**Project Team INRIA**: Cepage - HiePACS – Runtime

**Author of the proposal research subject**: E. Agullo (HiePACS); O. Beaumont (Cepage); S. Thibault (Runtime)

**Title of the proposal research subject:** Task-based programming paradigms and scheduling

**Scientific context (10-15 lines):**

With the advent of the Message Passing Interface (MPI) standard, Single Program Multiple Data (SPMD) has become the dominant parallel programming model in High Performance Computing (HPC). Most industrial-quality HPC libraries and applications implementing this paradigm may be roughly viewed as a two-step approach: a first step aims at performing an a priori load balancing and the actual computation is then performed statically in a second step. However, if the computation step has a complex pattern or a dynamic behavior, such a static approach may not be appropriate. For this reason, a few SPMD industrial-quality codes embed dynamic schedulers, which may decide which processor will execute a subpart of the work to be done. In order to achieve the highest possible performance, these schedulers are often tightly coupled with the computational sections of the code. Alternatively, the computational part of the code may be viewed (and designed) as a generic set of tasks to be scheduled, in which case amodular approach may be employed: a scheduling algorithm decides where to execute the tasks (and possibly in which order) while a runtime system handles the data consistency and orchestrates their actual execution on the available computational units, possibly providing online feedback to the scheduler.

**Goal (10-15 lines)**

The main objective of this PhD thesis is to propose a unified approach for scheduling large application graphs on future Exascale platforms. The main difficulty is to decide which fraction of the tasks should be allocated statically, and which fraction dynamically. In both cases, one has to determine the policy driving task graph exploration, physical resource allocation and migration-based or work stealing-based load-balancing.

**Project (10-15 lines)**

The architectural complexity of supercomputers has strongly increased in the past ten years in order to cope with the required degree of parallelism. All top supercomputers indeed now rely on nested parallelisms: manycore processors are aggregated with fast interconnects. Such manycore processors may furthermore be themselves hierarchical, such as cache coherent Non Uniform Memory Access (ccNUMA) processors. They may also be augmented with multiple accelerators, such as General Purpose Graphics Processing Units (GPGPUs) and co-processors. In this context, writing an efficient ad-hoc industrial-quality code for a given platform becomes extremely hard and time-consuming, even for the most regular of these codes. On the other hand, the modularity of the task-based approach allows the computational part of the code to be written once and then rely on advanced runtime systems and scheduling algorithms in order to benefit from the whole potential of the platform.

Such task-based algorithms may be represented with a directed acyclic graph (DAG) where a vertex is a task and an edge is a dependency between tasks. Although the runtime system has to parse the whole set of tasks and dependencies, so-called Enumerated Task Graph (ETG), providing it explicitly may be neither productive nor efficient. The ETG may indeed be inferred from a (valid) sequential list of the tasks, based on data hazards. This Sequential Task Flow (STF) programming paradigm is inherently sequential and thus ensures a very high productivity. Alternatively, in some cases, the dependencies may be expressed as rules parameterized with the indices of the task instances, leading to a much more compact expression. This Parameterized Task Graph (PTG) paradigm is harder to program (the expression of the dependencies may not be simple); on the other hand, it provides more flexibility to decide which task to schedule and allows a natural parallel traversal of the task graph. The flexibility offers scheduling opportunities while the parallel traversal is a requirement to ensure an efficient scalability.

The goal of this PhD thesis is to bridge the gap between STF and PTG programming paradigms thanks to automatic parallelization mechanisms and exploit the expressivity of each paradigm to design advanced task-based scheduling algorithms. A framework unifying these paradigms will be designed, based on the StarPU and ParSEC state-of-the-art runtime systems, which implement the STG and PTG paradigms, respectively. The impact of the resulting scheduling algorithms will be assessed on the performance of the task-based Magma dense linear algebra library, the PaSTiX sparse direct solver and the ScalFMM fast multipole method library.

**Advisors:** *E. Agullo, O. Beaumont, S. Thibault*

**Keywords (max 5-6 lines):** *high-performance computing, task-based parallelism, solvers, runtime, scheduling*

**Duration**: *36 months*