Master 2 internship proposal

Compiler-guided runtime scheduling under resource constraints

Advisors: Christophe Alias (Inria & ENS de Lyon), Jean-Baptiste Besnard (Paratools)
mail: `Christophe.Alias@ inria.fr`
web: `http://perso.ens-lyon.fr/christophe.alias`

Duration: 4 – 6 months (stip-end ≈ 500 euros/month)

Place: ENS de Lyon (Lyon). Teleworking might be negociated depending on the rules of your institution.

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, 6, 4] view the program as a composition of coarse-grain *tasks* to be executed in a dataflow fashion. The tasks are *submitted* to a runtime, in charge of orchestrating the computation and the data transfers to optimize execution metrics (latency, resource usage, energy, etc). Usually, scheduling decisions are based on a small window of tasks submitted, which may lead to sub-optimal performances.

Goals

In this internship, we investigate how a compiler might *infer an optimum submission order* for a runtime. The long-term goal is to put the compiler in the center of the parallelization process. We wish to investigate how compilers might infer information for coarse-grain parallelism, directly exploitable by runtimes; thereby reducing the runtime overhead and improving parallelization decisions. We focus on the *polyhedral model* [3], a mathematical framework to analyse, schedule and generate programs, focusing on compute-intensive loop kernels. As for the runtime, we focus on OpenMP [2] which features a task system and a simple annotation syntax. Specifically, the internship will address the following points:

- Given an execution and its task graph, implement a simple scheduling algorithm to minimize the overall latency.
- Infer a general polyhedral schedule prescribing the same (or almost the same) execution order and generate the corresponding OpenMP-annotated task program.

The performances will be evaluated on the benchmarks of the polyhedral community [5].

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] Leonardo Dagum and Ramesh Menon. Openmp: an industry standard api for shared-memory programming. *IEEE computational science and engineering*, 5(1):46–55, 1998.

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

[4] 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.

[5] Louis-Noël Pouchet. Polybench: The polyhedral benchmark suite. *URL: http://www. cs. ucla. edu/˜ pouchet/software/polybench/[cited July,]*, 2012.

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