Parallel programming issues, achievements and trends in high-performance and adaptive computing

Marco Danelutto, Marco Vanneschi, C. Bertolli, D. Buono, A. Mencagli, A. Pascucci

joint work with: M. Aldinucci, P. Kilpatrick, M. Meneghin, R. Ravazzolo, M. Torquati

The area of parallel high-performance programming is currently receiving a growing attention, owing to the technological (r)evolution of computer components and CPUs based on single-chip multi/many-core architectures. This phenomenon will imply radical changes in IT community and industry attitude and products, as the sequential programming model assumption is put into a rather difficult position and, in some way, it could be replaced by the parallel programming model assumption [7].

Currently, it is difficult to foresee the implications of this (r)evolution on programming methodologies, tools and frameworks on the software technology and applications of the next future. What we can assert is that future applications must be able to effectively exploit the parallel structure of emerging high-performance architectures, while the existing applications should be able at least to exploit their scalability features.

In the area of parallel programming we have assisted to several approaches with different goals and characteristics: from low-level and low-productivity approaches based on communication libraries (MPI, OpenMP), to high-level approaches specialized towards some classes of application structures (HPF, HPC). The most notable attempts to achieve a good trade-off between programmability, software productivity, portability and performance belong to what we call the Structured Programming Model class, like algorithmic skeletons, parallel paradigms, and design patterns [9, 14]. Our research group has produced many recognized results in this area, notably P3L/SkE [8], ASSIST [15], Lithium [1], Muskel [3], a first adaptive (w.r.t. parallelism degree) version ASSIST [6], as well as contribution to high-performance software component technology with the Behavioural Skeleton concept [2], for several kinds of architectural platforms (multiprocessors, homogeneous and heterogeneous clusters, high-performance grids).

Recently, some new structured programming frameworks are emerging, e.g. Google MapReduce and Apache Hadoop, Intel TBB (Thread Building Blocks library), as well as the recent Microsoft TPL (Task Parallel Library), though characterized by some limitations as far as concerns expressive power and software productivity in complex application development.

The main feature of the Structured Parallel Programming Model lies in its ability to define a parallel computation as the composition of parametric parallel paradigms with known implementation models and cost models. This feature has been extended in the ASSIST programming model to any computation
structure described by generic graphs whose nodes apply the parametric parallel paradigms, thus allowing the programmer to express arbitrarily complex and irregular compositions of parallel structured components. It also been exploited in the Behavioural Skeletons to support composition of autonomic management of non functional features (e.g. performance, security, fault tolerance and power management) in addition to functional composition of “business logic” code.

The experience with Structured Parallel Programming has shown that a static optimization approach based on analytical cost models is effective only on some, though notable application classes. There are cases in which the dynamic or irregular structure of the computation, as well as the heterogeneous nature of the underlying platform, prevent to achieve good optimizations according to a static approach. Interesting examples are currently investigated in the FIRB In.Sy.Eme project [11] in the context of heterogeneous platforms for emergency management, based on a variety of fixed and mobile nodes and networks.

These classes of applications are characterized by the adaptive capability of dynamically restructuring the parallel applications in order to respect given QoS requirements which, in turn, can change dynamically during the computation and according to contextual situations. The autonomic view of systems and applications [13] corresponds to this approach towards adaptive and context-aware parallel programming. Some notable results have been achieved in the European projects CoreGrid [10] and GridComp [12]. In these projects and in the follow-up research activities, we demonstrated that rule based autonomic managers can be associated to component based structured parallelism exploitation patterns. These managers allow to ensure, best effort, single, user provided, non functional contracts (e.g. throughput or secure computation requirements) in hierarchical composition of parallel patterns [4], as well as coordinated ensuring of multiple non functional contracts, each one related to a different non functional concern [5].

In this presentation we highlight the relationship between Structured Parallel Programming and adaptive/autonomic computing, report on the current status and experience, and discuss some interesting trends in this area, including methodologies, tools and approaches for dynamically adaptive model of high-performance computations.

References


