the future 1000's cores chip will significantly depend on their ability to be efficient on the execution of the control portions of the main thread as well as on the execution of sequential portions of the application.

3.4.3.3. The success of 1000's cores architecture will depend on single thread performance

While the current emphasis of computer architecture research is on the definition of scalable multi- many- core architectures for highly parallel applications, we believe that the success of the future 1000-core architecture will depend not only on their performance on parallel applications including sequential sections, but also on their performance on single thread workloads.

3.5. Performance evaluation/guarantee

Predicting/evaluating the performance of an application on a system without explicitly executing the application on the system is required for several usages. Two of these usages are central to the research of the ALF project-team: microarchitecture research (the system to be be evaluated does not exist) and Worst Case Execution Time estimation for real-time systems (the numbers of initial states or possible data inputs is too large).

When proposing a micro-architecture mechanism, its impact on the overall processor architecture has to be evaluated in order to assess its potential performance advantages. For microarchitecture research, this evaluation is generally done through the use of cycle-accurate simulation. Developing such simulators is quite complex and microarchitecture research was helped but also biased by some popular public domain research simulators (e.g. Simplescalar [52]). Such simulations are CPU consuming and simulations cannot be run on a complete application.

Real-time systems need a different use of performance prediction; on hard real-time systems, timing constraints must be respected independently from the data inputs and from the initial execution conditions. For such a usage, the Worst Case Execution Time (WCET) of an application must be evaluated and then checked against the timing constraints. While safe and tight WCET estimation techniques and tools exist for reasonably simple embedded processors (e.g. techniques based on abstract interpretation such as [55]), accurate evaluation of the WCET of an algorithm on a complex uniprocessor system is a difficult problem. Accurately modelling data cache behavior [3] and complex superscalar pipelines are still research questions as illustrated by the presence of so-called *timing anomalies* in dynamically scheduled processors, resulting from complex interactions between processor elements (among others, interactions between caching and instruction scheduling) [59].

With the advance of multicores, evaluating / guaranteeing a computer system response time is becoming much more difficult. Interactions between processes occurs at different levels. The execution time on each core depends on the behavior of the other cores. Simulations of 1000's cores micro-architecture will be needed in order to evaluate future many-core proposals. While a few multiprocessor simulators are available for the community, these simulators cannot handle realistic 1000's cores micro-architecture. New techniques have to be invented to achieve such simulations. WCET estimations on multicore platforms will also necessitate radically new techniques, in particular, there are predictability issues on a multicore where many resources are shared; those resources include the memory hierarchy, but also the processor execution units and all the hardware resources if SMT is implemented [66].

3.6. General research directions

The overall performance of a 1000's core system will depend on many parameters including architecture, operating system, runtime environment, compiler technology and application development. In the ALF project, we will essentially focus on architecture, compiler/execution environment as well as performance predictability, and in particular WCET estimation. Moreover, architecture research, and to a smaller extent, compiler and WCET estimation researches rely on processor simulation. A significant part of the effort in ALF will be devoted to define new processor simulation techniques.

3.6.1. Microarchitecture research directions

We have identified that high performance on single threads and sequential codes is one of the key issues for enabling overall high performance on a 1000's core system and we anticipate that the general architecture of such 1000's core chip will feature many simple cores and a few very complex cores.

Therefore our research in the ALF project will focus on refining the microarchitecture to achieve high performance on single process and/or sequential code sections within the general framework of such an heteregeneous architecture. This leads to two main research directions 1) enhancing the microarchitecture of high-end superscalar processors, 2) exploiting/modifying heterogeneous multicore architecture on a single process. The temperature wall is also a major technological/architectural issue for the design of future processor chips.

3.6.1.1. Enhancing complex core microarchitecture

Research on wide issue superscalar processors was merely stopped around 2002 due to limited performance returns and the power consumption wall.

When considering a heterogeneous architecture featuring hundreds of simple cores and a few complex cores, these two obstacles will partially vanish: 1) the complex cores will represent only a fraction of the chip and a fraction of its power consumption. 2) any performance gain on (critical) sequential threads will result in a performance gain of the whole system

On the complex core, the performance of a sequential code is limited by several factors. At first, on current architectures, it is limited by the peak performance of the processor. To push back this first limitation, we will explore new microarchitecture mechanisms to increase the potential peak performance of a complex core enabling larger instruction issue width. The processor performance is also limited by control dependencies.

To push back this limitation, we will explore new branch prediction mechanisms as well as new directions for reducing branch misprediction penalties [10]. As data dependencies may strongly limit performance, we will revisit data prediction. Processor performance is also often highly dependent on the presence or absence of data in a particular level of the memory hierarchy. For the ALF multicore, we will focus on sharing the access to the memory hierarchy in order to adapt the performance of the main thread to the performance of the other cores. All these topics should be studied with the new perspective of quasi unlimited silicon budget.

3.6.1.2. Exploiting heterogeneous multicores on single process

When executing a sequential section on the complex core, the simple cores will be free. Two main research directions to exploit thread level parallelism on a sequential thread have been initiated in late 90's within the context of simultaneous multithreading and early chip multiprocessor proposals: helper threads and speculative multithreading.

Helper threads were initially proposed to improve the performance of the main threads on simultaneous multithreaded architectures [53]. The main idea of helper threads is to execute codes that will accelerate the main thread without modifying its semantic.

In many cases, the compiler cannot determine if two code sections are independent due to some unresolved memory dependency. When no dependency occurs at execution time, the code sections can be executed in parallel. Thread-Level Speculation has been proposed to exploit coarse grain speculative parallelism. Several hardware-only proposals were presented [61], but the most promising solutions integrate hardware support for software thread-level speculation [64].

In the context of future manycores, thread-level speculation and helper threads should be revisited. Many simple cores will be available for executing helper threads or speculative thread execution during the execution of sequential programs or sequential code sections. The availability of these many cores is an opportunity as well as a challenge. For example, one can try to use the simple cores to execute many different helper threads that could not be implemented within a simultaneous multithreaded processor. For thread level speculation, the new challenge is the use of less powerful cores for speculative threads. Moreover the availability of many simple cores may lead to the use of helper threads and thread level speculation at the same time.

3.6.1.3. Temperature issues

Temperature is one of the constraints that have prevented the processor clock frequency to be increased in recent years. Besides techniques to decrease the power consumption, the temperature issue can be tackled with *dynamic thermal management* [9] through techniques such as clock gating or throttling and *activity migration* [62][5].

Dynamic thermal management (DTM) is now implemented on existing processors. For high performance, processors are dimensioned according to the average situation rather than to the worst case situation. Temperature sensors are used on the chip to trigger dynamic thermal management actions, for instance thermal throttling whenever necessary. On multicores, it is possible to migrate the activity from one core to another in order to limit temperature.

A possible way to increase sequential performance is to take advantage of the smaller gate delay that comes with miniaturization, which permits in theory to increase the clock frequency. However increasing the clock frequency generally requires to increase the instantaneous power density. This is why DTM and activity migration will be key techniques to deal with Amdahl's law in future many-core processors.

3.6.2. Processor simulation research

Architecture studies, and in particular microarchitecture studies, require extensive validations through detailed simulations. Cycle accurate simulators are needed to validate the microarchitectural mechanisms.

Within the ALF project, we can distinguish two major requirements on the simulation: 1) single process and sequential code simulations 2) parallel code sections simulations.

For simulating parallel code sections, a cycle-accurate microarchitecture simulator of a 1000-core architecture will be unacceptably slow. In [6], we showed that mixing analytical modeling of the global behavior of a processor with detailed simulation of a microarchitecture mechanism allows to evaluate this mechanism. Karkhanis and Smith [56] further developed a detailed analytical simulation model of a superscalar processor. Building on top of these preliminary researches, simulation methodology mixing analytical modeling of the simple cores with a more detailed simulation of the complex cores is appealing. The analytical model of the simple cores will aim at approximately modeling the impact of the simple core execution on the shared resources (e.g. data bandwidth, memory hierarchy) that are also used by the complex cores.

Other techniques such as regression modeling [57] can also be used for decreasing the time required to explore the large space of microarchitecture parameter values. We will explore these techniques in the context of many-core simulation.

In particular, research on temperature issues will require the definition and development of new simulation tools able to simulate several minutes or even hours of processor execution, which is necessary for modeling thermal effects faithfully.

3.6.3. Compiler research directions

3.6.3.1. General directions

Compilers are keystone solutions for any approach that deals with high performance on 100+ processors systems. But general-purpose compilers try to embrace so many domains and try to serve so many constraints that they frequently fail to achieve very high performance. They need to be deeply revisited. We identify four main compiler/software related issues that must be addressed in order to allow efficient use of multi- and many-cores: 1) programming 2) resource management 3) application deployment 4) portable performance. Addressing these challenges will require to revisit parallel programming and code generation extensively.

The past of parallel programming is scattered with hundreds of parallel languages. Most of these languages were designed to program homogeneous architectures and were targeting a small and well-trained community of HPC programmers. With the new diversity of parallel hardware platforms and the new community of non-expert developers, expressing parallelism is not sufficient anymore. Resource management, application deployment and portable performance are intermingled issues that require to be addressed holistically.

As many decisions should be taken according to the available hardware, resource management cannot be separated from parallel programming. Deploying applications on various systems without having to deal with thousands of hardware configurations (different numbers of cores, accelerators, ...) will become a major concern for software distribution. The grail of parallel computing is to be able to provide portable performance on a large set of parallel machines and varying execution contexts.

Recent techniques are showing promises. Iterative compilation techniques, exploiting the huge CPU cycle count now available, can be used to explore the optimization space at compile-time. Second, machine-learning techniques can be used to automatically improve compilers and code generation strategies. Speculation can be used to deal with necessary but missing information at compile-time. Finally, dynamic techniques can select or generate at run-time the most efficient code adapted to the execution context and available hardware resources.

Future compilers will benefit from past research, but they will also need to combine static and dynamic techniques. Moreover, domain specific approaches might be needed to ensure success. The ALF research effort will focus on these static and dynamic techniques to address the multicore application development challenges.

3.6.3.2. Portability of applications and performance through virtualization

The life cycle is much longer for applications than for hardware. Unfortunately the multicore era jeopardizes the old binary compatibility recipe. Binaries cannot automatically exploit additional computing cores or new accelerators available on the silicon. Moreover maintaining backward binary compatibility on future parallel architectures will rapidly become a nightmare, applications will not run at all unless some kind of dynamic binary translation is at work.

Processor virtualization addresses the problem of portability of functionalities. Applications are not compiled to the final native code but to a target independent format. This is the purpose of languages such as Java and .NET. Bytecode formats are often *a priori* perceived as inappropriate for performance intensive applications and for embedded systems. However, it was shown that compiling a C or C++ program to a bytecode format produces a code size similar to dense instruction sets [2]. Moreover, this bytecode representation can be compiled to native code with performance similar to static compilation [1]. Therefore processor virtualization for high performance, i.e., for languages like C or C++, provides significant advantages: 1) it simplifies software engineering with fewer tools to maintain and upgrade; 2) it allows better code readability and easier code maintenance since it avoids code specialization for specific targets using compile time macros such as #ifdef ; 3) the *execution code* deployed on the system is the execution code that has been debugged and validated, as opposed to the same *source code* has been recompiled for another platform; 4) new architectures will come with their JIT compiler. The JIT will (should) automatically take advantage of new architecture features such as SIMD/vector instructions or extra processors.

Our objective is to enrich processor virtualization to allow both functional portability and high performance using JIT at runtime, or bytecode-to-native code offline compiler. Split compilation can be used to annotate the bytecode with relevant information that can be helpful to the JIT at runtime or to the bytecode to native code offline compiler. Because the first compilation pass occurs offline, aggressive analyses can be run and their outcomes encoded in the bytecode. For example, such information include vectorizability, memory references (in)dependencies, suggestions derived from iterative compilation, polyhedral analysis, or integer linear programming. Virtualization allows to postpone some optimizations to run time, either because they increase the code size and would increase the cost of an embedded system or because the actual hardware platform characteristics are unknown.

