PIT: A Library for the Parallelization of Irregular Problems

Fabrizio Baiardi, Paolo Mori, and Laura Ricci

Dipartimento di Informatica, Università di Pisa
Corso Italia 40, 56125 - Pisa (Italia)
{baiardi, mori, ricci}@di.unipi.it

Abstract. A problem is irregular if its solution requires the computation of some properties for each of a set of elements irregularly distributed in a domain of interest. These problems satisfy a locality property because the properties of an element depend upon those of a few other elements, its neighbors, according to a dynamic, problem dependent stencil. The development of parallel algorithms for irregular problems on distributed memory architectures is not trivial, because the irregularity and the dynamicity of the distribution of the elements in the domain require complex strategies to manage the mapping of elements onto the processing nodes and to implement the processing nodes cooperation. This paper introduces PIT, a library to simplify the parallelization of irregular problems. The key assumption underlying the definition of PIT is that both the sequential and the parallel version of the application are structured in terms of operations on a tree that describes the distribution of the elements in the domain. In the parallel version, the tree is handled in parallel through the functions supplied by PIT in a way that is transparent to the user and that preserves most of the sequential code.

1 Introduction

Several physical phenomena, such as the motion of the stars in a galaxy or the illumination of objects in an scene, are modelled by time dependent partial differential equations systems that are usually solved through adaptive iterative algorithms. An iterative algorithm computes the final result through a sequence of approximations, each produced by updating the previous one. The domain of an irregular problem consists of a set of elements distributed in an irregular and dynamic way in a space of interest. Each element is characterized by its properties, as defined in the specific irregular problem. For instance, if the Barnes Hut method for the nbody problem is applied to the motion of stars, each element corresponds to a star and its properties are mass, position in the space and speed vector. The properties of an element are updated by computing its interactions with a set of other elements close to the considered one (neighbors). The rule to determine the set of neighbors of an element (neighborhood stencil) depends upon the specific problem, but, in general, the number of neighbors is different for distinct elements and it changes during the computation. The number of
elements changes as well. For instance, it increases to improve the accuracy of the final solution. Hence, due to the irregularity of the distribution of the elements and of the neighborhood stencil, the computation of the evolution of some subsets of the domain requires a larger computational effort than others and these subsets change as the computation goes on.

The development of parallel applications for the solution of irregular problems is not trivial, because of the irregular and dynamic distribution of the elements in the domain, that requires complex strategies to map the elements onto the processing nodes, p-nodes, and for the communication management. Since no useful information about the distribution of the elements can be deduced by a static analysis or by program profiling, sophisticated run time mapping strategies have to be adopted to produce a parallel code that can achieve satisfactory performance values. In particular, load balancing is critical to improve these values.

This paper presents PIT, Parallel Irregular Trees, a library to parallelize irregular problems derived from the methodology to structure a parallel application for irregular problems on distributed memory architectures presented and evaluated in [1,2,3,4]. The aim of PIT is to provide a simple and complete tool for the development of parallel solutions to irregular problems while preserving most of the code developed for the sequential application. The key point of the PIT approach is that both the sequential and the parallel version of the application may be structured in terms of operations on a tree that describes the distribution of the elements in the domain. In the parallel version, the tree is distributed among the local memories of the p-nodes of the architecture and is handled in parallel in a transparent way through the PIT functions. In the current version, PIT functions are implemented through C and MPI, however other languages can be exploited.

Alternative approaches to irregular problems are LPARX [5], Chaos and Multiblock Parti, [6] and that presented in [8]. However, these approaches are focused on the data mapping techniques.

In the following, we describe the methodology and the strategies underlying the definition of PIT. Section 2 describes the methodology to solve irregular problems, and how the sequential code can be transformed in the parallel one using the PIT functions. Sections 3, 4, 5 describe the details of the main functions provided by PIT, the H-Tree_creation, the H-Tree_completion and the H-Tree_update.

Section 6 shows that the user knowledge of the specific irregular problem can be exploited to combine the PIT functions to produce a more efficient parallel code. Section 7 draws the conclusions.

2 PIT Parallelization Strategy

PIT is a library to develop parallel applications to solve irregular problems. Its main aim is to simplify the development of such algorithms on distributed memory architecture so that it can be exploited by users that are not acquainted