Post-doctoral position in the INRIA ROMA project-team

Scheduling DAGs for memory footprint

Loris Marchal, Bora Uçar

Advisors:

Loris Marchal (chargé de recherche, CNRS),
Bora Uçar (chargé de recherche, CNRS).

Contacts:

Loris.Marchal@ens-lyon.fr, tel: 04 72 72 87 58, http://graal.ens-lyon.fr/~lmarchal/
Bora.Ucar@ens-lyon.fr, tel: 04 72 72 89 32, http://perso.ens-lyon.fr/bora.ucar/

Starting date:

February 1st, 2017 (or earlier)

Host team:

The postdoc will be carried out at the Inria team ROMA, within the LIP (Laboratoire de l'Informatique du Parallélisme, UMR CNRS - ENS Lyon - UCB Lyon 1 - INRIA 5668), located at the École normale supérieure de Lyon. ROMA is pursuing fundamental research on scheduling strategies and algorithm design for heterogeneous platforms, and direct solvers for sparse linear systems. The research activities of ROMA are organized around three themes, including one around combinatorial scientific computing and sparse direct solvers. The post-doc will be carried out with the supervision of Bora Uçar (CR1) and Loris Marchal (CR1) for a duration of 18 months.

Subject:

Parallel workloads are often modeled as tasks graphs that are directed and acyclic (DAGs), where nodes represent tasks and edges represent the dependencies between tasks. There is an abundant literature on task parallel graph scheduling where the objective is to minimize the total completion time, or makespan [7, 6]. However, as the size of the data to be processed increases, another objective is to minimize the memory footprint of the application. Modern computing platforms exhibit a complex memory hierarchy ranging from caches to RAM and disks and even sometimes to tape storage, with the classical property that the smaller the memory, the faster [4]. Thus, to avoid large makespan one must strive to keep the application memory footprint low, so as to avoid using low bandwidth memory as much as possible.

Each task in a task graph takes one or more input files, and generates one output file. To be executed, a task also requires an execution file. We consider the execution of the task graph on a single compute resource (a single processor and a single memory hierarchy), shared memory systems (multi/manycore processors, sharing one or two types of memory), and distributed memory systems (several nodes with their own memory, each node being a multi/manycore processor). All files have different sizes, and an execution of the application is defined by a traversal of the graph. The memory footprint of a traversal is the maximum amount of memory needed to store application files at any given time throughout the application execution. The question we study in this work is: How should the task graph be traversed so as to minimize the memory footprint?

We have looked at this problem in the context of multifrontal sparse direct solvers where the task graph is a tree. We proposed polynomial time algorithms for scheduling on single processor systems [5] and developed NP-completeness results for scheduling on shared memory systems [3]. We also studied the problem for a slightly broader class of graph, namely series-parallel graphs, for which we have designed an optimal algorithm. To tackle general graphs, a simple possibility is to transform a graph into a series-parallel graph, by adding fictitious dependencies, and then to apply the optimal algorithm for series-parallel graphs. Other heuristic solutions may be designed, possibly inspired by the literature in graph partitioning.

While the previous studies give us an insight on how to schedule a task graph on a single processor to minimize the memory footprint, our objective is to design scheduling strategies for parallel processing, to minimize the makespan when the memory is limited. We would like to tackle both the shared-memory and distributed-memory cases, although the second case adds a level of difficulty.

The objective of this post-doc is both to design memory-aware scheduling strategies for this problem and to implement and test them on task graphs coming from (dense or sparse) linear algebra, using both simulations (for example by using SimGrid [2] for the distributed memory systems) and real experiments relying on a runtime system such as starPU [1] (for the shared-memory case)..

Expertise/skills

The candidate should have a strong expertise in algorithms and scheduling, and writing and speaking skills in English. An experience in linear algebra is appreciated but not required.

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, Special Issue: Euro-Par 2009*, 23:187–198, February 2011.

[2] Henri Casanova, Arnaud Giersch, Arnaud Legrand, Martin Quinson, and Frédéric Suter. Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917, June 2014.

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

[4] Susan L Graham, Marc Snir, Cynthia A Patterson, et al. *Getting up to speed: The future of supercomputing*. National Academies Press, 2005.

[5] Mathias Jacquelin, Loris Marchal, Yves Robert, and Bora Uçar. On optimal tree traversals for sparse matrix factorization. In *25th IEEE International Symposium on Parallel and Distributed Processing, IPDPS 2011, Anchorage, Alaska, USA, 16-20 May, 2011 - Conference Proceedings*, pages 556–567, 2011.

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

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