IN PARTNERSHIP WITH:
Institut Polytechnique de Bordeaux

Université de Bordeaux

Activity Report 2017

Project-Team STORM

STatic Optimizations, Runtime Methods

IN COLLABORATION WITH: Laboratoire Bordelais de Recherche en Informatique (LaBRI)

RESEARCH CENTER
Bordeaux - Sud-Ouest

THEME
Distributed and High Performance Computing

Table of contents

Project-Team STORM

Creation of the Team: 2015 January 01, updated into Project-Team: 2017 July 01

Keywords:

Computer Science and Digital Science:

A1.1.1. - Multicore, Manycore
A1.1.2. - Hardware accelerators (GPGPU, FPGA, etc.)
A1.1.3. - Memory models
A1.1.4. - High performance computing
A1.1.5. - Exascale
A2.2.1. - Static analysis
A2.2.2. - Memory models
A2.2.3. - Run-time systems
A2.2.4. - Parallel architectures
A2.2.5. - GPGPU, FPGA, etc.

Other Research Topics and Application Domains:

B2.2.1. - Cardiovascular and respiratory diseases
B3.2. - Climate and meteorology
B3.3.1. - Earth and subsoil
B3.4.1. - Natural risks
B4.2. - Nuclear Energy Production
B5.2.3. - Aviation
B5.2.4. - Aerospace
B6.2.2. - Radio technology
B6.2.3. - Satellite technology
B6.2.4. - Optic technology

1. Personnel

Research Scientists

Olivier Aumage [Inria, Researcher]
Emmanuelle Saillard [Inria, Researcher, from Oct 2017]

Faculty Members

Denis Barthou [Team leader, Institut National Polytechnique de Bordeaux, Professor, HDR]
Marie-Christine Counilh [Univ de Bordeaux, Associate Professor]
Raymond Namyst [Univ de Bordeaux, Professor, HDR]
Samuel Thibault [Univ de Bordeaux, Associate Professor]
Pierre-André Wacrenier [Univ de Bordeaux, Associate Professor]

Post-Doctoral Fellow

Luka Stanisic [Inria, until Mar 2017]

PhD Students

Hugo Brunie [CEA]
Adrien Cassagne [Inria, from Oct 2017]

Terry Cojean [Inria, until Aug 2017]
Christopher Haine [Inria, until Jun 2017]
Pierre Huchant [Univ de Bordeaux]
Suraj Kumar [Inria, until Apr 2017]
Arthur Loussert [CEA]
Raphaël Prat [CEA]
Marc Sergent [Univ de Bordeaux, until Jan 2017]
Leo Villeveygoux [Univ de Bordeaux, from Oct 2017]

Technical staff

Jérôme Clet-Ortega [Inria, until Feb 2017]
Nathalie Furmento [CNRS]
Yanis Khorsi [Inria, from Jul 2017]
Chiheb Sakka [Inria]
Corentin Salingue [Inria]

Interns

Lisa Arneau [Inria, from May 2017 until Aug 2017]
Arthur Chevalier [Inria, from Apr 2017 until Sep 2017]
Baptiste Coye [Inria, from Feb 2017 until Aug 2017]
Etienne Fabre [Inria, from Apr 2017 until Aug 2017]
Erwan Leria [Inria, from Jun 2017 until Aug 2017]
Ludovic San Nicolas [Inria, from Jun 2017 until Aug 2017]
Leo Villeveygoux [Inria, from Feb 2017 until Aug 2017]

Administrative Assistants

Sabrina Blondel-Duthil [Inria]
Sylvie Embolla [Inria]

Visiting Scientist

Sean Mondesir [University of Central Florida, from Apr 2017 until Nov 2017]

External Collaborators

Adrien Cassagne [Institut National Polytechnique de Bordeaux, until Sep 2017]
Terry Cojean [Univ de Bordeaux, from Sep 2017]

2. Overall Objectives

2.1. Overall Objectives

A successful approach to deal with the complexity of modern architectures is centered around the use of runtime systems, to manage tasks dynamically, these runtime systems being either generic or specific to an application. Similarly, on the compiler side, optimizations and analyses are more aggressive in iterative compilation frameworks, fit for library generations, or DSL, in particular for linear algebra methods. To go beyond this state of the art and alleviate the difficulties for programming these machines, we believe it is necessary to provide inputs with richer semantics to runtime and compiler alike, and in particular by combining both approaches.

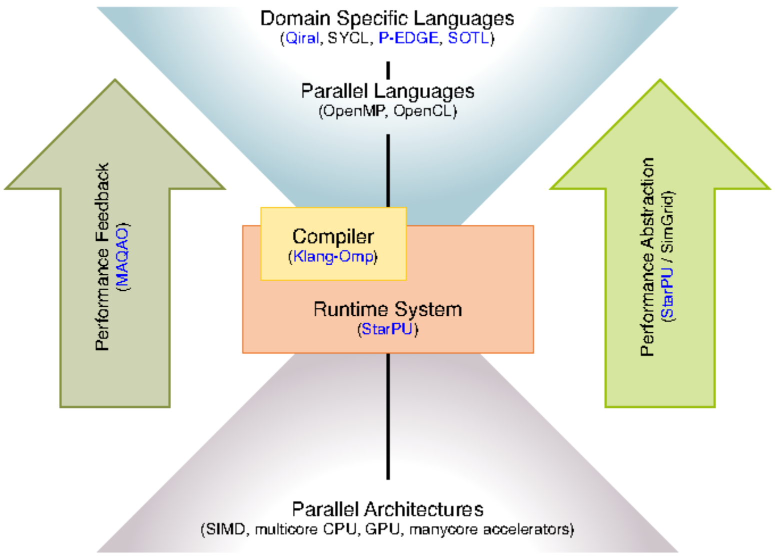

Figure 1. STORM Big Picture

This general objective is declined into two sub-objectives, the first concerning the expression of parallelism itself, the second the optimization and adaptation of this parallelism by compilers and runtimes.

- Expressing parallelism: As shown in the following figure, we propose to work on parallelism expression through Domain Specific Languages, able to capture the essence of the algorithms used through usual parallel languages such as OpenCL, OpenMP and through high performance libraries. The DSLs will be driven by applications, with the idea to capture at the algorithmic level the parallelism of the problem and perform dynamic data layout adaptation, parallel and algorithmic optimizations. The principle here is to capture a higher level of semantics, enabling users to express not only parallelism but also different algorithms.
- Optimizing and adapting parallelism: The goal here is to leverage the necessary adaptation to evolving hardware, by providing mechanisms allowing users to run the same code on different architectures. This implies to adapt parallelism, in particular the granularity of the work, to the architecture. This relies on the use of existing parallel libraries and their composition, and more generally the separation of concern between the description of tasks, that represent semantic units of work, and the tasks to be executed by the different processing units. Splitting or coarsening moldable tasks, generating code for these tasks and scheduling them is part of this work.

  Finally, the abstraction we advocate for requires to propose a feed back loop. This feed back has two objectives: To make users better understand their application and how to change the expression of parallelism if necessary, but also to propose an abstracted model for the machine. This allows to develop and formalize the compiling, scheduling techniques on a model, not too far from the real machine. Here, simulation techniques are a way to abstract the complexity of the architecture while preserving essential metrics.

3. Research Program

3.1. Parallel Computing and Architectures

Following the current trends of the evolution of HPC systems architectures, it is expected that future Exascale systems (i.e. Sustaining 10^{18} flops) will have millions of cores. Although the exact architectural details and trade-offs of such systems are still unclear, it is anticipated that an overall concurrency level of $O(10^9)$ threads/tasks will probably be required to feed all computing units while hiding memory latencies. It will obviously be a challenge for many applications to scale to that level, making the underlying system sound like “embarrassingly parallel hardware.”

