Master 2 internship proposal

Compiler support for data-driven parallelism

Advisors: Christophe Alias (Inria), Jean-Baptiste Besnard (Paratools), Patrick Carribault (CEA) and Julien Jaeger (CEA)
mail: `Christophe.Alias@ inria.fr`
web: `http://perso.ens-lyon.fr/christophe.alias`

Duration: 6 months (stip-end $\approx$ 1000 euros/month)

Place: Paratools (Bruyères-le-Châtel) or ENS de Lyon (Lyon)

Context

Since the early days of parallel computing, industry is pushing towards programming models, languages and compilers to help the programmer in the tedious task to parallelize a program. *Task-based programming models* [1, 8, 7] view the program as a composition of coarse-grain *tasks* to be executed in a dataflow fashion. A runtime is generally in charge of orchestrating the computation, and mapping the tasks to processing units so load balancing and data transfers are optimized. Usually, the task partitioning is driven algorithmically, (eg BLAS calls for StarPU), though better, but less intuitive, partionings may be inferred at compile-time [2] with compilation techniques from the *polyhedral model* [6].

Goals

In this internship, we investigate polyhedral compilation techniques to translate a sequential C kernel into a task-based program, and to guide the runtime for an optimized parallel execution. We focus on a *new runtime paradigm*, where processing units are viewed as data storages offering a *computing service*. During the execution, the processing units receive the *code* of the task to be executed, not the data (or as less as possible). Starting from a sequential C kernel, the intern will:

- Exploit affine loop tiling [3] to partition the computation into tasks at *compile-time* and propose a *specification langage* for the partitioning.
- Identify the compile-time informations required by the runtime (data dependences, in/out data flow, etc) and investigate compiler algorithms to compute them.
- Test the proposed approach on scientific computing benchmarks.

Skills expected. Notions in compilers, parallelism and experience with C++.

References

[1] Cédric Augonnet, Samuel Thibault, Raymond Namyst, and Pierre-André Wacrenier. Starpu: a unified platform for task scheduling on heterogeneous multicore architectures. *Concurrency and Computation: Practice and Experience*, 23(2):187–198, 2011.

[2] Uday Bondhugula, Vinayaka Bandishti, and Irshad Pananilath. Diamond tiling: Tiling techniques to maximize parallelism for stencil computations. *IEEE Transactions on Parallel and Distributed Systems*, 2016.

[3] Uday Bondhugula, Albert Hartono, J. Ramanujam, and P. Sadayappan. A practical automatic polyhedral parallelizer and locality optimizer. In *Proceedings of the ACM SIGPLAN 2008 Conference on Programming Language Design and Implementation, Tucson, AZ, USA, June 7-13, 2008*, pages 101–113, 2008.

[4] Philippe Clauss. Counting solutions to linear and nonlinear constraints through ehrhart polynomials: Applications to analyze and transform scientific programs. In *Proceedings of the 10th international conference on Supercomputing, ICS 1996, Philadelphia, PA, USA, May 25-28, 1996*, pages 278–285, 1996.

[5] Paul Feautrier. Dataflow analysis of array and scalar references. *International Journal of Parallel Programming*, 20(1):23–53, 1991.

[6] Paul Feautrier and Christian Lengauer. Polyhedron model. In *Encyclopedia of Parallel Computing*, pages 1581–1592. 2011.

[7] Josep M Perez, Rosa M Badia, and Jesus Labarta. A dependency-aware task-based programming environment for multi-core architectures. In *Cluster Computing, 2008 IEEE International Conference on*, pages 142–151. IEEE, 2008.

[8] Asim Yarkhan, Jakub Kurzak, and Jack Dongarra. Quark users' guide. *Electrical Engineering and Computer Science, Innovative Computing Laboratory, University of Tennessee*, 2011.