3.6.4. Performance predictability for real-time systems

While compiler and architecture research efforts often focus on maximizing average case performance, applications with real-time constraints do not need only high performance but also performance guarantees in all situations, including the worst-case situation. Worst-Case Execution Time estimates (WCET) need to be upper bounds of any possible execution time. The safety level required depends on the criticality of applications: missing a frame on a video in the airplane for passenger in seat 20B is less critical than a safety critical decision in the control of the airplane.

Within the ALF project, our objective is to study performance guarantees for both (i) sequential codes running on complex cores ; (ii) parallel codes running on the multicores. This results in two quite distinct problems.

For sequential code executing on a single core, one can expect that, in order to provide real-time possibility, the architecture will feature an execution mode where a given processor will be guaranteed to access a fixed portion of the shared resources (caches, memory bandwidth). Moreover, this guaranteed share could be optimized at compile time to enforce the respect of the time constraints. However, estimating the WCET of an application on a complex micro-architecture is still a research challenge. This is due to the complex interaction of micro-architectural elements (superscalar pipelines, caches, branch prediction, out-of-order execution) [59]. We will continue to explore pure analytical and static methods. However when accurate static hardware modeling methods cannot handle the hardware complexity, new probabilistic methods [58] might be needed to explore to obtain as safe as possible WCET estimates.

Providing performance guarantees for parallel applications executed on a multicore is a new and challenging issue. Entirely new WCET estimation methods have to be defined for these architectures to cope with dynamic resource sharing between cores, in particular on-chip memory (either local memory or caches) are shared, but also buses, network-on-chip and the access to the main memory. Current pure analytical methods are too pessimistic at capturing interferences between cores [67], therefore hardware-based or compiler methods such as [65] have to be defined to provide some degree of isolation between cores. Finally, similarly to simulation methods, new techniques to reduce the complexity of WCET estimation will be explored to cope with manycore architectures.

4. Application Domains

4.1. Any computer usage

The ALF 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 impacts 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

Pierre Michaud won the 2nd Data Prefetching Championship held in conjunction with ISCA 2015 (Portland, June 2015).

BEST PAPER AWARD:

[27]
P. MICHAUD. *A Best-Offset Prefetcher*, in "2nd Data Prefetching Championship", Portland, United States, June 2015, https://hal.inria.fr/hal-01165600

6. New Software and Platforms

6.1. ATC

Address Trace Compression
KEYWORDS: Compressing - Decompressing - Address traces
FUNCTIONAL DESCRIPTION

ATC is a utility and a C library for compressing/decompressing address traces. It implements a new lossless transformation, Bytesort, that exploits spatial locality in address traces. ATC leverages existing general-purpose compressors such as gzip and bzip2. ATC also provides a lossy compression mode that yields higher compression ratios while preserving certain important characteristics of the original trace.

- Participant: Pierre Michaud
- Contact: Pierre Michaud
- URL: https://team.inria.fr/alf/software/atc/

6.2. ATMI

Modeling microprocessor 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.

We have developed an ad hoc temperature model, ATMI (Analytical model of Temperature in MIcroprocessors), 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.

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

6.3. Barra

Modelisation of a GPU architecture
KEYWORDS: Simulator - GPU - Computer architecture
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 simulates CUDA programs at the assembly language level (Tesla ISA). Its ultimate goal is to provide a 100 % bit-accurate simulation, offering bug-for-bug compatibility with NVIDIA G80-based GPUs. It works directly with CUDA executables, neither source modification nor recompilation is required.

Barra is primarily intended as a tool for research in computer architecture, although it can also be used to debug, profile and optimize CUDA programs at the lowest level.

- Participants: Sylvain Collange, David Defour, Alexandre Kouyoumdjian and Fabrice Mouhartem
- Contact: Sylvain Collange
- URL: http://barra.gforge.inria.fr/

6.4. HEPTANE

Static analyser of Worst-Case Execution Time
KEYWORD: WCET
FUNCTIONAL DESCRIPTION

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.
Status: Registered with APP (Agence de Protection des Programmes). Available under GNU General Public License v3, with number IDDN.FR.001.510039.000.S.P.2003.000.10600.

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

6.5. If-memo

KEYWORD: Performance, function memoization, dynamic optimization
Status: Ongoing development, early prototype. Registered with APP (Agence de Protection des Programmes) under number IDDN.FR.001.250013.000.S.P.2015.000.10800.

SCIENTIFIC DESCRIPTION

Memoization is the technique of saving result of executions so that future executions can be omitted when the inputs repeat. Memoization has been proposed in previous literature at the instruction level, basic block level and function level using hardware as well as pure software level approaches including changes to programming language.

We proposed software memoization of pure functions for procedural languages. We rely on the operating system loader, taking advantage of the LD_PRELOAD feature of UNIX systems. By setting this variable to the path of a shared library, we instruct the loader to first look to missing symbols in that library. Our library redefines the functions we wish to intercept. The interception code is very straightforward: it receives the same parameter as the target function and checks in a table (a software cache) if this value is readily available. In the favorable case, the result value is immediately returned. Otherwise, we invoke the original function, and store the result in the cache before returning it.

Our technique does not require the availability of source code and thus can be applied even 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. We validated If-memo with x86-64 platform using both GCC and icc compiler tool-chains, and ARM cortex-A9 platform using GCC.

- Participants: Erven Rohou and Arjun Suresh
- Contact: Erven Rohou

6.6. Padrone

KEYWORDS: Legacy code - Optimization - Performance analysis - Dynamic Optimization
Status: Registered with APP (Agence de Protection des Programmes) under number IDDN.FR.001.250013.000.S.P.2015.000.108

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: Erven Rohou and Emmanuel Riou
- Contact: Erven Rohou
- https://team.inria.fr/alf/software/Padrone/

6.7. STiMuL

Steady temperature in Multi-Layers components
FUNCTIONAL DESCRIPTION

STiMuL is a C library for modeling steady-state heat conduction in microprocessors. It can be used to obtain temperature from power density or power density from temperature. It can also be used to model stacked dies. STiMuL does not model time-varying temperature. For time-varying temperature, other models must be used, such as ATMI.

- Participant: Pierre Michaud
- Contact: Pierre Michaud
- URL: https://team.inria.fr/alf/software/stimul/

6.8. TPCalc

Throughput calculator
KEYWORDS: Architecture - Performance analysis
FUNCTIONAL DESCRIPTION

TPCalc is a throughput calculator for microarchitecture studies concerned with multi-program workloads consisting of sequential programs. Because microarchitecture simulators are slow, it is difficult to simulate throughput experiments where a multicore executes many jobs that enter and leave the system. The usual practice of measuring instantaneous throughput on independent coschedules chosen more or less randomly is not a rigorous practice because it assumes that all the coschedules are equally important, which is not always true. TPCalc can compute the average throughput of a throughput experiment without actually doing the throughput experiment. The user first defines the workload heterogeneity (number of different job types), the multicore configuration (number of cores and symmetries). TPCalc provides a list of base coschedules. The user then simulates these coschedules, using some benchmarks of his choice, and feeds back to TPCalc the measured execution rates (e.g., instructions per cycle or instructions per second).TPCalc eventually outputs the average throughput.

- Participant: Pierre Michaud
- Partner: Ghent University
- Contact: Pierre Michaud
- URL: http://www.irisa.fr/alf/downloads/michaud/tpcalc.html

6.9. tiptop

KEYWORDS: Performance, hardware counters, analysis tool.
SCIENTIFIC DESCRIPTION

Status: Registered with APP (Agence de Protection des Programmes). Available under GNU General Public License v2, with number IDDN.FR.001.450006.000.S.P.2011.000.10800. Current version is 2.3, released June 2015.

Tiptop is a new simple and flexible user-level tool that collects hardware counter data on Linux platforms (version 2.6.31+). Tiptop has been integrated in major Linux distributions, such as Fedora, Debian, Ubuntu. FUNCTIONAL DESCRIPTION The goal is to make the collection of performance and bottleneck data as simple as possible, including simple installation and usage. In particular, we stress the following points.

- 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.
- No privilege is required, any user can run *tiptop* — non-privileged users can only watch processes they own, ability to monitor anybody's process opens the door to side-channel attacks.
- The usage is similar to *top*. There is no need for the source code of the applications of interest, making it possible to monitor proprietary applications or libraries. And since there is no probe to insert in the application, understanding of the structure and implementation of complex algorithms and code bases is not required.
- Applications do not need to be restarted, and monitoring can start at any time (obviously, only events that occur after the start of *tiptop* are observed).
- Events can be counted per thread, or per process.
- Any expression can be computed, using the basic arithmetic operators, constants, and counter values.
- A configuration file lets users define their prefered setup, as well as custom expressions.
- Participant: Erven Rohou
- Contact: Erven Rohou
- URL: http://tiptop.gforge.inria.fr

6.10. Parasuite

Participants: Sylvain Collange, Thibault Person, Erven Rohou, André Seznec.

Parasuite: parallel benchmarks for multi-core CPUs, clusters and accelerators

Despite the ubiquity of parallel architectures in all computing segments, the research community often lacks benchmarks representative of parallel applications. The Inria Parallel Benchmark Suite (Parasuite) seeks to address this need by providing a set of representative parallel benchmarks for the architecture, compiler and system research communities. Parasuite targets the main contemporary parallel programming technologies: shared-memory multi-thread parallelism for multi-core, message-passing parallelism for clusters and fine-grained data-level parallelism for GPU architectures and SIMD extensions.

All benchmarks come with input datasets of various sizes, to accommodate use cases ranging from microarchitecture simulation to large-scale performance evaluation. Correctness checks on the computed results enable automated regression testing. In order to support computer arithmetic optimization and approximate computing research scenarios, the correctness checks favor accuracy metrics evaluating domain-specific relevance rather than bit-exact comparisons against an arbitrary reference output.

Visit http://parasuite.inria.fr/

7. New Results

7.1. Processor Architecture

Participants: Pierre Michaud, Bharath Narasimha Swamy, Sylvain Collange, Erven Rohou, André Seznec, Arthur Perais, Surya Khizakanchery Natarajan, Sajith Kalathingal, Tao Sun, Andrea Mondelli, Aswinkumar Sridharan, Biswabandan Panda, Fernando Endo.

Processor, cache, locality, memory hierarchy, branch prediction, multicore, power, temperature

Multicore processors have now become mainstream for both general-purpose and embedded computing. Instead of working on improving the architecture of the next generation multicore, with the DAL project, we deliberately anticipate the next few generations of multicores. While multicores featuring 1000s of cores might become feasible around 2020, there are strong indications that sequential programming style will continue to be dominant. Even future mainstream parallel applications will exhibit large sequential sections. Amdahl's law indicates that high performance on these sequential sections is needed to enable overall high performance on the whole application. On many (most) applications, the effective performance of future computer systems using a 1000-core processor chip will significantly depend on their performance on both sequential code sections and single threads.

We envision that, around 2020, the processor chips will feature a few complex cores and many (maybe 1000's) simpler, more silicon and power effective cores.

In the DAL research project, https://team.inria.fr/alf/members/andre-seznec/defying-amdahls-law-dal/, we explore the microarchitecture techniques that will be needed to enable high performance on such heterogeneous processor chips. Very high performance will be required on both sequential sections, -legacy sequential codes, sequential sections of parallel applications-, and critical threads on parallel applications, -e.g. the main thread controlling the application. Our research focuses essentially on enhancing single process performance.

7.1.1. Microarchitecture

7.1.1.1. Branch prediction

Participant: André Seznec.

This research was done in collaboration with Joshua San Miguel and Jorge Albericio from University of Toronto

The most efficient branch predictors proposed in academic literature exploit both global branch history and local branch history. However, local history branch predictor components introduce major design challenges, particularly for the management of speculative histories. Therefore, most effective hardware designs use only global history components and very limited forms of local histories such as a loop predictor. The wormhole (WH) branch predictor was recently introduced to exploit branch outcome correlation in multidimensional loops. For some branches encapsulated in a multidimensional loop, their outcomes are correlated with those of the same branch in neighbor iterations, but in the previous outer loop iteration. Unfortunately, the practical implementation of the WH predictor is even more challenging than the implementation of local history predictors.

