Laboratoire de l'Informatique du Parallélisme ROMA team
(UMR CNRS - ENS Lyon - UCB Lyon 1 - INRIA 5668)

internship proposal - Master 2

Memory-aware scheduling fot the StarPU runtime

Loris Marchal, Samuel Thibault

Keywords: Parallel computing, scheduling, algorithms.

Advisors:

Loris Marchal: researcher, CNRS & Univ. de lyon,
(http://perso.ens-lyon.fr/loris.marchal/, loris.marchal@ens-lyon.fr)

Samuel Thibault: associate professor, Université de Bordeaux & INRIA,
on leave in the ROMA team in february–july 2019,
(http://dept-info.labri.fr/~thibault/index.html,samuel.thibault@inria.fr)

Research team

The internship will take place in the ROMA team, of the "Laboratoire de l'Informatique du Parallélisme". The ROMA team focuses on designing parallel algorithms and schedules for High Performance Computing platforms, in particular for scientific applications.

Context of the internship

Parallel computing platforms are used, among others, to perform large-scale numerical simulations and are becoming increasingly complex, as they now include computing accelerators, deep memory hierarchies, and the combination of shared and distributed memories. On way to take advantage of their raw power despite this complexity is to rely on lightweight scheduling runtimes, such as the StarPU runtime [1] (whose development is led by Samuel Thibault). A scientific computing application is then described as a directed graph of tasks, where arcs represent dependencies between tasks. The scheduler allocates available tasks on the various computing resources of the platform. One of the problem to efficiently solve such a scheduling problem is linked to memory: we have to avoid the scenario where the platform runs out of memory, otherwise the application would crash (or experienced very bad performance because of swapping).

Solutions to this problem have been proposed. First, schedulers taking into account the amount of available memory have been proposed for the case of a single computing resource [5] or for specific task graphs on parallel resources [2]. More recently, we have proposed a method to add new dependencies to the task graph such that any valid schedule of the resulting graph is guaranteed not to exceed a given amount of memory [6]. The benefit of this method is to treat memory bottlenecks before the execution, and to allow for runtime schedulers, which are able to dynamically adapt to changing conditions in the computing platforms. Unfortunately, the complexity of the algorithm and the large number of added dependencies make this method hard to use in practice.

Objective of the internship

The goal of this internship is to improve this method, by limiting the needed guarantees: instead of proving that all execution will not exceed the memory (the actual guarantee), it is indeed enough to make sure that no memory bottlenecks will happen, as runtime schedulers such as StarPU are able to block tasks requesting memories before some other tasks release enough memory. This way, we would be able to dramatically limit the number of new dependencies to add to the graph.

The internship will consist of the following steps.

1. Model the problem in graph concepts, by selecting the most appropriate memory model and by expressing the objective as a graph property.
2. Propose algorithmic solutions to solve the problem, with a special care on the complexity of the solutions.
3. Test the proposed solution first by simulation, on actual application graphs.
4. Implement some of these solutions on the StarPU runtime.

The distribution of the work between theory (designing algorithms and proving them) and practice (simulations and real implementation) is flexible and will depend on the skills and will of the candidate.

Required skills

The candidate need a good skill in algorithms, to be confident with programming in C or other common programming languages.

Added value for the intern

The intern will have the opportunity to join a dynamic research group, to acquire knowledges in scheduling and parallel computing, to increase his/her skills in programming and in the development of in the most recent tools for parallel computing.

Bibliography

[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] Lionel Eyraud-Dubois, Loris Marchal, Oliver Sinnen, and Frédéric Vivien. Parallel scheduling of task trees with limited memory. *TOPC*, 2(2):13:1–13:37, 2015.

[3] P. Hénon, P. Ramet, and J. Roman. PaStiX: A High-Performance Parallel Direct Solver for Sparse Symmetric Definite Systems. *Parallel Computing*, 28(2):301–321, January 2002.

[4] Joseph Y.-T. Leung, editor. *Handbook of Scheduling - Algorithms, Models, and Performance Analysis*. Chapman and Hall/CRC, 2004.

[5] Joseph W. H. Liu. An application of generalized tree pebbling to sparse matrix factorization. *SIAM J. Algebraic Discrete Methods*, 8(3):375–395, 1987.

[6] Loris Marchal, Hanna Nagy, Bertrand Simon, and Frédéric Vivien. Parallel scheduling of dags under memory constraints. In *IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pages 204–213, 2018.

[7] Alex Pothen and Chunguang Sun. A mapping algorithm for parallel sparse cholesky factorization. *SIAM Journal on Scientific Computing*, 14(5):1253–1257, 1993.

[8] Yves Robert. Task graph scheduling. In *Encyclopedia of Parallel Computing*, pages 2013–2025. Springer, 2011.