From the programming point of view, it becomes a matter of being able to expose extreme parallelism within applications to feed the underlying computing units. However, this increase in the number of cores also comes with architectural constraints that actual hardware evolution prefigures: computing units will feature extra-wide SIMD and SIMT units that will require aggressive code vectorization or “SIMDization”, systems will become hybrid by mixing traditional CPUs and accelerators units, possibly on the same chip as the AMD APU solution, the amount of memory per computing unit is constantly decreasing, new levels of memory will appear, with explicit or implicit consistency management, etc. As a result, upcoming extreme-scale system will not only require unprecedented amount of parallelism to be efficiently exploited, but they will also require that applications generate adaptive parallelism capable to map tasks over heterogeneous computing units.

The current situation is already alarming, since European HPC end-users are forced to invest in a difficult and time-consuming process of tuning and optimizing their applications to reach most of current supercomputers’ performance. It will go even worse at horizon 2020 with the emergence of new parallel architectures (tightly integrated accelerators and cores, high vectorization capabilities, etc.) featuring unprecedented degree of parallelism that only too few experts will be able to exploit efficiently. As highlighted by the ETP4HPC initiative, existing programming models and tools won’t be able to cope with such a level of heterogeneity, complexity and number of computing units, which may prevent many new application opportunities and new science advances to emerge.

The same conclusion arises from a non-HPC perspective, for single node embedded parallel architectures, combining heterogeneous multicores, such as the ARM big.LITTLE processor and accelerators such as GPUs or DSPs. The need and difficulty to write programs able to run on various parallel heterogeneous architectures has led to initiatives such as HSA, focusing on making it easier to program heterogeneous computing devices. The growing complexity of hardware is a limiting factor to the emergence of new usages relying on new technology.

3.2. Scientific and Societal Stakes

In the HPC context, simulation is already considered as a third pillar of science with experiments and theory. Additional computing power means more scientific results, and the possibility to open new fields of simulation requiring more performance, such as multi-scale, multi-physics simulations. Many scientific domains able to take advantage of Exascale computers, these “Grand Challenges” cover large panels of science, from seismic, climate, molecular dynamics, theoretical and astrophysics physics... Besides, embedded applications are also able to take advantage of these performance increase. There is still an on-going trend where dedicated hardware is progressively replaced by off-the-shelf components, adding more adaptability and lowering the cost of devices. For instance, Error Correcting Codes in cell phones are still hardware chips, but with the forthcoming 5G protocol, new software and adaptative solutions relying on low power multicores are also explored. New usages are also appearing, relying on the fact that large computing capacities are becoming more affordable and widespread. This is the case for instance with Deep Neural Networks where the training phase can be done on supercomputers and then used in embedded mobile systems. The same consideration applies for big data problems, of internet of things, where small sensors provide large amount of data that need to be processed in short amount of time. Even though the computing capacities required for such applications are in general a different scale from HPC infrastructures, there is still a need in the future for high performance computing applications.

However, the outcome of new scientific results and the development of new usages for mobile, embedded systems will be hindered by the complexity and high level of expertise required to tap the performance offered by future parallel heterogeneous architectures.

3.3. Towards More Abstraction

As emphasized by initiatives such as the European Exascale Software Initiative (EESI), the European Technology Platform for High Performance Computing (ETP4HPC), or the International Exascale Software Initiative (IESP), the HPC community needs new programming APIs and languages for expressing heterogeneous massive parallelism in a way that provides an abstraction of the system architecture and promotes high performance and efficiency. The same conclusion holds for mobile, embedded applications that require performance on heterogeneous systems.

This crucial challenge given by the evolution of parallel architectures therefore comes from this need to make high performance accessible to the largest number of developpers, abstracting away architectural details providing some kind of performance portability. Disruptive uses of the new technology and groundbreaking new scientific results will not come from code optimization or task scheduling, but they require the design of new algorithms that require the technology to be tamed in order to reach unprecedented levels of performance.

Runtime systems and numerical libraries are part of the answer, since they may be seen as building blocks optimized by experts and used as-is by application developers. The first purpose of runtime systems is indeed to provide *abstraction*. Runtime systems offer a uniform programming interface for a specific subset of hardware (e.g., OpenGL or DirectX are well-established examples of runtime systems dedicated to hardware-accelerated graphics) or low-level software entities (e.g., POSIX-thread implementations). They are designed as thin user-level software layers that complement the basic, general purpose functions provided by the operating system calls. Applications then target these uniform programming interfaces in a portable manner. Low-level, hardware dependent details are hidden inside runtime systems. The adaptation of runtime systems is commonly handled through drivers. The abstraction provided by runtime systems thus enables portability.

Abstraction alone is however not enough to provide portability of performance, as it does nothing to leverage low-level-specific features to get increased performance. Consequently, the second role of runtime systems is to *optimize* abstract application requests by dynamically mapping them onto low-level requests and resources as efficiently as possible. This mapping process makes use of scheduling algorithms and heuristics to decide the best actions to take for a given metric and the application state at a given point in its execution time. This allows applications to readily benefit from available underlying low-level capabilities to their full extent without breaking their portability. Thus, optimization together with abstraction allows runtime systems to offer portability of performance. Numerical libraries provide sets of highly optimized kernels for a given field (dense or sparse linear algebra, FFT, etc.) either in an autonomous fashion or using an underlying runtime system.

Application domains cannot resort to libraries for all codes however, computation patterns such as stencils are a representative example of such difficulty. The compiler technology plays here a central role, in managing high level semantics, either through templates, domain specific languages or annotations. Compiler optimizations, and the same applies for runtime optimizations, are limited by the level of semantics they manage. Providing part of the algorithmic knowledge of an application, for instance knowing that it computes a 5-point stencil and then performs a dot product, would lead to more opportunities to adapt parallelism, memory structures, and is a way to leverage the evolving hardware.

Compilers and runtime play a crucial role in the future of high performance applications, by defining the input language for users, and optimizing/transforming it into high performance code. The objective of STORM is to propose better interactions between compiler and runtime and more semantics for both approaches. We recall in the following section the expertise of the team.

4. Application Domains

4.1. Application Domains

The application of our work concerns linear algebra, solvers and fast-multipole methods, in collaboration with other Inria teams and with industry. This allows a wide range of scientific and industrial applications possibly interested in the techniques we propose, in the domain of high performance computing but also in order to compute intensive embedded applications. In terms of direct application, the software developed in the team are used in applications in various fields, ranging from seismic, mechanic of fluids, molecular dynamics, high energy physics or material simulations. Similarly, the domains of image processing and signal processing can take advantage of the expertise and software of the team.

5. New Software and Platforms

5.1. Chameleon

KEYWORDS: Runtime system - Task-based algorithm - Dense linear algebra - HPC - Task scheduling
SCIENTIFIC DESCRIPTION: Chameleon is part of the MORSE (Matrices Over Runtime Systems @ Exascale) project. The overall objective is to develop robust linear algebra libraries relying on innovative runtime systems that can fully benefit from the potential of those future large-scale complex machines.

We expect advances in three directions based first on strong and closed interactions between the runtime and numerical linear algebra communities. This initial activity will then naturally expand to more focused but still joint research in both fields.

1. Fine interaction between linear algebra and runtime systems. On parallel machines, HPC applications need to take care of data movement and consistency, which can be either explicitly managed at the level of the application itself or delegated to a runtime system. We adopt the latter approach in order to better keep up with hardware trends whose complexity is growing exponentially. One major task in this project is to define a proper interface between HPC applications and runtime systems in order to maximize productivity and expressivity. As mentioned in the next section, a widely used approach consists in abstracting the application as a DAG that the runtime system is in charge of scheduling. Scheduling such a DAG over a set of heterogeneous processing units introduces a lot of new challenges, such as predicting accurately the execution time of each type of task over each kind of unit, minimizing data transfers between memory banks, performing data prefetching, etc. Expected advances: In a nutshell, a new runtime system API will be designed to allow applications to provide scheduling hints to the runtime system and to get real-time feedback about the consequences of scheduling decisions.