In [36], we introduce practical predictor components to exploit this branch outcome correlation in multidimensional loops: the IMLI-based predictor components. The iteration index of the inner most loop in an application can be efficiently monitored at instruction fetch time using the Inner Most Loop Iteration (IMLI) counter. The outcomes of some branches are strongly correlated with the value of this IMLI counter. A single PC+IMLI counter indexed table, the IMLI-SIC table, added to a neural component of any recent predictor (TAGE-based or perceptron-inspired) captures this correlation. Moreover, using the IMLI counter, one can efficiently manage the very long local histories of branches that are targeted by the WH predictor. A second IMLI-based component, IMLI-OH, allows for tracking the same set of hard-to-predict branches as WH. Managing the speculative states of the IMLI-based predictor components is quite simple. Our experiments show that augmenting a state-of-the-art global history predictor with IMLI components outperforms previous state-of-the-art academic predictors leveraging local and global history at much lower hardware complexity (i.e., smaller storage budget , smaller number of tables and simpler management of speculative states).

7.1.1.2. Revisiting Value Prediction

Participants: Arthur Perais, André Seznec.

Value prediction was proposed in the mid 90's to enhance the performance of high-end microprocessors. The research on Value Prediction techniques almost vanished in the early 2000's as it was more effective to increase the number of cores than to dedicate some silicon area to Value Prediction. However high end processor chips currently feature 8-16 high-end cores and the technology will allow to implement 50-100 of such cores on a single die in a foreseeable future. Amdahl's law suggests that the performance of most workloads will not scale to that level. Therefore, dedicating more silicon area to value prediction in high-end cores might be considered as worthwhile for future multicores.

At a first step, we showed that all predictors are amenable to very high accuracy at the cost of some loss on prediction coverage [7]. This greatly diminishes the number of value mispredictions and allows to delay validation until commit-time. As such, no complexity is added in the out-of-order engine because of VP (save for ports on the register file) and pipeline squashing at commit-time can be used to recover.

This allows to leverage the possibility of validating predictions at commit to introduce a new microarchitecture, EOLE [19]. EOLE features *Early Execution* to execute simple instructions whose operands are ready in parallel with Rename and *Late Execution* to execute simple predicted instructions and high confidence branches just before Commit. EOLE depends on Value Prediction to provide operands for *Early Execution* and predicted instructions for *Late Execution*. However, Value Prediction requires EOLE to become truly practical. That is, EOLE allows to reduce the out-of-order issue-width by 33% without impeding performance. As such, the number of ports on the register file diminishes. Furthermore, optimizations of the register file such as *banking* further reduce the number of required ports. Overall EOLE possesses a register file whose complexity is on-par with that of a regular wider-issue superscalar while the out-of-order components (scheduler, bypass) are greatly simplified. Moreover, thanks to Value Prediction, speedup is obtained on many benchmarks of the SPEC'00/'06 suite.

However complexity in the value predictor infrastructure itself is also problematic. First, multiple predictions must be generated each cycle, but multi-ported structures should be avoided. Second, the predictor should be small enough to be considered for implementation, yet coverage must remain high enough to increase performance. In [32], to address these remaining concerns, we first propose a block-based value prediction scheme mimicking current instruction fetch mechanisms, BeBoP. It associates the predicted values with a fetch block rather than distinct instructions. Second, to remedy the storage issue, we present the Differential VTAGE predictor. This new tightly coupled hybrid predictor covers instructions predictable by both VTAGE and Stride-based value predictors, and its hardware cost and complexity can be made similar to those of a modern branch predictor. Third, we show that block-based value prediction allows to implement the checkpointing mechanism needed to provide D-VTAGE with last computed/predicted values at moderate cost. Overall, we establish that EOLE with a 32.8KB block-based D-VTAGE predictor and a 4-issue OoO engine can significantly outperform a baseline 6-issue superscalar processor, by up to 62.2 % and 11.2 % on average (gmean), on our benchmark set.

The overall study on value prediction is presented in Arthur Perais's PhD [14].

7.1.1.3. Cost-Effective Speculative Scheduling in High Performance Processors

Participants: André Seznec, Arthur Perais, Pierre Michaud.

This study was done in collaboration with Andreas Sembrant and Erik Hagersten from Upsala University

To maximize performance, out-of-order execution processors sometimes issue instructions without having the guarantee that operands will be available in time; e.g. loads are typically assumed to hit in the L1 cache and dependent instructions are issued assuming a L1 hit. This form of speculation ?that we refer to as speculative scheduling? has been used for two decades in real processors, but has received little attention from the research community. In particular, as pipeline depth grows and the distance between the Issue and the Execute stages increases, it becomes critical to issue dependents on variable-latency instructions as soon as possible, rather than to wait for the actual cycle at which the result becomes available. Unfortunately, due to the uncertain nature of speculative scheduling, the scheduler may wrongly issue an instruction that will not have its source(s) on the bypass network when it reaches the Execute stage. Therefore, this instruction must be canceled and replayed, which can potentially impair performance and increase energy consumption.

In [31] we focus on ways to reduce the number of replays that are agnostic of the replay scheme. First, we propose an easily implementable, low-cost solution to reduce the number of replays caused by L1 bank conflicts. Schedule Shifting always assumes that, given a dual-load issue capacity, the second load issued in a given cycle will be delayed because of a bank conflict. Its dependents are thus always issued with a corresponding delay. Second, we also improve on existing L1 hit/miss prediction schemes by taking into account instruction criticality. That is, for some criterion of criticality and for loads whose hit/miss behavior is hard to predict, we show that it is more cost-effective to stall dependents if the load is not predicted critical. In total, in our experiments assuming a 4-cycle issue-to-execute delay, we found that the vast majority of instructions replays due to L1 data cache banks conflicts and L1 hit mispredictions can be avoided, thus leading to a 3.4% performance gain and a 13.4% decrease in the number of issued instructions, over a baseline speculative scheduling scheme.

7.1.1.4. Criticality-aware Resource Allocation in OOO Processors

Participants: André Seznec, Arthur Perais, Pierre Michaud.

This study was done in collaboration with Andreas Sembrant, Erik Hagersten, David Black-Schaffer and Trevor Carlson from Upsala University.

Modern processors employ large structures (IQ, LSQ, register file, etc.) to expose instruction-level parallelism (ILP) and memory-level parallelism (MLP). These resources are typically allocated to instructions in program order. This wastes resources by allocating resources to instructions that are not yet ready to be executed and by eagerly allocating resources to instructions that are not part of the application's critical path. In [35], we explore the possibility of allocating pipeline resources only when needed to expose MLP, and thereby enabling a processor design with significantly smaller structures, without sacrificing performance. First we identify the classes of instructions that should not reserve resources in program order and evaluate the potential performance gains we could achieve by delaying their allocations. We then use this information to "park" such instructions in a simpler, and therefore more efficient, Long Term Parking (LTP) structure. The LTP stores instructions until they are ready to execute, without allocating pipeline resources, and thereby keeps the pipeline available for instructions that can generate further MLP. LTP can accurately and rapidly identify which instructions to park, park them before they execute, wake them when needed to preserve performance, and do so using a simple queue instead of a complex IQ. We show that even a very simple queue-based LTP design allows us to significantly reduce IQ (64 $\rightarrow$ 32) and register file (128 $\rightarrow$ 96) sizes while retaining MLP performance and improving energy efficiency.

7.1.1.5. Efficient Execution on Guarded Instruction Sets

Participant: André Seznec.

ARM ISA based processors are no longer low complexity processors. Nowadays, ARM ISA based processor manufacturers are struggling to implement medium-end to high-end processor cores which implies implementing a state-of-the-art out-of-order execution engine. Unfortunately providing efficient out-of-order execution on legacy ARM codes may be quite challenging due to guarded instructions.

Predicting the guarded instructions addresses the main serialization impact associated with guarded instructions execution and the multiple definition problem. Moreover, guard prediction allows to use a global branch-and-guard history predictor to predict both branches and guards, often improving branch prediction accuracy. Unfortunately such a global branch-and-guard history predictor requires the systematic use of guard predictions. In that case, poor guard prediction accuracy would lead to poor overall performance on some applications.

Building on top of recent advances in branch prediction and confidence estimation, we propose a hybrid branch and guard predictor, combining a global branch history component and global branch-and-guard history component. The potential gain or loss due to the systematic use of guard prediction is dynamically evaluated at run-time. Two computing modes are enabled: systematic guard prediction and high confidence only guard prediction. Our experiments show that on most applications, an overwhelming majority of guarded instructions are predicted. Therefore a relatively inefficient but simple hardware solution can be used to execute the few unpredicted guarded instructions. Significant performance benefits are observed on most applications while applications with poorly predictable guards do not suffer from performance loss [8].

This study was accepted to ACM Transactions on Architecture and Compiler Optimizations (Dec. 2014) and presented at the HIPEAC conference in January 2015.

7.1.1.6. Clustered microarchitecture

Participants: Andrea Mondelli, Pierre Michaud, André Seznec.

In the last 10 years, the clock frequency of high-end superscalar processors did not increase significantly. Performance keeps being increased mainly by integrating more cores on the same chip and by introducing new instruction set extensions. However, this benefits only to some applications and requires rewriting and/or recompiling these applications. A more general way to increase performance is to increase the IPC, the number of instructions executed per cycle.

In [18], we argue that some of the benefits of technology scaling should be used to increase the IPC of future superscalar cores. Starting from microarchitecture parameters similar to recent commercial high-end cores, we show that an effective way to increase the IPC is to increase the issue width. But this must be done without impacting the clock cycle. We propose to combine two known techniques: clustering and register write specialization. The objective of past work on clustered microarchitecture was to allow a higher clock frequency while minimizing the IPC loss. This led researchers to consider narrow-issue clusters. Our objective, instead, is to increase the IPC without impacting the clock cycle, which means wide-issue clusters. We show that, on a wide-issue dual cluster, a very simple steering policy that sends 64 consecutive instructions to the same cluster, the next 64 instructions to the other cluster, and so on, permits tolerating an inter-cluster delay of several cycles. We also propose a method for decreasing the energy cost of sending results of one cluster to the other cluster.

7.1.1.7. Adaptive Intelligent Memory Systems

Participants: André Seznec, Aswinkumar Sridharan.

Multi-core processors employ shared Last Level Caches (LLC). This trend will continue in the future with large multi-core processors (16 cores and beyond) as well. At the same time, the associativity of this LLC tends to remain in the order of sixteen. Consequently, with large multicore processors, the number of cores that share the LLC becomes larger than the associativity of the cache itself. LLC management policies have been extensively studied for small scale multi-cores (4 to 8 cores) and associativity degree in the 16 range. However, the impact of LLC management on large multi-cores is essentially unknown, in particular when the associativity degree is smaller than the number of cores.

In [43], we introduce Adaptive Discrete and deprioritized Application PrioriTization (ADAPT), an LLC management policy addressing the large multi-cores where the LLC associativity degree is smaller than the number of cores. ADAPT builds on the use of the Footprint-number metric. Footprint-number is defined as the number of unique accesses (block addresses) that an application generates to a cache set in an interval of time. We propose a monitoring mechanism that dynamically samples cache sets to estimate the Footprint-number of applications and classifies them into discrete (distinct and more than two) priority buckets. The cache replacement policy leverages this classification and assigns priorities to cache lines of applications during cache replacement operations. Footprint-number is computed periodically to account the dynamic changes in applications behavior. We further find that de- prioritizing certain applications during cache replacement is beneficial to the overall performance. We evaluate our proposal on 16, 20 and 24-core multi-programmed workloads and discuss other aspects in detail.

[43] has been accepted for publication at the IPDPS 2016 conference.

7.1.1.8. Hardware data prefetching

Participant: Pierre Michaud.

Hardware prefetching is an important feature of modern high-performance processors. When an application's working set is too large to fit in on-chip caches, disabling hardware prefetchers may result in severe performance reduction. We propose a new hardware data prefetcher, the Best-Offset (BO) prefetcher. The BO prefetcher is an offset prefetcher using a new method for selecting the best prefetch offset taking into account prefetch timeliness. The hardware required for implementing the BO prefetcher is very simple. The BO prefetcher won the last Data Prefetching Championship [27].

A paper describing and studying the BO prefetcher has been accepted for publication at the HPCA 2016 conference.

7.1.1.9. Prediction-based superpage-friendly TLB designs

Participant: André Seznec.

This research was done in collaboration with Misel-Myrto Papadopoulou, Xin Tong and Andreas Moshovos from University of Toronto

In [30], we demonstrate that a set of commercial and scale-out applications exhibit significant use of superpages and thus suffer from the fixed and small superpage TLB structures of some modern core designs. Other processors better cope with superpages at the expense of using power-hungry and slow fully-associative TLBs. We consider alternate designs that allow all pages to freely share a single, power-efficient and fast set-associative TLB. We propose a prediction-guided multi-grain TLB design that uses a superpage prediction mechanism to avoid multiple lookups in the common case. In addition, we evaluate the previously proposed skewed TLB which builds on principles similar to those used in skewed associative caches . We enhance the original skewed TLB design by using page size prediction to increase its effective associativity. Our prediction-based multi-grain TLB design delivers more hits and is more power efficient than existing alternatives. The predictor uses a 32-byte prediction table indexed by base register values.

7.1.2. Microarchitecture Performance Modeling

7.1.2.1. Symbiotic scheduling on SMT cores and symmetric multicores

Participant: Pierre Michaud.

This research was done in collaboration with Stijn Eyerman and Wouter Rogiest from Ghent University.

When several independent tasks execute concurrently on a simultaneous multithreaded (SMT) core or on a multicore, they share hardware resources. Hence the execution rate of a task is influenced by the other tasks running at the same time. Based on this observation, Snavely and Tullsen proposed *symbiotic* scheduling, i.e., the idea that performance can be increased by co-scheduling tasks that do not stress the same shared resources [63]. They claim that, when the number of concurrent tasks exceeds the number of logical cores, symbiotic scheduling increases performance substantially. A more recent study by Eyerman and Eeckhout reached similar conclusions [54].

We have revisited symbiotic scheduling for SMT cores and symmetric multicores [22], and we obtained very modest throughput gains, which seemingly contradicts the above mentioned studies. We analyzed the reasons for this discrepancy and found that previous studies did not measure throughput but average response time. Response time reductions can be magnified by setting the job arrival rate very close to the maximum throughput, which turns a tiny throughput increase into a large response time reduction. Also, the proposed scheduling policies are approximately equivalent to scheduling the shortest jobs first, which mechanically reduces the average response time independently of any symbiosis effect.

We identified three typical situations where symbiotic scheduling yields little to no throughput gain: (1) most of the time is spent executing a single type of job, or (2) jobs' execution rates barely depend on which other jobs are running concurrently, or (3) jobs' execution rates are proportional to the fraction they get of a certain shared resource (e.g., instruction decode bandwidth in an SMT core). In our experiments, most workloads were close to one of the three situations above.

7.1.2.2. Modeling multi-threaded programs execution time in the many-core era

Participants: Surya Khizakanchery Natarajan, Bharath Narasimha Swamy, André Seznec.

Estimating the potential performance of parallel applications on the yet-to-be-designed future many cores is very speculative. The simple models proposed by Amdahl's law (fixed input problem size) or Gustafson's law (fixed number of cores) do not completely capture the scaling behaviour of a multi-threaded (MT) application leading to over estimation of performance in the many-core era. On the other hand, modeling many-core by simulation is too slow to study the applications performance. In [28], [13], we propose a more refined but still tractable, high level empirical performance model for multi-threaded applications, the Serial/Parallel Scaling (SPS) Model to study the scalability and performance of application in many-core era. SPS model learns the application behavior on a given architecture and provides realistic estimates of the performance in future many-cores. Considering both input problem size and the number of cores in modeling, SPS model can help in making high level decisions on the design choice of future many-core applications and architecture. We validate the model on the Many-Integrated Cores (MIC) xeon-phi with 240 logical cores.

7.1.2.3. Optimal cache replacement

Participant: Pierre Michaud.

This research was done in collaboration with Mun-Kyu Lee, Jeong Seop Sim and DaeHun Nyang from Inha University.

The replacement policy for a cache is the algorithm, implemented in hardware, selecting a block to evict for making room for an incoming block. This research topic has been revitalized in recent years. The MIN replacement policy, which evicts the block referenced furthest in the future, was introduced by Belady [51] and was later shown to be optimal by Mattson et al. [60]. The MIN policy is an offline policy that cannot be implemented in real processors, as it needs the knowledge of future memory accesses. Still, a possible way to improve online replacement policies would be to emulate the MIN policy, trying to use past references to predict future ones. However, the MIN policy is not intuitive, and Mattson et al.'s proof of optimality is quite involved. We believe that new intuition about the MIN policy will help microarchitects improve cache replacement policies. As a first step toward this goal, we produced a new, intuitive proof of optimality of the MIN policy [17].

7.1.3. Hardware/Software Approaches

7.1.3.1. Helper threads

Participants: Bharath Narasimha Swamy, André Seznec.

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

In [12] we present a hardware/software framework called core-tethering to support efficient helper threading on heterogeneous many-cores. Core-tethering provides a co-processor like interface to the small cores that (a) enables a large core to directly initiate and control helper execution on the helper core and (b) allows efficient transfer of execution context between the cores, thereby reducing the performance overhead of accessing small cores for helper execution. Our evaluation on a set of memory intensive programs chosen from the standard benchmark suites show that, helper threads using moderately sized small cores can significantly accelerate a larger core compared to using a hardware prefetcher alone. We also find that a small core provides a good trade-off against using an equivalent large core to run helper threads in a HMC.

In summary, despite the latency overheads of accessing prefetched cache lines from the shared L3 cache, helper thread based prefetching on small cores looks as a promising way to improve single thread performance on memory intensive workloads in HMC architectures.

This research was partially done in collaboration with Alain Ketterlin from the Inria Camus project-team in Strasbourg.

7.1.3.2. Branch Prediction and Performance of Interpreter

Participants: Erven Rohou, André Seznec, Bharath Narasimha Swamy.

Interpreters have been used in many contexts. They provide portability and ease of development at the expense of performance. The literature of the past decade covers analysis of why interpreters are slow, and many software techniques to improve them. A large proportion of these works focuses on the dispatch loop, and in particular on the implementation of the switch statement: typically an indirect branch instruction. Folklore attributes a significant penalty to this branch, due to its high misprediction rate. In [34], we revisit this assumption, considering state-of-the-art branch predictors and the three most recent Intel processor generations on current interpreters. Using both hardware counters on Haswell, the latest Intel processor generation, and simulation of the ITTAGE predictor [10], we show that the accuracy of indirect branch prediction is no longer critical for interpreters. We further compare the characteristics of these interpreters and analyze why the indirect branch is less important than before.

7.1.3.3. Augmenting superscalar architecture for efficient many-thread parallel execution

Participants: Sylvain Collange, André Seznec, Sajith Kalathingal.

Threads of Single-Program Multiple-Data (SPMD) applications often exhibit very similar control flows, i.e. they execute the same instructions on different data. In [42] we propose the Dynamic Inter-Thread Vectorization Architecture (DITVA) to leverage this implicit Data Level Parallelism on SPMD applications to create dynamic vector instructions at runtime. DITVA extends an in-order SMT processor with SIMD units with an inter-thread vectorization execution mode. In this mode, identical instructions of several threads running in lockstep are aggregated into a single SIMD instruction. DITVA leverages existing SIMD units and maintains binary compatibility with existing CPU architectures. To balance TLP and DLP, threads are statically grouped into fixed-size warps, inside which threads run in lockstep. At instruction fetch time, if the instruction streams of several threads within a warp are synchronized, then DITVA aggregates the instructions of the threads as dynamic vectors. To maximize vectorization opportunities, we use resource sharing arbitration policies that favor thread synchronization within warps. The policies do not require any compiler hints or modified algorithms for the existing SPMD applications and allow to run unmodified CPU binaries. A dynamic vector instruction is executed as a single unit. This allows to execute m identical instructions from m different threads on m parallel execution lanes while activating the I-fetch, the decode, and the overall pipeline control only once.

Our evaluation on the SPMD applications from the PARSEC and SPLASH benchmarks shows that a 4-warp 4-lane 4-issue DITVA architecture with a realistic bank-interleaved cache achieves 44% higher performance than a 4-thread 4-issue SMT architecture with AVX instructions while fetching and issuing 40 % fewer instrructions, achieving an overall 22% energy reduction.

7.2. Compiler, vectorization, interpretation

Participants: Erven Rohou, Emmanuel Riou, Bharath Narasimha Swamy, Arjun Suresh, André Seznec, Nabil Hallou, Sylvain Collange.

7.2.1. Improving sequential performance through memoization

Participants: Erven Rohou, Emmanuel Riou, Bharath Narasimha Swamy, André Seznec, Arjun Suresh.

Many applications perform repetitive computations, even when properly programmed and optimized. Performance can be improved by caching results of pure functions, and retrieving them instead of recomputing a result (a technique called memoization).

We propose [20] a simple technique for enabling software memoization of any dynamically linked pure function and we illustrate our framework using a set of computationally expensive pure functions – the transcendental functions.

Our technique does not need the availability of source code and thus can be applied even 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 tool-chains, and can work with a variety of current architectures.

We present experimental results for x86-64 platform using both gcc and icc compiler tool-chains, and for ARM cortex-A9 platform using gcc. Our experiments include a mix of real world programs and standard benchmark suites: SPEC and Splash2x. On standard benchmark applications that extensively call the transcendental functions we report memoization benefits of upto 16 %, while much higher gains were realized for programs that call the expensive Bessel functions. Memoization was also able to regain a performance loss of 76 % in *bwaves* due to a known performance bug in the gcc libm implementation of *pow* function.

This work has been published in ACM TACO 2015 [20] and accepted for presentation at the International Conference HiPEAC 2016.

7.2.2. Code Obfuscation

Participant: Erven Rohou.

This research is done in collaboration with the group of Prof. Ahmed El-Mahdy at E-JUST, Alexandria, Egypt.

We propose [24] to leverage JIT compilation to make software tamper-proof. The idea is to constantly generate different versions of an application, even while it runs, to make reverse engineering hopeless. More precisely a JIT engine is used to generate new versions of a function each time it is invoked, applying different optimizations, heuristics and parameters to generate diverse binary code. A strong random number generator will guarantee that generated code is not reproducible, though the functionality is the same.

This work was presented in January 2015 at the International Workshop on Dynamic Compilation Everywhere (DCE-2015) [24].

7.2.3. Dynamic Binary Re-vectorization

Participants: Erven Rohou, Nabil Hallou, Emmanuel Riou.

This work is done in collaboration with Philippe Clauss and Alain Ketterlin (Inria CAMUS).

Applications are often under-optimized for the hardware on which they run. Several reasons contribute to this unsatisfying situation, including the use of legacy code, commercial code distributed in binary form, or deployment on compute farms. In fact, backward compatibility of instruction sets guarantees only the functionality, not the best exploitation of the hardware. In particular SIMD instruction sets are always evolving.

We proposed [23] a runtime re-vectorization platform that dynamically adapts applications to execution hardware. The platform is built on top of Padrone. Programs distributed in binary forms are re-vectorized at runtime for the underlying execution hardware. Focusing on the x86 SIMD extensions, we are able to automatically convert loops vectorized for SSE into the more recent and powerful AVX. A lightweight mechanism leverages the sophisticated technology put in a static vectorizer and adjusts, at minimal cost, the width of vectorized loops. We achieve speedups in line with a native compiler targeting AVX. Our re-vectorizer is implemented inside a dynamic optimization platform; its usage is completely transparent to the user and requires neither access to source code nor rewriting binaries.