2. Runtime systems. A runtime environment is an intermediate layer between the system and the application. It provides low-level functionality not provided by the system (such as scheduling or management of the heterogeneity) and high-level features (such as performance portability). In the framework of this proposal, we will work on the scalability of runtime environment. To achieve scalability it is required to avoid all centralization. Here, the main problem is the scheduling of the tasks. In many task-based runtime environments the scheduler is centralized and becomes a bottleneck as soon as too many cores are involved. It is therefore required to distribute the scheduling decision or to compute a data distribution that impose the mapping of task using, for instance the so-called “owner-compute” rule. Expected advances: We will design runtime systems that enable an efficient and scalable use of thousands of distributed multicore nodes enhanced with accelerators.

3. Linear algebra. Because of its central position in HPC and of the well understood structure of its algorithms, dense linear algebra has often pioneered new challenges that HPC had to face. Again, dense linear algebra has been in the vanguard of the new era of petascale computing with the design of new algorithms that can efficiently run on a multicore node with GPU accelerators. These algorithms are called “communication-avoiding” since they have been redesigned to limit the amount of communication between processing units (and between the different levels of memory hierarchy). They are expressed through Direct Acyclic Graphs (DAG) of fine-grained tasks that are dynamically scheduled. Expected advances: First, we plan to investigate the impact of these principles in the case of sparse applications (whose algorithms are slightly more complicated but often rely on dense kernels). Furthermore, both in the dense and sparse cases, the scalability on thousands of nodes is still limited, new numerical approaches need to be found. We will specifically design sparse hybrid direct/iterative methods that represent a promising approach.

Overall end point. The overall goal of the MORSE associate team is to enable advanced numerical algorithms to be executed on a scalable unified runtime system for exploiting the full potential of future exascale machines. FUNCTIONAL DESCRIPTION: Chameleon is a dense linear algebra software relying on sequential task-based algorithms where sub-tasks of the overall algorithms are submitted to a Runtime system. A Runtime system such as StarPU is able to manage automatically data transfers between not shared memory area (CPUs-GPUs, distributed nodes). This kind of implementation paradigm allows to design high performing linear algebra algorithms on very different type of architecture: laptop, many-core nodes, CPUs-GPUs, multiple nodes. For example, Chameleon is able to perform a Cholesky factorization (double-precision) at 80 TFlop/s on a dense matrix of order 400 000 (e.i. 4 min).

RELEASE FUNCTIONAL DESCRIPTION: Chameleon includes the following features:

- BLAS 3, LAPACK one-sided and LAPACK norms tile algorithms - Support QUARK and StarPU runtime systems - Exploitation of homogeneous and heterogeneous platforms through the use of BLAS/LAPACK CPU kernels and cuBLAS/MAGMA CUDA kernels - Exploitation of clusters of interconnected nodes with distributed memory (using OpenMPI)

- Participants: Cédric Castagnede, Samuel Thibault, Emmanuel Agullo, Florent Pruvost and Mathieu Faverge

- Partners: Innovative Computing Laboratory (ICL) - King Abdullha University of Science and Technology - University of Colorado Denver
- Contact: Emmanuel Agullo
- URL: https://project.inria.fr/chameleon/

5.2. hwloc

Hardware Locality
KEYWORDS: NUMA - Multicore - GPU - Affinities - Open MPI - Topology - HPC - Locality
FUNCTIONAL DESCRIPTION: Hardware Locality (hwloc) is a library and set of tools aiming at discovering and exposing the topology of machines, including processors, cores, threads, shared caches, NUMA memory nodes and I/O devices. It builds a widely-portable abstraction of these resources and exposes it to applications so as to help them adapt their behavior to the hardware characteristics. They may consult the hierarchy of resources, their attributes, and bind task or memory on them.

hwloc targets many types of high-performance computing applications, from thread scheduling to placement of MPI processes. Most existing MPI implementations, several resource managers and task schedulers, and multiple other parallel libraries already use hwloc.

- Participants: Brice Goglin and Samuel Thibault
- Partners: Open MPI consortium - Intel - AMD
- Contact: Brice Goglin
- Publications: hwloc: a Generic Framework for Managing Hardware Affinities in HPC Applications - Managing the Topology of Heterogeneous Cluster Nodes with Hardware Locality (hwloc) - A Topology-Aware Performance Monitoring Tool for Shared Resource Management in Multicore Systems - Exposing the Locality of Heterogeneous Memory Architectures to HPC Applications - Towards the Structural Modeling of the Topology of next-generation heterogeneous cluster Nodes with hwloc - On the Overhead of Topology Discovery for Locality-aware Scheduling in HPC
- URL: http://www.open-mpi.org/projects/hwloc/

5.3. KaStORS

The KaStORS OpenMP Benchmark Suite
KEYWORDS: OpenMP - Task scheduling - Task-based algorithm - HPC - Benchmarking - Data parallelism
FUNCTIONAL DESCRIPTION: The KaStORS benchmarks suite has been designed to evaluate implementations of the OpenMP dependent task paradigm, introduced as part of the OpenMP 4.0 specification.

- Participants: François Broquedis, Nathalie Furmento, Olivier Aumage, Philippe Virouleau, Pierrick Brunet, Samuel Thibault and Thierry Gautier
- Contact: Thierry Gautier
- URL: http://kastors.gforge.inria.fr/#!index.md

5.4. KStar

The KStar OpenMP Compiler
KEYWORDS: Source-to-source compiler - OpenMP - Task scheduling - Compilers - Data parallelism
FUNCTIONAL DESCRIPTION: The KStar software is a source-to-source OpenMP compiler for languages C and C++. The KStar compiler translates OpenMP directives and constructs into API calls from the StarPU runtime system or the XKaapi runtime system. The KStar compiler is virtually fully compliant with OpenMP 3.0 constructs. The KStar compiler supports OpenMP 4.0 dependent tasks and accelerated targets.

- Participants: Nathalie Furmento, Olivier Aumage, Samuel Pitoiset and Samuel Thibault
- Contact: Olivier Aumage
- URL: http://kstar.gforge.inria.fr/#!index.md

5.5. MAQAO

SCIENTIFIC DESCRIPTION: MAQAO relies on binary codes for Intel x86 and ARM architectures. For x86 architecture, it can insert probes for instrumention directly inside the binary. There is no need to recompile. The static/dynamic approach of MAQAO analysis is the main originality of the tool, combining performance model with values collected through instrumentation.

MAQAO has a static performance model for x86 and ARM architectures. This model analyzes performance of the codes on the architectures and provides some feed-back hints on how to improve these codes, in particular for vector instructions.

The dynamic collection of data in MAQAO enables the analysis of thread interactions, such as false sharing, amount of data reuse, runtime scheduling policy, ...
FUNCTIONAL DESCRIPTION: MAQAO is a performance tuning tool for OpenMP parallel applications. It relies on the static analysis of binary codes and the collection of dynamic information (such as memory traces). It provides hints to the user about performance bottlenecks and possible workarounds.

- Participants: Christopher Haine, Denis Barthou, James Tombi A Mba and Olivier Aumage
- Contact: Denis Barthou

5.6. StarPU

The StarPU Runtime System
KEYWORDS: Multicore - GPU - Scheduling - HPC - Performance
SCIENTIFIC DESCRIPTION: Traditional processors have reached architectural limits which heterogeneous multicore designs and hardware specialization (eg. coprocessors, accelerators, ...) intend to address. However, exploiting such machines introduces numerous challenging issues at all levels, ranging from programming models and compilers to the design of scalable hardware solutions. The design of efficient runtime systems for these architectures is a critical issue. StarPU typically makes it much easier for high performance libraries or compiler environments to exploit heterogeneous multicore machines possibly equipped with GPGPUs or Cell processors: rather than handling low-level issues, programmers may concentrate on algorithmic concerns.Portability is obtained by the means of a unified abstraction of the machine. StarPU offers a unified offloadable task abstraction named "codelet". Rather than rewriting the entire code, programmers can encapsulate existing functions within codelets. In case a codelet may run on heterogeneous architectures, it is possible to specify one function for each architectures (eg. one function for CUDA and one function for CPUs). StarPU takes care to schedule and execute those codelets as efficiently as possible over the entire machine. In order to relieve programmers from the burden of explicit data transfers, a high-level data management library enforces memory coherency over the machine: before a codelet starts (eg. on an accelerator), all its data are transparently made available on the compute resource.Given its expressive interface and portable scheduling policies, StarPU obtains portable performances by efficiently (and easily) using all computing resources at the same time. StarPU also takes advantage of the heterogeneous nature of a machine, for instance by using scheduling strategies based on auto-tuned performance models.