7.2.4. Dynamic Parallelization of Binary Executables

Participants: Erven Rohou, Nabil Hallou, Emmanuel Riou.

We address runtime automatic parallelization of binary executables, assuming no previous knowledge on the executable code. The Padrone platform is used to identify candidate functions and loops. Then we disassemble the loops and convert them to the intermediate representation of the LLVM compiler (thanks to the external tool McSema). This allows us to leverage the power of the polyhedral model for auto-parallelizing loops. Once optimized, new native code is generated just-in-time in the address space of the target process.

Our approach enables user transparent auto-parallelization of legacy and/or commercial applications with auto-parallelization.

This work is done in collaboration with Philippe Clauss (Inria CAMUS).

7.2.5. Hardware Accelerated JIT Compilation for Embedded VLIW Processors

Participant: Erven Rohou.

Just-in-time (JIT) compilation is widely used in current embedded systems (mainly because of Java Virtual Machine). When targeting Very Long Instruction Word (VLIW) processors, JIT compilation back-ends grow more complex because of the instruction scheduling phase. This tends to reduce the benefits of JIT compilation for such systems. We propose a hybrid JIT compiler where JIT management is handled in software and the back-end is performed by specialized hardware. Experimental studies show that this approach leads to a compilation up to 15 times faster and 18 times more energy efficient than a pure software compilation.

This work is done in collaboration with the CAIRN team (Steven Derrien and Simon Rokicki).

7.2.6. Performance Assessment of Sequential Code

Participant: Erven Rohou.

The advent of multicore and manycore processors, including GPUs, in the customer market encouraged developers to focus on extraction of parallelism. While it is certainly true that parallelism can deliver performance boosts, parallelization is also a very complex and error-prone task, and many applications are still dominated by sequential sections. Micro-architectures have become extremely complex, and they usually do a very good job at executing fast a given sequence of instructions. When they occasionally fail, however, the penalty is severe. Pathological behaviors often have their roots in very low-level details of the micro-architecture, hardly available to the programmer. In [33], we argue that the impact of these low-level features on performance has been overlooked, often relegated to experts. We show that a few metrics can be easily defined to help assess the overall performance of an application, and quickly diagnose a problem. Finally, we illustrate our claim with a simple prototype, along with use cases.

7.2.7. Compilers for emerging throughput architectures

Participant: Sylvain Collange.

This work is done in collaboration with Douglas de Couto and Fernando Pereira from UFMG.

The increasing popularity of Graphics Processing Units (GPUs) has brought renewed attention to old problems related to the Single Instruction, Multiple Data execution model. One of these problems is the reconvergence of divergent threads. A divergence happens at a conditional branch when different threads disagree on the path to follow upon reaching this split point. Divergences may impose a heavy burden on the performance of parallel programs.

We have proposed a compiler-level optimization to mitigate the performance loss due to branch divergence on GPUs. This optimization consists in merging function call sites located at different paths that sprout from the same branch. We show that our optimization adds negligible overhead on the compiler. When not applicable, it does not slow down programs and it accelerates substantially those in which it is applicable. As an example, we have been able to speed up the well known SPLASH Fast Fourier Transform benchmark by 11 %.

7.2.8. *Deterministic floating-point primitives for high-performance computing*

Participant: Sylvain Collange.

This work is done in collaboration with David Defour (UPVD), Stef Graillat and Roman Iakymchuk (LIP6).

Parallel algorithms such as reduction are ubiquitous in parallel programming, and especially high-performance computing. Although these algorithms rely on associativity, they are used on floating-point data, on which operations are not associative. As a result, computations become non-deterministic, and the result may change according to static and dynamic parameters such as machine configuration or task scheduling.

We introduced a solution to compute deterministic sums of floating-point numbers efficiently and with the best possible accuracy. A multi-level algorithm incorporating a filtering stage that uses fast vectorized floating-point expansions and an accumulation stage based on superaccumulators in a high-radix carry-save representation guarantees accuracy to the last bit even on degenerate cases while maintaining high performance in the common cases [16]. Leveraging these algorithms, we build a reproducible BLAS library [49] and extend the approach to triangular solvers [25].

7.3. WCET estimation and optimization

Participants: Hanbing Li, Isabelle Puaut, Erven Rohou, Damien Hardy, Viet Anh Nguyen, Benjamin Rouxel.

7.3.1. *WCET estimation for architectures with faulty caches*

Participants: Damien Hardy, Isabelle Puaut.

This is joint work with Yannakis Sazeides from University of Cyprus

Fine-grained disabling and reconfiguration of hardware elements (functional units, cache blocks) will become economically necessary to recover from permanent failures, whose rate is expected to increase dramatically in the near future. This fine-grained disabling will lead to degraded performance as compared to a fault-free execution.

Until recently, all static worst-case execution time (WCET) estimation methods were assuming fault-free processors, resulting in unsafe estimates in the presence of faults. The first static WCET estimation technique dealing with the presence of permanent faults in instruction caches was proposed in [4]. This study probabilistically quantified the impact of permanent faults on WCET estimates. It demonstrated that the probabilistic WCET (pWCET) estimates of tasks increase rapidly with the probability of faults as compared to fault-free WCET estimates.

New results show that very simple reliability mechanisms allow mitigating the impact of faulty cache blocks on pWCETs. Two mechanisms, that make part of the cache resilient to faults are analyzed. Experiments show that the gain in pWCET for these two mechanisms are on average 48% and 40% as compared to an architecture with no reliability mechanism.

This work will appear at DATE 2016.

7.3.2. Speeding up Static Probabilistic Timing Analysis

Participants: Damien Hardy, Isabelle Puaut.

This is joint work with Suzana Milutinovic, Jaume Abella, Eduardo Quinones and Francisco J. Cazorla from Barcelona Supercomputing Center.

Probabilistic Timing Analysis (PTA) has emerged recently to derive trustworthy and tight WCET estimates. For its static variant, called SPTA, we identify one of the main elements that jeopardizes its scalability to real-size programs: its high computation time cost. This SPTA's high computational costs are due to convolution, a mathematical operator used by SPTA and also deployed in many domains including signal and image processing.

In [40], we show how convolution is applied in SPTA, and qualitatively and quantitatively evaluate optimizations developed in other domains to reduce convolution time cost when applied to SPTA, and SPTA-specific optimizations. We show that SPTA-specific optimizations provide larger execution time reductions than generic cores.

7.3.3. Traceability of flow information for WCET estimation

Participants: Hanbing Li, Isabelle Puaut, Erven Rohou.

This research is part of the ANR W-SEPT project.

Control-flow information is mandatory for WCET estimation, to guarantee that programs terminate (e.g. provision of bounds for the number of loop iterations) but also to obtain tight estimates (e.g. identification of infeasible or mutually exclusive paths). Such flow information is expressed through annotations, that may be calculated automatically by program/model analysis, or provided manually.

The objective of this work is to address the challenging issue of the mapping and transformation of the flow information from high level down to machine code. In our recent work, we have proposed a framework to systematically transform flow information from source code to machine code. The framework [11] defines a set of formulas to transform flow information for standard compiler optimizations. Transforming the flow information is done within the compiler, in parallel with transforming the code. There thus is no guessing what flow information have become, it is transformed along with the code.

Our most recent results in this framework were to add support for vectorization [26]. We implemented our approach in the LLVM compiler. In addition, we show through measurements on single-path programs that vectorization improves not only average-case performance but also WCETs. The WCET improvement ratio ranges from 1.18x to 1.41x depending on the target architecture on a benchmark suite designed for vectorizing compilers (TSVC).

This work is part of a more general traceability framework, designed and implemented within the ANR W-SEPT project and described in paper [21]. In this paper, we introduce a complete semantic-aware WCET estimation workflow. We introduce some program analysis to find infeasible paths: they can be performed at design, C or binary level, and may take into account information provided by the user. We design an annotation-aware compilation process that enables to trace the infeasible path properties through the program transformations performed by the compilers. Finally, we adapt the WCET estimation tool to take into account the kind of annotations produced by the workflow.

7.3.4. WCET estimation for many core processors

Participants: Viet Anh Nguyen, Damien Hardy, Isabelle Puaut.

This research is part of the PIA Capacités project.

The overall goal of this research is to defined 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 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.

In [41], we address this issue by developing an optimal integer linear programming formulation for solving the scheduling problem, whose objective is to minimize the WCET of a parallel application. Our proposed static partitioned non-preemptive mapping strategy addresses the effect of local caches to tighten the estimated WCET of the parallel application. We report preliminary results obtained on synthetic parallel applications.

8. Bilateral Contracts and Grants with Industry

8.1. Bilateral Contracts with Industry

8.1.1. Intel research grant ALF-INTEL2014-8957

Participants: André Seznec, Fernando Endo.

Intel is supporting the research of the ALF project-team on "Mixing branch and value prediction to enable high sequential performance".

8.2. Bilateral Grants with Industry

8.2.1. Nano 2017 PSAIC

Participants: Arif Ali Ana-Pparakkal, Erven Rohou, Emmanuel Riou.

Nano 2017 PSAIC is a collaborative R&D program involving Inria and STMicroelectronics. The PSAIC (Performance and Size Auto-tuning through Iterative Compilation) project concerns the automation of program optimization through the combination of several tools and techniques such as: compiler optimization, profiling, trace analysis, iterative optimization and binary analysis/rewriting. For any given application, the objective is to devise through a fully automated process a compiler profile optimized for performance and code size. For this purpose, we are developing instrumentation techniques that can be focused and specialized to a specific part of the application aimed to be monitored.

The project involves the Inria teams ALF, AriC, CAMUS and CORSE. ALF contributes program analyses at the binary level, as well as binary transformations. We will also study the synergy between static (compiler-level) and dynamic (run-time) analyses.

9. Partnerships and Cooperations

9.1. National Initiatives

9.1.1. Capacités: Projet "Investissement d'Avenir", 1/11/14 to 31/01/2018

Participants: Damien Hardy, Isabelle Puaut.

The project objective is to develop a hardware and software platform based on manycore architectures, and to demonstrate the relevance of these manycore architectures (and more specifically the Kalray manycore) for several industrial applications. The Kalray MPPA manycore architecture is currently the only one able to meet the needs of embedded systems simultaneously requiring high performance, lower power consumption, and the ability to meet the requirements of critical systems (low latency I/O, deterministic processing times, and dependability). The project partners are Kalray (lead), Airbus, Open-Wide, Safran Sagem, IS2T, Real Time ar Work, Dassault Aviation, Eurocopter, MBDA, ProbaYes, IRIT, Onera, Verimag, Inria, Irisa, Tima and Armines.

9.1.2. Inria Project Lab: Multicore 2013-2016

Participants: Erven Rohou, Nabil Hallou.

The Inria Project Lab (formerly *Action d'Envergure*) started in 2013. It is entitled "Large scale multicore virtualization for performance scaling and portability". Partner project-teams include: ALF, ALGORILLE, CAMUS, REGAL, RUNTIME, as well as DALI. This project aims to build collaborative virtualization mechanisms that achieve essential tasks related to parallel execution and data management. We want to unify the analysis and transformation processes of programs and accompanying data into one unique virtual machine.

9.1.3. ADT IPBS 2013-2015

Participants: Sylvain Collange, Erven Rohou, André Seznec, Thibault Person.

As multi-core CPUs and parallel accelerators become pervasive, all execution platforms are now parallel. Research on architecture, compilers and systems now focuses on parallel platforms. New contributions need to be validated against parallel applications that are expected to be representative of current or future workloads. The research community relies today on a few benchmarks sets (SPLASH, PARSEC ...) Existing parallel benchmarks are scarce, and some of them have issues such as aging workloads or non-representative input sets. The IPBS initiative aims at leveraging the diversity of parallel applications developed within Inria to provide a set of benchmarks, named the Inria Parallel Benchmark Suite http://parasuite.inria.fr/, to the research community.

9.1.4. ANR Continuum 2015–2019

Participant: Erven Rohou.

The CONTINUUM project aims to address the energy-efficiency challenge in future computing systems by investigating a design continuum for compute nodes, which seamlessly goes from software to technology levels via hardware architecture. Power saving opportunities exist at each of these levels, but the real measurable gains will come from the synergistic focus on all these levels as considered in this project. Then, a cross-disciplinary collaboration is promoted between computer science and microelectronics, to achieve two main breakthroughs: i) combination of state-of-the-art heterogeneous adaptive embedded multicore architectures with emerging communication and memory technologies and, ii) power-aware dynamic compilation techniques that suitably match such a platform.

Continuum started on Oct 1st 2015. Partners are LIRMM and Cortus SAS.

9.1.5. ANR CHIST-ERA SECODE 2016-2018

Participants: Damien Hardy, Erven Rohou.

SECODE (Secure Codes to thwart Cyber-physical Attacks) was accepted, and will start on January 1st 2016.

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. Theses results are demonstrated by using a smart sensor application with hardened embedded firmware and tamper-proof hardware platform.

Partners are Télécom Paris Tech, Université Paris 8, University of Sabancı(Turkey), and Université Catholique de Louvain (Belgium).

9.1.6. ANR W-SEPT 2012-2015

Participants: Hanbing Li, Isabelle Puaut, Erven Rohou.

Critical embedded systems are generally composed of repetitive tasks that must meet drastic timing constraints, such as termination deadlines. Providing an upper bound of the worst-case execution time (WCET) of such tasks at design time is thus necessary to prove the correctness of the system. Static WCET estimation methods, although safe, may produce largely over-estimated values. The objective of the project is to produce tighter WCET estimates by discovering and transforming flow information at all levels of the software design process, from high level-design models (e.g. Scade, Simulink) down to binary code. The ANR W-SEPT project partners are Verimag Grenoble, IRIT Toulouse, Inria Rennes. A case study is provided by Continental Toulouse.

9.2. European Initiatives

9.2.1. FP7 & H2020 Projects

9.2.1.1. ANTAREX

Participant: Erven Rohou.

Title: Auto-Tuning and Adaptivity appRoach for Energy efficient exascale HPC Systems

Programm: H2020

Duration: September 2015 - September 2018

Coordinator: Politecnico di Milano, Italy (POLIMI)

Partners:

Consorzio Interuniversitario Cineca (Italy)

Dompe Farmaceutici Spa (Italy)

Eidgenoessische Technische Hochschule Zuerich (Switzerland)

Vysoka Skola Banska - Technicka Univerzita Ostrava (Czech Republic)

Politecnico di Milano (Italy)

Sygic As (Slovakia)

Universidade Do Porto (Portugal)

Inria contact: Erven Rohou

Energy-efficient heterogeneous supercomputing architectures need to be coupled with a radically new software stack capable of exploiting the benefits offered by the heterogeneity at all the different levels (supercomputer, job, node) to meet the scalability and energy efficiency required by Exascale supercomputers. ANTAREX will solve these challenging problems by proposing a disruptive holistic approach spanning all the decision layers composing the supercomputer software stack and exploiting effectively the full system capabilities (including heterogeneity and energy management). The main goal of the ANTAREX project is to provide a breakthrough approach to express application self-adaptivity at design-time and to runtime manage and autotune applications for green and heterogenous High Performance Computing (HPC) systems up to the Exascale level.

9.2.1.2. Eurolab-4-HPC

Participant: André Seznec.

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

Programm: H2020

Duration: September 2015 - September 2017

Coordinator: CHALMERS TEKNISKA HOEGSKOLA AB

Partners:

Barcelona Supercomputing Center - Centro Nacional de Supercomputacion (Spain)

Chalmers Tekniska Hoegskola (Sweden)

Ecole Polytechnique Federale de Lausanne (Switzerland)

Foundation for Research and Technology Hellas (Greece)

Universitaet Stuttgart (Germany)

Rheinisch-Westfaelische Technische Hochschule Aachen (Germany)

Technion - Israel Institute of Technology (Israel)

Universitaet Augsburg (Germany)

The University of Edinburgh (United Kingdom)

Universiteit Gent (Belgium)

The University of Manchester (United Kingdom)

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 longterm 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.

9.2.1.3. DAL

Participants: Pierre Michaud, Bharath Narasimha Swamy, Sylvain Collange, Erven Rohou, André Seznec, Arthur Perais, Surya Khizakanchery Natarajan, Sajith Kalathingal, Tao Sun, Andrea Mondelli, Aswinkumar Sridharan.

Title: DAL: Defying Amdahl's Law

Program: FP7

Type: ERC

Duration: April 2011 - March 2016

Coordinator: Inria

Inria contact: André Seznec

Multicore processors have now become mainstream for both general-purpose and embedded computing. Instead of working on improving the architecture of the next generation multicore, with the DAL project, we deliberately anticipate the next few generations of multicores. While multicores featuring 1000's of cores might become feasible around 2020, there are strong indications that sequential programming style will continue to be dominant. Even future mainstream parallel applications will exhibit large sequential sections. Amdahl's law indicates that high performance on these sequential sections is needed to enable overall high performance on the whole application. On many (most) applications, the effective performance of future computer systems using a 1000-core processor chip will significantly depend on their performance on both sequential code sections and single thread. We envision that, around 2020, the processor chips will feature a few complex cores and many (may be 1000's) simpler, more silicon and power effective cores. In the DAL research project, we will explore the microarchitecture techniques that will be needed to enable high performance on such heterogeneous processor chips. Very high performance will be required on both sequential sections -legacy sequential codes, sequential sections of parallel applications- and critical threads on parallel applications -e.g. the main thread controlling the application. Our research will focus on enhancing single process performance. On the microarchitecture side, we will explore both a radically new approach, the sequential accelerator, and more conventional processor architectures. We will also study how to exploit heterogeneous multicore architectures to enhance sequential thread performance.

9.2.1.4. ARGO

Participants: Isabelle Puaut, Damien Hardy.

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

Program: H2020

Type: RIA

Duration: Jan 2016 - Dec 2018

Coordinator: Karlsruher Institut fuer Technologie (KIT)

Université Rennes I contact: Steven Derrien

Partners:

Karlsruher Institut fuer Technologie (KIT)

SCILAB enterprises SAS

Recore Systems BV

Université de Rennes 1

Technologiko Ekpaideftiko Idryma (TEI) Dytikis Elladas

Absint GmbH

Deutsches Zentrum fuer Luft - und Raumfahrt EV

Fraunhofer

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.

9.2.2. Collaborations in European Programs, except FP7 & H2020

9.2.2.1. COST Action TACLe - Timing Analysis on Code-Level (http://www.tacle.eu) 10-2012/09-2016

Participants: Damien Hardy, Isabelle Puaut.

Embedded systems increasingly permeate our daily lives. Many of those systems are business- or safety-critical, with strict timing requirements. Code-level timing analysis (used to analyze software running on some given hardware w.r.t. its timing properties) is an indispensable technique for ascertaining whether or not these requirements are met. However, recent developments in hardware, especially multi-core processors, and in software organization render analysis increasingly more difficult, thus challenging the evolution of timing analysis techniques.

New principles for building "timing-composable" embedded systems are needed in order to make timing analysis tractable in the future. This requires improved contacts within the timing analysis community, as well as with related communities dealing with other forms of analysis such as model-checking and type-inference, and with computer architectures and compilers. The goal of this COST Action is to gather these forces in order to develop industrial-strength code-level timing analysis techniques for future-generation embedded systems, through several working groups:

- WG1 Timing models for multi-cores and timing composability
- WG2 Tooling aspects
- WG3 Early-stage timing analysis
- WG4 Resources other than time

Isabelle Puaut is in the management committee of the COST Action TACLe - Timing Analysis on Code-Level (http://www.tacle.eu). She is responsible of Short Term Scientific Missions (STSM) within TACLe.

9.2.3. Collaborations with Major European Organizations

9.2.3.1. HiPEAC3 NoE

Participants: Pierre Michaud, Erven Rohou, André Seznec.

P. Michaud, A. Seznec and E. Rohou are members of the European Network of Excellence HiPEAC3. HiPEAC3 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. International Initiatives

9.3.1. Inria Associate Teams not involved in an Inria International Labs

9.3.1.1. PROSPIEL

Title: Profiling and specialization for locality

International Partner (Institution - Laboratory - Researcher):

Universidade Federal de Minas Gerais (Brazil) - Dpt of Computer Science - Fernando Magno Quintao Pereira

Start year: 2015

See also: https://team.inria.fr/alf/prospiel/

The PROSPIEL project aims at optimizing parallel applications for high performance on new throughput-oriented architectures: GPUs and many-core processors. Traditionally, code optimization is driven by a program analysis performed either statically at compile-time, or dynamically at run-time. Static program analysis is fully reliable but often over-conservative. Dynamic analysis provides more accurate data, but faces strong execution time constraints and does not provide any guarantee. By combining profiling-guided specialization of parallel programs with runtime checks for correctness, PROSPIEL seeks to capture the advantages of both static analysis and dynamic analysis. The project relies on the polytope model, a mathematical representation for parallel loops, as a theoretical foundation. It focuses on analyzing and optimizing performance aspects that become increasingly critical on modern parallel computer architectures: locality and regularity.

9.3.2. Inria International Partners

9.3.2.1. Informal International Partners

The ALF project-team has informal collaborations (visits, common publications) with University of Wisconsin at Madison (Pr Wood), University of Toronto (Pr Moshovos), University of Ghent (Dr Eyerman), University of Upsalla (Pr Hagersten), University of Cyprus (Pr Sazeides), the Egyptian-Japanese University of Science and Technology (Pr Ahmed El-Mahdy).

9.3.3. Participation In other International Programs

9.3.3.1. UFMG Chair (Brazil)

Program: Cátedras Francesas UFMG

Title: A language runtime with fault-resiliency for approximate computing

Inria principal investigator: Sylvain Collange

International Partner (Institution - Laboratory - Researcher):

Universidade Federal de Minas Gerais (UFMG) - Computer Science Department - Fernando Pereira

Duration: Sep 2015 - Oct 2015

In this project we propose to implement fault tolerance at the runtime level within a virtual machine for a managed language. Our approach consists in developing a just-in-time compiler analysis that identifies and extracts side-effect free computations, such as pure functions, within the code. For each of these computations, an approximate implementation will be generated in addition to the regular native code. When the computation is invoked during execution, the runtime will first execute the approximate implementation. In case the quality or accuracy of the result is not sufficient at the time it is needed, the runtime will transparently re-execute the computation in exact mode.

9.4. International Research Visitors

9.4.1. Visits to International Teams

9.4.1.1. Explorer programme

Perais Arthur

Date: Jan 2015 - Apr 2015

Institution: Carnegie Mellon University (United States)

9.4.1.2. Research stays abroad

Sylvain Collange has been invited on a professor chair at Universidade Federal de Minas Gerais, Brazil (September-October 2015).

10. Dissemination

10.1. Promoting Scientific Activities

10.1.1. Scientific events organisation

10.1.1.1. General chair, scientific chair

- André Seznec is the PC chair for ISCA 2016.
- Isabelle Puaut and Damien Hardy have organized the Ecole d'été Temps Réel in Rennes in August 2015.
- Isabelle Puaut is the chair of the RTNS steering committee.

10.1.1.2. Member of the organizing committees

- Isabelle Puaut is a member of the Executive committee of IEEE Technical Committee on Real-Time Systems
- Isabelle Puaut is member of the Steering committees of ECRTS and WCET conferences.
- Sylvain Collange was a member of the organization committee of Rencontres Arithmétiques de l'Informatique Mathématique (RAIM) 2015 in Rennes.