StarPU is a task programming library for hybrid architectures

The application provides algorithms and constraints: - CPU/GPU implementations of tasks - A graph of tasks, using either the StarPU's high level GCC plugin pragmas or StarPU's rich C API

StarPU handles run-time concerns - Task dependencies - Optimized heterogeneous scheduling - Optimized data transfers and replication between main memory and discrete memories - Optimized cluster communications

Rather than handling low-level scheduling and optimizing issues, programmers can concentrate on algorithmic concerns!

FUNCTIONAL DESCRIPTION: StarPU is a runtime system that offers support for heterogeneous multicore machines. While many efforts are devoted to design efficient computation kernels for those architectures (e.g. to implement BLAS kernels on GPUs), StarPU not only takes care of offloading such kernels (and implementing data coherency across the machine), but it also makes sure the kernels are executed as efficiently as possible.

- Participants: Corentin Salingue, Andra Hugo, Benoît Lize, Cédric Augonnet, Cyril Roelandt, François Tessier, Jérôme Clet-Ortega, Ludovic Courtes, Ludovic Stordeur, Marc Sergent, Mehdi Juhoor, Nathalie Furmento, Nicolas Collin, Olivier Aumage, Pierre-André Wacrenier, Raymond Namyst, Samuel Thibault, Simon Archipoff and Xavier Lacoste
- Contact: Olivier Aumage
- URL: http://starpu.gforge.inria.fr/

5.7. PARCOACH

PARallel Control flow Anomaly CHecker
KEYWORDS: High-Performance Computing - Program verification - Debug - MPI - OpenMP - Compilation
SCIENTIFIC DESCRIPTION: PARCOACH verifies programs in two steps. First, it statically verifies applications with a data- and control-flow analysis and outlines execution paths leading to potential deadlocks. The code is then instrumented, displaying an error and synchronously interrupting all processes if the actual scheduling leads to a deadlock situation.
FUNCTIONAL DESCRIPTION: Supercomputing plays an important role in several innovative fields, speeding up prototyping or validating scientific theories. However, supercomputers are evolving rapidly with now millions of processing units, posing the questions of their programmability. Despite the emergence of more widespread and functional parallel programming models, developing correct and effective parallel applications still remains a complex task. As current scientific applications mainly rely on the Message Passing Interface (MPI) parallel programming model, new hardwares designed for Exascale with higher node-level parallelism clearly advocate for an MPI+X solutions with X a thread-based model such as OpenMP. But integrating two different programming models inside the same application can be error-prone leading to complex bugs - mostly detected unfortunately at runtime. PARallel COntrol flow Anomaly CHecker aims at helping developers in their debugging phase.

- Participants: Emmanuelle Saillard, Denis Barthou and Pierre Huchant
- Partner: CEA
- Contact: Emmanuelle Saillard
- URL: https://esaillar.github.io/PARCOACH/

5.8. AFF3CT

A Fast Forward Error Correction Tool
KEYWORDS: High-Performance Computing - Signal processing - Error Correction Code
FUNCTIONAL DESCRIPTION: AFF3CT proposes high performance Error Correction algorithms for Polar, Turbo, LDPC, RSC (Recursive Systematic Convolutional), Repetition and RA (Repeat and Accumulate) codes. These signal processing codes can be parameterized in order to optimize some given metrics, such as Bit Error Rate, Bandwidth, Latency, ...using simulation. For the designers of such signal processing chain, AFF3CT proposes also high performance building blocks so to develop new algorithms. AFF3CT compiles with many compilers and runs on Windows, Mac OS X, Linux environments and has been optimized for x86 (SSE, AVX instruction sets) and ARM architectures (NEON instruction set).

- Authors: Adrien Cassagne, Bertrand Le Gal, Camille Leroux, Denis Barthou and Olivier Aumage
- Partner: IMS
- Contact: Adrien Cassagne
- URL: https://aff3ct.github.io/

5.9. MORSE

KEYWORDS: High performance computing - Matrix calculation - Fast multipole method - Runtime system
FUNCTIONAL DESCRIPTION: MORSE (Matrices Over Runtime Systems @ Exascale) is a scientific project, its objectives are to solve matrix problems on complex architectures, using runtime systems. More specifically, the goal is to write codes that reach a high level of performance for all architectures. The algorithms are written independently of the architecture, and the runtime system dispatches the different computational parts to the different computing units. This methodology has been validated on three classes of problems: dense linear algebra, sparse and dense, and fast multipole methods. The corresponding codes have been incorporated into several softwares, MAGMA, Pastix and ScalFMM.

- Contact: Emmanuel Agullo
- URL: http://icl.cs.utk.edu/morse/

6. New Results

6.1. Distributed Sequential Task Flow with StarPU

The emergence of accelerators as standard computing resources on supercomputers and the subsequent architectural complexity increase revived the need for high-level parallel programming paradigms. Sequential task-based programming model has been shown to efficiently meet this challenge on a single multicore node possibly enhanced with accelerators, which motivated its support in the OpenMP 4.0 standard. We showed [5] that this paradigm can also be employed to achieve high performance on modern supercomputers composed of multiple such nodes, with extremely limited changes in the user code. To prove this claim, we have extended the StarPU runtime system with an advanced inter-node data management layer that supports this model by posting communications automatically. We illustrates our discussion with the task-based tile Cholesky algorithm that we implemented on top of this new runtime system layer. We showed that it enables very high productivity while achieving a performance competitive with both the pure Message Passing Interface (MPI)-based ScaLAPACK Cholesky reference implementation and the DPLASMA Cholesky code, which implements another (non-sequential) task-based programming paradigm.

6.2. Distributed StarPU on top of a High Performance Communication Library

A new implementation of the StarPU's distributed engine is being currently developed on top of the NewMadeleine library. The first version of this engine had been written directly on top of MPI. The performance were not as good as expected when dealing with applications exchanging huge number of messages, and we had to implement within StarPU mechanisms to control the memory subscription [14].

NewMadeleine is a high performance communication library for clusters developed in the Tadaam team. It applies optimization strategy on data flows through dynamic packet scheduling, and is usable on various high performance networks. The new implementation of the StarPU's distributed engine no longer has to deal with communication-related issues, and provides a better reactivity as the communications progress is dealt with by NewMadeleine itself. First experiments with the Chameleon solver show promising results.

6.3. Bridging the Gap between a Standard Parallel Language and a Task-based Runtime System

With the advent of complex modern architectures, the low-level paradigms long considered sufficient to build High Performance Computing (HPC) numerical codes have met their limits. Achieving efficiency, ensuring portability, while preserving programming tractability on such hardware prompted the HPC community to design new, higher level paradigms while relying on runtime systems to maintain performance. However, the common weakness of these projects is to deeply tie applications to specific expert-only runtime system APIs. The OpenMP specification, which aims at providing common parallel programming means for shared-memory platforms, appears as a good candidate to address this issue thanks to the latest task-based constructs introduced in its revision 4.0. We assessed [4] the effectiveness and limits of this support for designing a high-performance numerical library, ScalFMM, implementing the fast multipole method (FMM) that we have deeply re-designed with respect to the most advanced features provided by OpenMP 4. We showed that OpenMP 4 allows for significant performance improvements over previous OpenMP revisions on recent multicore processors and that extensions to the 4.0 standard allow for strongly improving the performance, bridging the gap with the very high performance that was so far reserved to expert-only runtime system APIs. Our proposal for an OpenMP extension to let the programmer express the property of commutativity between multiple tasks has been presented by Inria and successfully voted-on and integrated as the notion of mutually exclusive input/output sets (mutexinoutset keyword) in OpenMP ARB's Technical Report 6: OpenMP Version 5.0 Preview 2, the last pre-version of the upcoming OpenMP 5.0 specification.