10.1.2. Scientific events selection

10.1.2.1. Chair of conference program committees

- André Seznec is the PC chair for ISCA 2016.

10.1.2.2. Member of the conference program committees

- Isabelle Puaut was a member of ECRTS 2015, RTAS 2015, RTCSA 2015 RTNS 2015 and WCET 2015 program committees. She is a member of the ECRTS 2016 program committee.
- Damien Hardy was a member of RTNS 2015 and WCET 2015 program committees
- André Seznec was a member of the MICRO 2015 and the SAMOS 2015 program committees.
- Pierre Michaud was a member of the program committee of the OMHI 2015 workshop.
- Erven Rohou was a member of the program committee of DITAM-PARMA and ISPA 2015.
- Sylvain Collange is a member of the ISCA 2016 program committee.

10.1.2.3. Reviewer

ALF members have been reviewers for many conferences and journals.

10.1.3. Journal

10.1.3.1. Member of the editorial boards

- André Seznec is a member of the editorial board of IEEE Micro and an associate editor of ACM TACO.

10.1.3.2. Reviewer - Reviewing activities

ALF members have been reviewers for many conferences and journals.

10.1.4. Invited talks

- Isabelle Puaut. "Multicore timing verification: towards more integrated WCET and schedulability analysis. TACLe COST action Focussed meeting on timing analysis of parallel programs in many-core architectures". Porto, Portugal, Nov. 2015.
- André Seznec. "Skewed Compressed Cache", AMD, Sunnyvale, Aug. 2015
- Arthur Perais. "Cost-Effective Speculative Scheduling in High Performance Processors" Intel, Hillsboro June 2015
- Damien Hardy . "Worst Case Execution Time Estimation and Permanent Faults" at "9ièmes rencontres de la communauté française de compilation", Jan. 2015.
- Erven Rohou. "Branch Prediction and the Performance of Interpreters: Don't Trust Folklore", at Dixièmes rencontres de la communauté française de compilation, Sep 2015.
- Erven Rohou. "Traceability of Flow Information: Reconciling Compiler Optimizations and WCET Estimation", at Neuvièmes rencontres de la communauté française de compilation, Jan 2015.

10.1.5. Scientific expertise

André Seznec was member of the ERC Advanced grants panel.

10.1.6. Research administration

- Erven Rohou is a member of the Inria CDT (Commission du Développement Technologique)
- As “correspondant scientifique des relations internationales” for Inria Rennes Bretagne Atlantique, Erven Rohou is a member of the Inria COST GTRI (Groupe de Travail "Relations Internationales" du Comité d’Orientation Scientifique et Technologique).
- A. Seznec is an elected member of the administration board of Inria since November 2014.
- Erven Rohou was a member of the Inria CR2 recruitment committee.
- Erven Rohou was a member of the recruitment committee of an assistant professor at Uppsala University (Sweden).
- Isabelle Puaut is member of the ISTIC council.

10.2. Teaching - Supervision - Juries

10.2.1. Teaching

Master research : A. Seznec, E.Rohou, I. Puaut, Performance et Microarchitecture, 24 hours, M2, Université de Rennes I, France

Master research: I. Puaut, E. Rohou, Rédaction d’articles scientifiques, 18 hours, M2, Université de Rennes I, France

Master: I. Puaut, D. Hardy, Operating systems, 130 hours, M1, Université de Rennes I, France

Master: I. Puaut, D. Hardy, Systèmes temps-réel, 69 hours, M1, Université de Rennes I, France

Licence: D. Hardy, Informatique temps-réel, 40 hours, L3, Université de Rennes I, France

Master: D. Hardy Systèmes d’exploitation, 40 hours, M1, Université de Rennes I, France

Master: D. Hardy Systèmes d’exploitation, 60 hours, M2 CCI, Université de Rennes I, France

Master: S. Collange, Programmation parallèle, 22 hours, M1, Université de Rennes I, France

Master/PhD: S. Collange, GPU programming, 30 hours, MSc/PhD programs, Universidade Federal de Minas Gerais, Brazil

10.2.2. Supervision

HdR: Erven Rohou, Infrastructures and Compilation Strategies for the Performance of Computing Systems, Université Rennes 1, Nov 2015

PhD : Hanbing Li, Extraction and Traceability of Annotations for WCET Estimation, Université Rennes 1, Oct 2015, co-advisors E. Rohou and I. Puaut

PhD : Arthur Perais, "Increasing the performance of superscalar processors through value prediction", Université Rennes 1, Sept 2015, advisor A. Seznec

PhD : Surya Khizakanchery Natarajan,"Modeling performance of serial and parallel sections of multi-threaded programs in manycore era", Université Rennes 1, June 2016, advisor A. Seznec

PhD : Bharath Narasimha Swamy, "Exploiting heterogeneous manycores on sequential code", Université Rennes 1, March 2015, advisor A. Seznec

PhD in progress: Nabil Hallou, Université Rennes 1, Feb 2013, co-advisors E. Rohou and P. Clauss (EPI Camus Inria Strasbourg)

PhD in progress: Sajith Kalathingal, Université Rennes 1, Dec 2012, co-advisors S. Collange and A. Seznec

PhD in progress: Andrea Mondelli, Université Rennes 1, Oct 2013, co-advisors P. Michaud and A. Seznec

PhD in progress: Aswinkumar Sridharan, Université Rennes 1, Oct 2013, advisor A. Seznec

PhD in progress: Arjun Suresh , Université Rennes 1, Dec 2012, co-advisors E. Rohou and A. Seznec

PhD in progress : Rabab Bouziane, Université Rennes 1, Nov 2015, advisor E. Rohou

PhD in progress : Arif Ali Ana-Pparakkal, Université Rennes 1, Feb 2015, advisor E. Rohou

PhD in progress : Simon Rokicki, Université Rennes 1, Sep 2015, co-advisors E. Rohou and Steven Derrien (CAIRN)

PhD in progress: Benjamin Rouxel, Université Rennes 1, Oct 2015, co-advisors I. Puaut and S. Derrien

PhD in progress: Viet Anh Nguyen, Université Rennes 1, Jan 2015, co-advisors D. Hardy and I. Puaut

10.2.3. Juries

The member of ALF have participated to many PhD defense juries in France and abroad.

11. Bibliography

Major publications by the team in recent years

[1] M. Cornero, R. Costa, R. Fernández Pascual, A. Ornstein, E. Rohou. *An Experimental Environment Validating the Suitability of CLI as an Effective Deployment Format for Embedded Systems*, in "Conference on HiPEAC", Göteborg, Sweden, P. Stenström, M. Dubois, M. Katevenis, R. Gupta, T. Ungerer (editors), Springer, January 2008, pp. 130–144

[2] R. Costa, E. Rohou. *Comparing the size of .NET applications with native code*, in "3rd Intl Conference on Hardware/software codesign and system synthesis", Jersey City, NJ, USA, P. Eles, A. Jantsch, R. A. Bergamaschi (editors), ACM, September 2005, pp. 99–104

[3] D. Hardy, I. Puaut. *WCET analysis of multi-level non-inclusive set-associative instruction caches*, in "Proc. of the 29th IEEE Real-Time Systems Symposium", Barcelona, Spain, December 2008

[4] 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

[5] P. Michaud, Y. Sazeides, A. Seznec, T. Constantinou, D. Fetis. *A study of thread migration in temperature-constrained multi-cores*, in "ACM Transactions on Architecture and Code Optimization", 2007, vol. 4, n^{o} 2, 9 p.

[6] P. Michaud, A. Seznec, S. Jourdan. *An Exploration of Instruction Fetch Requirement in Out-of-Order Superscalar Processors*, in "International Journal of Parallel Programming", 2001, vol. 29, n^{o} 1, pp. 35-58

[7] 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

[8] N. Prémillieu, A. Seznec. *Efficient Out-of-Order Execution of Guarded ISAs*, in "ACM Transactions on Architecture and Code Optimization (TACO) ", December 2014, 21 p. [*DOI :* 10.1145/2677037], https://hal.inria.fr/hal-01103230

[9] E. ROHOU, M. SMITH. *Dynamically managing processor temperature and power*, in "Second Workshop on Feedback-Directed Optimizations", 1999

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

Publications of the year

Doctoral Dissertations and Habilitation Theses

[11] H. LI. *Extraction and traceability of annotations for WCET estimation*, Université Rennes 1, October 2015, https://tel.archives-ouvertes.fr/tel-01232613

[12] B. NARASIMHA SWAMY. *Exploiting heterogeneous manycores on sequential code*, UNIVERSITE DE RENNES 1, March 2015, https://hal.inria.fr/tel-01126807

[13] S. N. NATARAJAN. *Modeling performance of serial and parallel sections of multi-threaded programs in manycore era*, Inria Rennes - Bretagne Atlantique and University of Rennes 1, France, June 2015, https://hal.inria.fr/tel-01170039

[14] A. PERAIS. *Increasing the Performance of Superscalar Processors through Value Prediction*, Rennes 1, September 2015, https://hal.inria.fr/tel-01235370

[15] E. ROHOU. *Infrastructures and Compilation Strategies for the Performance of Computing Systems*, Université de Rennes 1, November 2015, Habilitation à diriger des recherches, https://hal.inria.fr/tel-01237164

Articles in International Peer-Reviewed Journals

[16] S. COLLANGE, D. DEFOUR, S. GRAILLAT, R. IAKYMCHUK. *Numerical Reproducibility for the Parallel Reduction on Multi- and Many-Core Architectures*, in "Parallel Computing", September 2015, vol. 49, pp. 83-97 [*DOI :* 10.1016/J.PARCO.2015.09.001], http://hal-lirmm.ccsd.cnrs.fr/lirmm-01206348

[17] M.-K. LEE, P. MICHAUD, J. S. SIM, D. NYANG. *A simple proof of optimality for the MIN cache replacement policy*, in "Information Processing Letters", September 2015, 3 p. [*DOI :* 10.1016/J.IPL.2015.09.004], https://hal.inria.fr/hal-01199424

[18] 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

[19] A. PERAIS, A. SEZNEC. *EOLE: Toward a Practical Implementation of Value Prediction*, in "IEEE Micro", June 2015, vol. 35, n^o 3, pp. 114 - 124 [*DOI :* 10.1109/MM.2015.45], https://hal.inria.fr/hal-01193287

[20] A. SURESH, B. NARASIMHA SWAMY, E. ROHOU, A. SEZNEC. *Intercepting Functions for Memoization: A Case Study Using Transcendental Functions*, in "ACM Transactions on Architecture and Code Optimization (TACO) ", July 2015, vol. 12, n^o 2, 23 p. [*DOI :* 10.1145/2751559], https://hal.inria.fr/hal-01178085

International Conferences with Proceedings

[21] A. Bonenfant, F. Carrier, H. Cassé, P. Cuenot, D. Claraz, N. Halbwachs, H. Li, C. Maiza, M. De Michiel, V. Mussot, C. Parent-Vigouroux, I. Puaut, P. Raymond, E. Rohou, P. Sotin. *When the worst-case execution time estimation gains from the application semantics*, in "8th European Congress on Embedded Real-Time Software and Systems", Toulouse, France, January 2016, https://hal.inria.fr/hal-01235781

[22] S. Eyerman, P. Michaud, W. Rogiest. *Revisiting Symbiotic Job Scheduling*, in "IEEE International Symposium on Performance Analysis of Systems and Software", Philadelphia, United States, March 2015 [*DOI :* 10.1109/ISPASS.2015.7095791], https://hal.inria.fr/hal-01139807

[23] N. Hallou, E. Rohou, P. Clauss, A. Ketterlin. *Dynamic Re-Vectorization of Binary Code*, in "International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation - SAMOS XV", Agios Konstantinos, Greece, July 2015, https://hal.inria.fr/hal-01155207

[24] 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", Amsterdam, Netherlands, January 2015, https://hal.inria.fr/hal-01162998

[25] R. Iakymchuk, D. Defour, S. Collange, S. Graillat. *Reproducible Triangular Solvers for High-Performance Computing*, in "ITNG'2015: 12th International Conference on Information Technology - New Generations", Las Vegas, NV, United States, April 2015, pp. 353-358 [*DOI :* 10.1109/ITNG.2015.63], http://hal-lirmm.ccsd.cnrs.fr/lirmm-01206371

[26] H. Li, I. Puaut, E. Rohou. *Tracing Flow Information for Tighter WCET Estimation: Application to Vectorization*, in "21st IEEE International Conference on Embedded and Real-Time Computing Systems and Applications", Hong-Kong, China, August 2015, 10 p. , https://hal.inria.fr/hal-01177902

[27] *Best Paper*
P. Michaud. *A Best-Offset Prefetcher*, in "2nd Data Prefetching Championship", Portland, United States, June 2015, https://hal.inria.fr/hal-01165600.