6.4. Combining a Component Model and a Task Parallelism Model

We demonstrated the feasibility of efficiently combining both a software component model and a task-based model [6]. Task based models are known to enable efficient executions on recent HPC computing nodes while component models ease the separation of concerns of application and thus improve their modularity and adaptability.

This paper describes a prototype version of the COMET programming model combining concepts of task-based and component models, and a preliminary version of the COMET runtime built on top of StarPU and L2C. Evaluations of the approach have been conducted on a real-world use-case analysis of a sub-part of the production application GYSELA.

Results show that the approach is feasible and that it enables easy composition of independent software codes without introducing overheads. Performance results are equivalent to those obtained with a plain OpenMP based implementation.

6.5. Tackling the granularity problem

One of the main issues encountered when trying to exploit both CPUs and accelerators is that these devices have very different characteristics and requirements. Indeed, GPUs typically exhibit better performance when executing kernels applied to large data sets while regular CPU cores reach their peak performance with fine grain kernels working on a reduced memory footprint. To work around this granularity problem, task-based applications running on such heterogeneous platforms typically adopt a medium granularity, chosen as a trade-off between coarse-grain and fine-grain kernels. To tackle this granularity problem, we investigated different complementary technics. The first two technics are based on StarPU, performing both load-balancing and scheduling, the third one splits automatically kernels at compile-time and then performs load-balancing.

- The first technic is based on resource aggregation : we aggregate CPU cores to execute coarse grain tasks in a parallel manner. We have showed that this technic for a dense Cholesky factorization kernel outperforms state of the art implementations on a platform equipped with 24 CPU cores and 4 GPU devices (reaching a peak performance of 4.8 TFlop/s) and on the Intel KNL processor (reaching a peak performance 1.58 TFlop/s).
- The second technic splits dynamically coarse grain tasks when they are assigned to CPU cores. Tasks can be replaced by a subgraph of tasks of finer granularity, allowing for a finer handling of dependencies and a better pipelining of kernels. This mechanism allowing to deal with hierarchical task graphs has been designed within StarPU. Moreover, it allows to parallelize the task submission flow while preserving the simplicity of the sequential task flow submission paradigm. First experimental results for dense Cholesky factorization kernel show good performance improvements with respect to the native StarPU's implementation.

- The third technic extends our previous work that provides an automatic compiler and runtime technique to execute single OpenCL kernels on heterogeneous multi-device architectures. Our technique splits computation and data automatically across the multiple computing devices. OpenCL applications that consist in a chain of data-dependent kernels in an iterative computation are now considered.

  The technique proposed is completely transparent to the user, and does not require off-line training or a performance model. It manages sometimes conflicting needs between load balancing each kernel in the chain and minimizing data transfer between consecutive kernels, taking data locality into account. Load-balancing issues, resulting from hardware heterogeneity, load imbalance within a kernel itself, and load variations between repeated executions are also managed.

  Experiments on some benchmarks show the interest of our approach and we are currently implementing it in an OpenCL N-body computation with short-range interactions.

6.6. Interfacing MAQAO and BOAST Frameworks for Kernel Autotuning on ARM Platforms

In Project MontBlanc 2's deliverable D5.11 [13] we presented the integration of STORM's MAQAO software (a binary-level code analysis framework) with BOAST (an automatic performance tuning framework for meta-programming and optimizing computing kernels) developed at LIG's NANOSIM. From source meta-kernels written in the RUBY language, BOAST generates multiple versions, in various target languages, optionally applying optimization transformations and strategies, and exploring the space of compiler flags combinations, to discover the most effective kernel tuning parameters. MAQAO offers a scriptable framework to disassemble kernel binaries, explore binary instruction flows, register-level data dependencies, program control structures, to patch, re-assemble and instrument kernel binaries for tracing data access patterns, and to process them from custom analyzers written in the LUA language. This integration work built on the complementarity of these two environments by enabling MAQAO to process binary kernels generated by BOAST, and lead developers in a guided tuning cycle.

6.7. Using heterogeneous memories

Heterogeneous memories, such as the MCDRAM in the Xeon Phi architecture, with different latency and bandwidth characteristics, complexify the way the users allocate and use memory. In 2017, we have designed, in collaboration with CEA, an automatic tool to characterize the bandwidth needs of an application, in particular finding the functions and the arrays in these functions that would benefit the most of a high bandwidth. This tool is a plugin of gcc, and has been applied successfully to large CORAL benchmarks (Lulesh, MiniFE, AMG2013, Mcb and Snap). This characterization is essential in the common case where all data cannot fit into the MCDRAM but a more selective use of the MCDRAM is needed. The transformation of the memory allocations is automatic, based on these metrics. The development of better metrics, allowing to choose the most appropriate array is on going work.

6.8. Rewriting System for Profile-Guided Data Layout Transformations on Binaries

Careful data layout design is crucial for achieving high performance. However exploring data layouts is time-consuming and error-prone, and assessing the impact of a layout transformation on performance is difficult without performing it. We proposed [7] a method and implemented a prototype to guide application programmers through data layout restructuring for improving kernel performance and SIMDizability, by providing a comprehensive multidimensional description of the initial layout, built from trace analysis, and then by giving a performance evaluation of the transformations tested and an expression of each transformed layout. The programmer can limit the exploration to layouts matching some patterns. We apply this method to two multithreaded applications. The performance prediction of multiple transformations matches within 5% the performance of hand-transformed layout code.

6.9. Correctness of HPC Applications

The current supercomputer hardware trends lead to more complex HPC applications (heterogeneity in hardware and combinations of parallel programming models) that pose programmability challenges. Furthermore, progress to exascale stresses the requirement for convenient and scalable debugging methods to help developers fully exploit the future machines. Despite advances in the domain, this still remains a manual complex task. We aim to develop tools and methods to aid developers with problems of correctness in HPC applications for exascale systems. There are several requirements for such tools: 1) precision - report and handle only real problems, areas of interest; 2) scalability in LoCs and execution time; 3) heterogeneity - ability to handle multiple languages, runtime and execution models; and 4) soundness - ability to prove code properties. In order to improve developer productivity, we aim to develop a combination of static and dynamic analyses. Static analysis techniques will enable soundness and scalability in execution time. Dynamic analysis techniques will enable precision, scalability in LoCs and heterogeneity for hybrid parallelism.

The achieved results this year allow to perform an interprocedural static data- and control-flow analysis: its improves precision, by only detecting possible correctness issues related to MPI rank dependent variables. It improves scalability also by reducing the amount of dead-lock avoiding code added. This new method has been applied to CUDA, MPI, OpenMP and UPC parallel codes to detect collective deadlocks.

6.10. AMR-Based Dynamic Load Balancing for Molecular Dynamics Simulations

Modern parallel architectures require applications to generate enough parallelism to feed many cores, which require in turn regular data-parallel instructions to exploit large vector units. We revisit the extensively-studied Classical Molecular Dynamics N-body problem in the light of these hardware constraints. A new data layout is proposed with efficient force computation methods focusing on adaptive mesh refinement techniques, multi-threading, vectorization-friendly, using low memory footprint. Our design is guided by the need for load balancing and adaptivity raised by highly dynamic particle sets, as typically observed in simulations of strong shocks resulting in material micro-jetting. We analyze performance results on several simulation scenarios, over clusters equipped with Intel Xeon Phi knl processors. Performance obtained with our implementation using OpenMP is close to state-of-the-art implementations (LAMMPS) using MPI on steady particles simulations, and outperform them by 1.2 on micro-jetting simulations on Intel Xeon Phi (KNL).

6.11. Resource-Management Study in HPC Runtime-Stacking Context

With the advent of multicore and manycore processors as building blocks of HPC supercomputers, many applications shift from relying solely on a distributed programming model (e.g., MPI) to mixing distributed and shared-memory models (e.g., MPI+OpenMP), to better exploit shared-memory communications and reduce the overall memory footprint. One side effect of this programming approach is runtime stacking: mixing multiple models involve various runtime libraries to be alive at the same time and to share the underlying computing resources. This paper explores different configurations where this stacking may appear and introduces algorithms to detect the misuse of compute resources when running a hybrid parallel application. We have implemented our algorithms inside a dynamic tool that monitors applications and outputs resource usage to the user. We validated this tool on applications from CORAL benchmarks. This leads to relevant information which can be used to improve runtime placement, and to an average overhead lower than 1% of total execution time.

7. Bilateral Contracts and Grants with Industry

7.1. Bilateral Contracts with Industry

- HiBOX project, with Airbus and IMACS (2013-2017).

8. Partnerships and Cooperations

8.1. National Initiatives

8.1.1. PIA

ELCI The ELCI project (Software Environment for HPC) aims to develop a new generation of software stack for supercomputers, numerical solvers, runtime and programming development environments for HPC simulation. The ELCI project also aims to validate this software stack by showing its capacity to offer improved scalability, resilience, security, modularity and abstraction on real applications. The coordinator is Bull, and the different partners are CEA, Inria, SAFRAN, CERFACS, CNRS CORIA, CENAERO, ONERA, UVSQ, Kitware and AlgoTech.

8.1.2. ANR

ANR SOLHAR (http://solhar.gforge.inria.fr/doku.php?id=start).

ANR MONU 2013 Program, 2013 - 2017 (36 months extended)

Identification: ANR-13-MONU-0007

Coordinator: Inria Bordeaux/LaBRI

Other partners: CNRS-IRIT, Inria-LIP Lyon, CEA/CESTA, EADS-IW

Abstract: This project aims at studying and designing algorithms and parallel programming models for implementing direct methods for the solution of sparse linear systems on emerging computers equipped with accelerators. The ultimate aim of this project is to achieve the implementation of a software package providing a solver based on direct methods for sparse linear systems of equations. Several attempts have been made to accomplish the porting of these methods on such architectures; the proposed approaches are mostly based on a simple offloading of some computational tasks (the coarsest grained ones) to the accelerators and rely on fine hand-tuning of the code and accurate performance modeling to achieve efficiency. This project proposes an innovative approach which relies on the efficiency and portability of runtime systems, such as the StarPU tool developed in the runtime team (Bordeaux). Although the SOLHAR project will focus on heterogeneous computers equipped with GPUs due to their wide availability and affordable cost, the research accomplished on algorithms, methods and programming models will be readily applicable to other accelerator devices such as ClearSpeed boards or Cell processors.

ANR Songs Simulation of next generation systems (http://infra-songs.gforge.inria.fr/).

ANR INFRA 2011, 01/2012 - 12/2015 (48 months)

Identification: ANR-11INFR01306

Coordinator: Martin Quinson (Inria Nancy)

Other partners: Inria Nancy, Inria Rhône-Alpes, IN2P3, LSIIT, Inria Rennes, I3S.

Abstract: The goal of the SONGS project is to extend the applicability of the SimGrid simulation framework from Grids and Peer-to-Peer systems to Clouds and High Performance Computation systems. Each type of large-scale computing system will be addressed through a set of use cases and lead by researchers recognized as experts in this area.

8.1.3. ADT - Inria Technological Development Actions

ADT SwLoc (http://swloc.gforge.inria.fr/)

Participants: Raymond Namyst, Pierre-André Wacrenier, Andra Hugo, Brice Goglin, Corentin Salingue.

Inria ADT Campaign 2017, 10/2017 - 9/2019 (24 months)

Coordinator: Raymond Namyst

Abstract: The Inria action ADT SwLoc has the aim to develop a new library allowing dynamic flexible partitioning of computing resources in order to execute parallel regions.

8.1.4. IPL - Inria Project Lab

C2S@Exa - Computer and Computational Sciences at Exascale **Participant:** Olivier Aumage.

Inria IPL 2013 - 2017 (48 months)

Coordinator: Stéphane Lantéri (team Nachos, Inria Sophia)

Since January 2013, the team is participating to the C2S@Exa http://www-sop.inria.fr/c2s_at_exa Inria Project Lab (IPL). This national initiative aims at the development of numerical modeling methodologies that fully exploit the processing capabilities of modern massively parallel architectures in the context of a number of selected applications related to important scientific and technological challenges for the quality and the security of life in our society. This collaborative effort involves computer scientists that are experts of programming models, environments and tools for harnessing massively parallel systems, algorithmists that propose algorithms and contribute to generic libraries and core solvers in order to take benefit from all the parallelism levels with the main goal of optimal scaling on very large numbers of computing entities and, numerical mathematicians that are studying numerical schemes and scalable solvers for systems of partial differential equations in view of the simulation of very large-scale problems.

HAC-SPECIS - High-performance Application and Computers, Studying PErformance and Correctness In Simulation **Participants:** Samuel Thibault, Luka Stanisic, Emmanuelle Saillard.

Inria IPL 2016 - 2020 (48 months)

Coordinator: Arnaud Legrand (team Polaris, Inria Rhône Alpes)

Since June 2016, the team is participating to the HAC-SPECIS http://hacspecis.gforge.inria.fr/ Inria Project Lab (IPL). This national initiative aims at answering methodological needs of HPC application and runtime developers and allowing to study real HPC systems both from the correctness and performance point of view. To this end, it gathers experts from the HPC, formal verification and performance evaluation community.

8.2. European Initiatives

8.2.1. FP7 & H2020 Projects

8.2.1.1. INTERTWINE

Title: Programming Model INTERoperability ToWards Exascale

Programm: H2020

Duration: October 2015 - October 2018

Coordinator: EPCC

Partners:

Barcelona Supercomputing Center - Centro Nacional de Supercomputacion (Spain)

Deutsches Zentrum für Luft - und Raumfahrt Ev (Germany)

Fraunhofer Gesellschaft Zur Forderung Der Angewandten Forschung Ev (Germany)

Institut National de Recherche en Informatique et en Automatique (France)

Kungliga Tekniska Hoegskolan (Sweden)

T-Systems Solutions for Research (Germany)

The University of Edinburgh (United Kingdom)

Universitat Jaume I de Castellon (Spain)

The University of Manchester (United Kingdom)

Inria contact: Olivier Aumage

This project addresses the problem of programming model design and implementation for the Exascale. The first Exascale computers will be very highly parallel systems, consisting of a hierarchy of architectural levels. To program such systems effectively and portably, programming APIs with efficient and robust implementations must be ready in the appropriate timescale. A single, "silver bullet" API which addresses all the architectural levels does not exist and seems very unlikely to emerge soon enough. We must therefore expect that using combinations of different APIs at different system levels will be the only practical solution in the short to medium term. Although there remains room for improvement in individual programming models and their implementations, the main challenges lie in interoperability between APIs. It is this interoperability, both at the specification level and at the implementation level, which this project seeks to address and to further the state of the art. INTERTWinE brings together the principal European organisations driving the evolution of programming models and their implementations. The project will focus on seven key programming APIs: MPI, GASPI, OpenMP, OmpSs, StarPU, QUARK and PaRSEC, each of which has a project partner with extensive experience in API design and implementation. Interoperability requirements, and evaluation of implementations will be driven by a set of kernels and applications, each of which has a project partner with a major role in their development. The project will implement a co- design cycle, by feeding back advances in API design and implementation into the applications and kernels, thereby driving new requirements and hence further advances.

8.2.1.2. Mont-Blanc 2

Title: Mont-Blanc

Programm: FP7

Duration: Sep. 2013 - Mar. 2017

Coordinator: BSC

Partners:

Barcelona Supercomputing Center - Centro Nacional de Supercomputacion (Spain)

Atos/Bull (France)

ARM (United Kingdom)

Jülich (Germany)

LRZ (Germany)

University of Stuttgart (Germany)

CINECA (Italy)

CNRS (France)

CEA (France)

University of Bristol (United Kingdom)

Allinea Software (United Kingdom)

University of Cantabria (Spain)

Inria contact: Olivier Aumage

The Mont-Blanc project aims to develop a European Exascale approach leveraging on commodity power-efficient embedded technologies. The project has developed a HPC system software stack on ARM, and will deploy the first integrated ARM-based HPC prototype by 2014, and is also working on a set of 11 scientific applications to be ported and tuned to the prototype system.