[28] S. N. Natarajan, B. Narasimha Swamy, A. Seznec. *An Empirical High Level Performance Model For FutureMany-cores*, in "Proceedings of the 12th ACM International Conference on Computing Frontiers", Ischia, Italy, 2015 [*DOI :* 10.1145/2742854.2742867], https://hal.inria.fr/hal-01170038

[29] S. N. Natarajan, A. Seznec. *Sequential and Parallel Code Sections are Different: they may require different Processors*, in "PARMA-DITAM '15 - 6th Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures", Amsterdam, Netherlands, 6th Workshop on Parallel Programming and Run-Time Management Techniques for Many-core Architectures and 4th Workshop on Design Tools and Architectures for Multicore Embedded Computing Platforms, ACM, January 2015, pp. 13-18 [*DOI :* 10.1145/2701310.2701314], https://hal.inria.fr/hal-01170061

[30] M.-M. Papadopoulou, X. Tong, A. Seznec, A. Moshovos. *Prediction-based superpage-friendly TLB designs*, in "21st IEEE symposium on High Performance Computer Architecture", San Francisco, United States, 2015 [*DOI :* 10.1109/HPCA.2015.7056034], https://hal.inria.fr/hal-01193176

[31] A. Perais, A. Seznec, P. Michaud, A. Sembrant, E. Hagersten. *Cost-Effective Speculative Scheduling in High Performance Processors*, in "International Symposium on Computer Architecture", Portland,

United States, Proceedings of the International Symposium on Computer Architecture, ACM/IEEE, June 2015, vol. 42, pp. 247-259 [*DOI :* 10.1145/2749469.2749470], https://hal.inria.fr/hal-01193233

[32] A. PERAIS, A. SEZNEC. *BeBoP: A Cost Effective Predictor Infrastructure for Superscalar Value Prediction*, in "International Symposium on High Performance Computer Architecture", San Francisco, United States, IEEE, February 2015, vol. 21, pp. 13 - 25) [*DOI :* 10.1109/HPCA.2015.7056018], https://hal.inria.fr/hal-01193175

[33] E. ROHOU, D. GUYON. *Sequential Performance: Raising Awareness of the Gory Details*, in "International Conference on Computational Science", Reykjavik, Iceland, June 2015 [*DOI :* 10.1016/J.PROCS.2015.05.347], https://hal.inria.fr/hal-01162336

[34] 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

[35] 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, 11 p. , https://hal.inria.fr/hal-01225019

[36] 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

[37] A. SEZNEC. *Bank-interleaved cache or memory indexing does not require euclidean division*, in "11th Annual Workshop on Duplicating, Deconstructing and Debunking", Portland, United States, June 2015, https://hal.inria.fr/hal-01208356

[38] C. SILVANO, G. AGOSTA, A. BARTOLINI, A. R. BECCARI, L. BENINI, J. BISPO, R. CMAR, J. M. P. CARDOSO, C. CAVAZZONI, J. MARTINOVIČ, G. PALERMO, M. PALKOVIČ, P. PINTO, E. ROHOU, N. SANNA, K. SLANINOVÁ. *AutoTuning and Adaptivity appRoach for Energy efficient eXascale HPC systems: the ANTAREX Approach*, in "Design, Automation, and Test in Europe", Dresden, Germany, Design, Automation, and Test in Europe, March 2016, https://hal.inria.fr/hal-01235741

[39] C. SILVANO, G. AGOSTA, A. BARTOLINI, A. BECCARI, L. BENINI, J. M. P. CARDOSO, C. CAVAZZONI, J. MARTINOVIČ, G. PALERMO, M. PALKOVIČ, E. ROHOU, N. SANNA, K. SLANINOVA. *ANTAREX – AutoTuning and Adaptivity appRoach for Energy efficient eXascale HPC systems*, in "18th IEEE International Conference on Computational Science and Engineering", Porto, Portugal, October 2015, https://hal.inria.fr/hal-01235713

[40] M. SUZANA, J. ABELLA, D. HARDY, E. QUINONES, I. PUAUT, F. J. CAZORLA. *Speeding up Static Probabilistic Timing Analysis*, in "International Conference on Architecture of Computing Systems", Porto, Portugal, Springer Lecture Notes on Computer Science (LNCS) series, March 2015, https://hal.inria.fr/hal-01235544

Conferences without Proceedings

[41] V. A. NGUYEN, D. HARDY, I. PUAUT. *Scheduling of parallel applications on many-core architectures with caches: bridging the gap between WCET analysis and schedulability analysis*, in "9th Junior Researcher Workshop on Real-Time Computing (JRWRTC 2015)", Lille, France, November 2015, https://hal.inria.fr/hal-01236191

Research Reports

[42] S. KALATHINGAL, S. COLLANGE, B. NARASIMHA SWAMY, A. SEZNEC. *Transforming TLP into DLP with the Dynamic Inter-Thread Vectorization Architecture*, Inria Rennes Bretagne Atlantique, December 2015, n^{o} RR-8830, https://hal.inria.fr/hal-01244938

[43] A. SRIDHARAN, A. SEZNEC. *Discrete Cache Insertion Policies for Shared Last Level Cache Management on Large Multicores*, Inria-IRISA Rennes Bretagne Atlantique, équipe ALF, December 2015, n^{o} RR-8816, https://hal.inria.fr/hal-01236706

Other Publications

[44] S. COLLANGE, D. DEFOUR, S. GRAILLAT, R. IAKYMCHUK. *Numerical Reproducibility for the Parallel Reduction on Multi- and Many-Core Architectures*, September 2015, working paper or preprint, https://hal.archives-ouvertes.fr/hal-00949355

[45] R. IAKYMCHUK, S. COLLANGE, D. DEFOUR, S. GRAILLAT. *ExBLAS: Reproducible and Accurate BLAS Library*, July 2015, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01202396

[46] R. IAKYMCHUK, S. COLLANGE, D. DEFOUR, S. GRAILLAT. *Reproducibility and Accuracy for High-Performance Computing*, April 2015, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01140531

[47] R. IAKYMCHUK, D. DEFOUR, S. COLLANGE, S. GRAILLAT. *Reproducible and Accurate Matrix Multiplication for GPU Accelerators*, January 2015, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01102877

[48] R. IAKYMCHUK, D. DEFOUR, S. COLLANGE, S. GRAILLAT. *Reproducible Triangular Solvers for High-Performance Computing*, February 2015, working paper or preprint, https://hal.archives-ouvertes.fr/hal-01116588

[49] R. IAKYMCHUK, S. GRAILLAT, S. COLLANGE, D. DEFOUR. *ExBLAS: Reproducible and Accurate BLAS Library*, April 2015, RAIM'2015: 7ème Rencontre Arithmétique de l'Informatique Mathématique, Poster, https://hal.archives-ouvertes.fr/hal-01140280

References in notes

[50] G. M. AMDAHL. *Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities*, in "SJCC", 1967, pp. 483–485

[51] L. A. BELADY. *A study of replacement algorithms for a virtual-storage computer*, in "IBM Systems Journal", 1966, vol. 5, n^{o} 2, pp. 78-101

[52] D. BURGER, T. M. AUSTIN. *The simplescalar tool set, version 2.0*, 1997

[53] R. S. CHAPPELL, J. STARK, S. P. KIM, S. K. REINHARDT, Y. N. PATT. *Simultaneous subordinate microthreading (SSMT)*, in "ISCA '99: Proceedings of the 26th annual international symposium on Computer architecture", Washington, DC, USA, IEEE Computer Society, 1999, pp. 186–195, http://doi.acm.org/10.1145/300979.300995

[54] S. EYERMAN, L. EECKHOUT. *Probabilistic job symbiosis modeling for SMT processor scheduling*, in "Proceedings of the 15th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)", 2010

[55] C. FERDINAND, R. WILHELM. *Efficient and Precise Cache Behavior Prediction for Real-Time Systems*, in "Real-Time Systems", 1999, vol. 17, n^{o} 2-3, pp. 131–181, http://dx.doi.org/10.1023/A:1008186323068

[56] T. S. KARKHANIS, J. E. SMITH. *A First-Order Superscalar Processor Model*, in "Proceedings of the International Symposium on Computer Architecture", Los Alamitos, CA, USA, IEEE Computer Society, 2004, 338 p. , http://doi.ieeecomputersociety.org/10.1109/ISCA.2004.1310786

[57] B. LEE, J. COLLINS, H. WANG, D. BROOKS. *CPR : composable performance regression for scalable multiprocessor models*, in "Proceedings of the 41st International Symposium on Microarchitecture", 2008

[58] Y. LIANG, T. MITRA. *Cache modeling in probabilistic execution time analysis*, in "DAC '08: Proceedings of the 45th annual conference on Design automation", New York, NY, USA, ACM, 2008, pp. 319–324, http://doi.acm.org/10.1145/1391469.1391551

[59] T. LUNDQVIST, P. STENSTRÖM. *Timing Anomalies in Dynamically Scheduled Microprocessors*, in "RTSS '99: Proceedings of the 20th IEEE Real-Time Systems Symposium", Washington, DC, USA, IEEE Computer Society, 1999

[60] R. L. MATTSON, J. GECSEI, D. R. SLUTZ, I. L. TRAIGER. *Evaluation techniques for storage hierarchies*, in "IBM Systems Journal", 1970, vol. 9, n^{o} 2, pp. 78-117

[61] L. RAUCHWERGER, Y. ZHAN, J. TORRELLAS. *Hardware for Speculative Run-Time Parallelization in Distributed Shared-Memory Multiprocessors*, in "HPCA '98: Proceedings of the 4th International Symposium on High-Performance Computer Architecture", Washington, DC, USA, IEEE Computer Society, 1998, 162 p.

[62] K. SKADRON, M. STAN, W. HUANG, S. VELUSAMY. *Temperature-aware microarchitecture*, in "Proceedings of the International Symposium on Computer Architecture", 2003

[63] A. SNAVELY, D. M. TULLSEN. *Symbiotic jobscheduling for a simultaneous multithreading processor*, in "Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)", 2000

[64] J. G. STEFFAN, C. COLOHAN, A. ZHAI, T. C. MOWRY. *The STAMPede approach to thread-level speculation*, in "ACM Transactions on Computer Systems", 2005, vol. 23, n^{o} 3, pp. 253–300, http://doi.acm.org/10.1145/1082469.1082471

[65] V. SUHENDRA, T. MITRA. *Exploring locking & partitioning for predictable shared caches on multi-cores*, in "DAC '08: Proceedings of the 45th annual conference on Design automation", New York, NY, USA, ACM, 2008, pp. 300–303, http://doi.acm.org/10.1145/1391469.1391545

[66] D. M. TULLSEN, S. EGGERS, H. M. LEVY. *Simultaneous Multithreading: Maximizing On-Chip Parallelism*, in "Proceedings of the 22th Annual International Symposium on Computer Architecture", 1995

[67] J. YAN, W. ZHAN. *WCET Analysis for Multi-Core Processors with Shared L2 Instruction Caches*, in "Proceedings of Real-Time and Embedded Technology and Applications Symposium, 2008. RTAS '08", 2008, pp. 80-89