8.2.2. Collaborations in European Programs, Except FP7 & H2020

Program: PRACE
Project acronym: PRACE-5IP
Project title: PRACE Fifth Implementation Phase
Duration: 01/2017
Coordinator: PRACE
Abstract: The objectives of PRACE-5IP are to build on and seamlessly continue the successes of PRACE and start new innovative and collaborative activities proposed by the consortium. These include:

- assisting the transition to PRACE2 including analysis of TransNational Access;
- strengthening the internationally recognised PRACE brand;
- continuing and extend advanced training which so far provided more than 18 800 person-training days;
- preparing strategies and best practices towards Exascale computing;
- coordinating and enhancing the operation of the multi-tier HPC systems and services;
- supporting users to exploit massively parallel systems and novel architectures.

A high level Service Catalogue is provided. The proven project structure will be used to achieve each of the objectives in 6 dedicated work packages. The activities are designed to increase Europe's research and innovation potential especially through:

- seamless and efficient Tier-0 services and a pan-European HPC ecosystem including national capabilities;
- promoting take-up by industry and new communities and special offers to SMEs;
- implementing a new flexible business model for PRACE 2;
- proposing strategies for deployment of leadership systems;
- collaborating with the ETP4HPC, CoEs and other European and international organisations on future architectures, training, application support and policies.

Inria contact for team STORM: Olivier Aumage

9. Dissemination

9.1. Promoting Scientific Activities

9.1.1. Scientific Events Organisation

9.1.1.1. Member of the Organizing Committees

- Samuel Thibault has organized a workshop panel at HCW'17

9.1.2. Scientific Events Selection

9.1.2.1. Chair of Conference Program Committees

- Denis Barthou has been Chair of EuroPar 2017

9.1.2.2. Member of the Conference Program Committees

- Denis Barthou has been Program Committee member of Europar 2017, UCHPC'17, HPCS'17, PACT'17
- Olivier Aumage has been Program Committee member of the Cluster 2017 conference.
- Samuel Thibault has been Program Committee member of P3MA'17, HCW'17, EuroPar 2017, ComPAS'2017.
- Raymond Namyst has been Program Committee member of PPAM'17, SC'17 and EuroMPI/USA'17.

9.1.2.3. Reviewer

STORM members have conducted wide reviewing activities for the following conferences and workshops: CCGRID, Cluster, EuroMPI, P3MA, PDSEC, HCW, Compas.

9.1.3. Journal

9.1.3.1. Reviewer - Reviewing Activities

STORM members have conducted reviewing activities for the following journals: IEEE TPDS, GPAA IJPEDS, PARCO, Scientific Programming, JPDC

9.1.4. Invited Talks

- Denis Barthou has been invited at the PASC conference mini-symposium titled "From linear algebra to High performance code", 26/06/2017 Lugano
- Denis Barthou has been invited at a Bird of Feather session at SC conference, 2017-11.
- Olivier Aumage has been invited at the special day event 'Runtimes' organized by Group 'Calcul' from CNRS (2017/01/20, PARIS).
- Samuel Thibault has been invited at the "Journées Scientifiques Inria 2017".
- Samuel Thibault has been invited at the HPC Workshop at Total (2017-06-29, Pau).
- Samuel Thibault has been invited at the JLESC workshop (2017-07-20, Urbana-Champaign, US).
- Samuel Thibault has been invited at the "École thématique GPU" (2017-11-09, Grenoble).
- Raymond Namyst has been invited to the Dagstuhl Seminar on Performance Portability in Extreme Scale Computing: Metrics, Challenges, Solutions (2017-10, Schloss Dagstuhl, Germany).

9.1.5. Scientific Expertise

- Olivier Aumage represents Inria at the OpenMP ARB consortium of standardization for the OpenMP language.
- Samuel Thibault participates to the HiHAT working group.
- Raymond Namyst is member of the Khronos OpenCL advisory panel.

9.2. Teaching - Supervision - Juries

9.2.1. Teaching

Licence: Samuel Thibault is responsible for the computer science topic of the first university semester.

Licence: Samuel Thibault is responsible for the creation of the new Licence Pro ADSILLH (Administrateur et Développeur de Systèmes Informatiques sous Licences Libres et Hybrides).

Licence: Samuel Thibault, Introduction to Computer Science, 32HeTD, L1, University of Bordeaux.

Licence: Samuel Thibault, Networking, 80HeTD, L3, University of Bordeaux.

Licence: Samuel Thibault, Tutored project, 5HeTD, L3, University of Bordeaux.

Licence: Marie-Christine Counilh, Introduction to Computer Science, 64HeTD, L1, University of Bordeaux.

Licence: Marie-Christine Counilh, Introduction to C programming, 52HeTD, L1, University of Bordeaux.

Master MIAGE: Marie-Christine Counilh, Object oriented programming in Java, 30HeTD, M1, University of Bordeaux.

Master: Marie-Christine Counilh, Internship supervision, 4HeTD, M2, University ob Bordeaux.

Engineering School: Emmanuelle Saillard, Structures arborescentes, 16HeTD, L3, ENSEIRB-MATMECA.

Engineering School: Pierre Huchant, Projet d'algorithmique et de programmation, 30HeTD, L3, ENSEIRB-MATMECA.

Engineering School: Pierre Huchant, Programmation Systeme, 18HeTD, M1, ENSEIRB-MATMECA.
Engineering School: Pierre Huchant, Compilation, 14HeTD, M1, ENSEIRB-MATMECA.
Engineering School: Pierre Huchant, Projet de Compilation, 4HeTD, M1, ENSEIRB-MATMECA.
Engineering School: Olivier Aumage, High Performance Communication Libraries, 20HeTD, M2, ENSEIRB-MATMECA.
Engineering School: Olivier Aumage, Languages and Supports for Parallelism, 14HeTD, M2, ENSEIRB-MATMECA.
Engineering School: Denis Barthou, Parallel architectures, 24HeTD, M2, ENSEIRB-MATMECA,
Engineering School: Denis Barthou, 3D synthesis in real time, 35HeTD, M2, ENSEIRB-MATMECA,
Engineering School: Denis Barthou, Architecture, 50HeTD, L3, ENSEIRB-MATMECA,
Engineering School: Denis Barthou, Tutored project, 25HeTD, M1, ENSEIRB-MATMECA,
Engineering School: Denis Barthou, Compilation, 28HeTD, M1, ENSEIRB-MATMECA,
Engineering School: Denis Barthou, Compilation, 28HeTD, M1, ENSEIRB-MATMECA,
Engineering School: Denis Barthou, C project, 30HeTD, L3, ENSEIRB-MATMECA,
Engineering School: Denis Barthou was responsible till Sept 2017 of a 3rd year specialization in cybersecurity, system and network (M2) at ENSEIRB-MATMECA, certified by the ANSSSI. From Sept 2017, he is responsible of the computer science teaching departement of ENSEIRB-MATMECA.
Licence: Pierre-André Wacrenier, Introduction to Computer Science, 64HeTD, L1, University of Bordeaux.
Licence: Pierre-André Wacrenier, System programming, 32HeTD, L3, University of Bordeaux
Master: Pierre-André Wacrenier, Parallel programming, 48HeTD, M1, University of Bordeaux
Master: Pierre-André Wacrenier is responsible for the Master's Degree with specialization in HPC, M2, University of Bordeaux
University of Bordeaux: Raymond Namyst is vice-chair of the computer science training department
Licence: Raymond Namyst, UNIX programming, 54HeTD, L3, University of Bordeaux
Master: Raymond Namyst, Operating Systems, 60HeTD, M1, University of Bordeaux
Master: Raymond Namyst, Parallel programming, 33HeTD, M1, University of Bordeaux
Engineering School: Raymond Namyst, Microprocessors, 24HeTD, L3, ENSEIRB-MATMECA
Engineering School: Raymond Namyst, Parallel programming, 33HeTD, M1, ENSEIRB-MATMECA

9.2.2. Supervision

PhD in progress : Pierre Huchant, October 2015, M.-C. Counilh, D.Barthou and R. Namyst
PhD in progress : Adrien Cassagne, October 2017, O. Aumage, D. Barthou, C. Jego and C. Leroux
PhD in progress : Léo Villeveygoux, October 2017, R. Namyst
PhD in progress : Terry Cojean, October 2014, P.-A. Wacrenier and R. Namyst
PhD in progress : Hugo Brunie, October 2015, J. Jaeger, P. Carribault and D. Barthou
PhD in progress : Raphael Prat, October 2017, R. Namyst
PhD in progress : Arthur Loussert, October 2015, J. Jaeger, P. Carribault and R. Namyst
PhD: Christopher Haine, January 2014-June 2017, O. Aumage and D. Barthou
PhD: Marc Sergent, Passage à l'échelle d'un support d'exécution à base de tâches pour l'algèbre linéaire dense, October 2014-January 2017, O. Aumage, S. Thibault and R. Namyst
PhD: Suraj Kumar, Scheduling of Dense Linear Algebra Kernels on Heterogeneous Resources, Université de Bordeaux, April 2017, E. Agullo, O. Beaumont, L. Eyraud, S. Thibault

9.2.3. Juries

Denis Barthou has participated to the following PhD/HDR juries
Nabil Hallou, PhD, U.Rennes, dec. 2017 (reviewer)
Abdou Guermouche, HDR, U. Bordeaux, dec. 2017 (member)

Raymond Namyst has participated to the following PhD/HDR committees:

- Sébastien Gougeaud, PhD, U. Versailles Saint-Quentin, may 2017 (reviewer)
- Yunsong Wang, PhD, Ecole Polytechnique, dec. 2017 (reviewer)
- Pierre Ramet, HDR, U. Bordeaux, dec. 2017 (member)

9.3. Popularization

- Séminaire Inria Bordeaux Sud-Ouest, Unithé ou Café, septembre 2017 sur la programmation parallèle (D. Barthou)
- Fête de la Science à l'Inria Bordeaux Sud-Ouest, ateliers Datagramme et Digit'elles, octobre 2017 (E. Saillard)
- Fête de la Science, ateliers Datagramme et Sciences Manuelles du Numérique, octobre 2017 (Y. Khorsi et C. Salingue)
- Printemps de la Mixité, atelier Datagramme, avril 2017 (C. Salingue)
- J'ai un bug, qu'est-ce que je peux faire ?, Mars 2017 (S. Thibault)
- Mon code en 180 secondes: StarPU+KStar, Journées SED Bordeaux, October 2017 (O. Aumage)

10. Bibliography

Major publications by the team in recent years

[1] E. AGULLO, O. AUMAGE, B. BRAMAS, O. COULAUD, S. PITOISET. *Bridging the gap between OpenMP and task-based runtime systems for the fast multipole method*, in "IEEE Transactions on Parallel and Distributed Systems", April 2017, 14 p. [*DOI :* 10.1109/TPDS.2017.2697857], https://hal.inria.fr/hal-01517153

[2] O. BEAUMONT, L. EYRAUD-DUBOIS, S. KUMAR. *Approximation Proofs of a Fast and Efficient List Scheduling Algorithm for Task-Based Runtime Systems on Multicores and GPUs*, in "IEEE International Parallel & Distributed Processing Symposium (IPDPS)", Orlando, United States, May 2017, https://hal.inria.fr/hal-01386174

Publications of the year

Doctoral Dissertations and Habilitation Theses

[3] S. KUMAR. *Scheduling of Dense Linear Algebra Kernels on Heterogeneous Resources*, Université de Bordeaux, April 2017, https://tel.archives-ouvertes.fr/tel-01538516

Articles in International Peer-Reviewed Journals

[4] E. AGULLO, O. AUMAGE, B. BRAMAS, O. COULAUD, S. PITOISET. *Bridging the gap between OpenMP and task-based runtime systems for the fast multipole method*, in "IEEE Transactions on Parallel and Distributed Systems", April 2017, 14 p. [*DOI :* 10.1109/TPDS.2017.2697857], https://hal.inria.fr/hal-01517153

[5] E. AGULLO, O. AUMAGE, M. FAVERGE, N. FURMENTO, F. PRUVOST, M. SERGENT, S. THIBAULT. *Achieving High Performance on Supercomputers with a Sequential Task-based Programming Model*, in "IEEE Transactions on Parallel and Distributed Systems", 2017, forthcoming [*DOI :* 10.1109/TPDS.2017.2766064], https://hal.inria.fr/hal-01618526

International Conferences with Proceedings

[6] O. AUMAGE, J. BIGOT, H. COULLON, C. PÉREZ, J. RICHARD. *Combining Both a Component Model and a Task-based Model for HPC Applications: a Feasibility Study on GYSELA*, in "17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid)", Madrid, Spain, May 2017, https://hal.inria.fr/hal-01518730

[7] O. AUMAGE, C. HAINE, D. BARTHOU. *Rewriting System for Profile-Guided Data Layout Transformations on Binaries*, in "Euro-Par 2017 - 23rd International European Conference on Parallel and Distributed Computing", Santiago de Compostela, Spain, Euro-Par 2017: Euro-Par 2017: Parallel Processing, Spinger, August 2017, pp. 260-272 [*DOI :* 10.1007/978-3-319-64203-1_19], https://hal.inria.fr/hal-01666179

[8] O. BEAUMONT, L. EYRAUD-DUBOIS, S. KUMAR. *Approximation Proofs of a Fast and Efficient List Scheduling Algorithm for Task-Based Runtime Systems on Multicores and GPUs*, in "IEEE International Parallel & Distributed Processing Symposium (IPDPS)", Orlando, United States, May 2017, https://hal.inria.fr/hal-01386174

[9] L. STANISIC, L. C. MELLO SCHNORR, A. DEGOMME, F. C. HEINRICH, A. LEGRAND, B. VIDEAU. *Characterizing the Performance of Modern Architectures Through Opaque Benchmarks: Pitfalls Learned the Hard Way*, in "IPDPS 2017 - 31st IEEE International Parallel & Distributed Processing Symposium (RepPar workshop)", Orlando, United States, June 2017, https://hal.inria.fr/hal-01470399

Research Reports

[10] E. AGULLO, B. BRAMAS, O. COULAUD, M. KHANNOUZ, L. STANISIC. *Task-based fast multipole method for clusters of multicore processors*, Inria Bordeaux Sud-Ouest, March 2017, n^o RR-8970, 15 p. , https://hal.inria.fr/hal-01387482

[11] E. AGULLO, B. BRAMAS, O. COULAUD, L. STANISIC, S. THIBAULT. *Modeling Irregular Kernels of Task-based codes: Illustration with the Fast Multipole Method*, Inria Bordeaux, February 2017, n^o RR-9036, 35 p. , https://hal.inria.fr/hal-01474556

Other Publications

[12] V. GARCIA PINTO, L. M. SCHNORR, L. STANISIC, A. LEGRAND, S. THIBAULT, V. DANJEAN. *A Visual Performance Analysis Framework for Task-based Parallel Applications running on Hybrid Clusters*, October 2017, working paper or preprint, https://hal.inria.fr/hal-01616632

References in notes

[13] O. AUMAGE, B. VIDEAU, J.-F. MÉHAUT. *MAQAO in BOAST*, Inria,CNRS,UJF, 2017

[14] M. SERGENT, D. GOUDIN, S. THIBAULT, O. AUMAGE. *Controlling the Memory Subscription of Distributed Applications with a Task-Based Runtime System*, in "21st International Workshop on High-Level Parallel Programming Models and Supportive Environments", Chicago, United States, May 2016, https://hal.inria.fr/hal-01284004