A Hierarchical Approach to Irregular Problems*

Fabrizio Baiardi, Primo Becuzzi, Sarah Chiti, Paolo Mori, and Laura Ricci

Dipartimento di Informatica, Universitá di Pisa
Corso Italia 40, 50125 - PISA
<last name>@di.unipi.it

Abstract. Irregular problems require the computation of some properties for a set of elements irregularly distributed in a domain in a dynamic way. Most irregular problems satisfy a locality property because the properties of an element e depend upon the elements "close" to e. We propose a methodology to develop a highly parallel solution based on load balancing strategies that respects locality, i.e. e and most of the elements close to e are mapped onto the same processing node. We present the experimental results of the application of the methodology to the n-boby problem and to the adaptive multigrid method.

1 Introduction

The solution of an irregular problem requires the computation of some properties for each of a set of elements that are distributed in a n-dimensional domain in an irregular way, that changes during the computation. Most irregular problems satisfy a locality property because the probability that the properties of an element e_i affects those of e_j decreases with the distance from e_i to e_j. Examples of irregular problems are the Barnes-Hut method [2], the adaptive multigrid method [3] and the hierarchical radiosity method [5].

This paper proposes a parallelization methodology for irregular problems in the case of distributed memory architectures with a sparse interconnection network. The methodology defines two load balancing strategies to, respectively, map the elements onto the processing nodes, p-nodes, and update the mapping as the distribution changes and a further strategy to collect information on elements mapped onto other p-nodes. To evaluate its generality, the methodology has been applied to the Barnes-Hut method for the n-body problem, NBP, and to the adaptive multigrid method, AMM. Sect. 2 describes the representation of the domain and the load balancing strategies and Sect. 3 presents the strategy to collect remote data. Experimental results are discussed in Sect. 4.

2 Data Mapping and Runtime Load Balancing

All the strategies in our methodology are defined in terms of a hierarchical representation of the domain and of the element distribution. At each hierarchical

* This work was partially supported by CINECA

A. Bode et al. (Eds.): Euro-Par 2000, LNCS 1900, pp. 218–222, 2000.

level, the domain is partitioned into a set of equal subdomains, or spaces. The hierarchy is described through the **Hierarchical Tree**, *H-Tree* [7, 8]; the root represents the whole domain, each other node N, **hnode**, represents a space, *space(N)*, and it records information on the elements in *space(N)*. A space A that violates a problem dependent condition, is partitioned into 2^n equal subspaces by halving each of its sides. A is partitioned if contains more than one body in the NBP, and if the current approximation error in its vertexes is larger than a threshold in AMM. The sons of N describe the partitioning of *space(N)*. In the following, *hnode(A)* denotes the hnode representing the space A, and the level of A is the depth of *hnode(A)* in the H-Tree. Hnodes representing larger spaces record a less detailed information than those representing smaller spaces. In the NBP, each leaf L records the mass, the position in the space and the speed vector of the body in *space(L)*, while any other hnode N records the center of gravity and the total mass of the bodies in *space(N)*. In the AMM, each hnode N records the coordinates, the approximated solution of the differential equation and the evaluation of the error of the point on the leftmost upward vertex of *space(N)*. At run time, the hierarchy and the H-Tree are updated according to the current elements distribution. Since the H-Tree is too large to be replicated in each p-node, we consider a subset that is replicated in each p-node, the RH-Tree, and one further subset, the private H-Tree, for each p-node.

To take locality into account, we define the initial mapping in three steps: spaces ordering, workload determination and spaces mapping onto p-nodes.

The spaces are ordered through a space filling curve *sf* built on the spaces hierarchy [6]; *sf* also defines a visit *v(sf)* of the H-Tree that returns a sequence $S(v(sf)) = [N_0, .., N_m]$ of hnodes. The load of a hnode N evaluates the amount of computations due to the elements in *space(N)*. In the NBP, the load of a leaf L is due to the computation of the force on the body in *space(L)*. This load is distinct for each leaf and it is measured during the computation, because it depends upon the current body distribution. No load is assigned to the other hnodes because no forces are computed on them. Since in the AMM the same computation is executed on each space, the same load is assigned to each hnode.

The *np* p-nodes are ordered in a sequence $SP = [P_0, .., P_{np}]$ such that the cost of an interaction between P_i and P_{i+1} is not larger than the cost of the same interaction between P_i and any other p-node. Since each p-node executes one process, P_k denotes also the process executed on the *k*-th p-node of SP.

S(v(sf)) is partitioned into *np* segments, whose overall load is as close as possible to *average_load*, the ratio between the overall load and *np*. We cannot assume that the load of each segment S is equal to *average_load* because each hnode is assigned to one segment; in the following, $= (S, C)$ denotes that the load of S is as close as possible to C. The first segment of $S(v(sf))$ is mapped onto P_0, the second onto P_1 and so on. This mapping satisfies the **range property**: *if the hnodes N_i and N_{i+j} are assigned to P_h, then all the hnodes in-between N_i and N_{i+j} in $S(v(sf))$, are assigned to P_h as well.* Due to the property of space filling curves, any mapping satifying this property allocates elements that are close to each other to the same p-node. Furthermore, two consecutive segments are mapped onto p-nodes that are close in the interconnection network.

PH-Tree(P_h), the private H-Tree of P_h, describes Do_h, the segment assigned to P_h, and includes a hnode N if *space(N)* belongs to Do_h. The RH-Tree is the union of the paths from the H-Tree root to the root of each private H-Tree; each hnode N records the position of *space(N)* and the owner process. In the NBP, a hnode N belongs to PH-Tree(P_h) iff all the leaves in *Sub(N)*, the subtree rooted in N, belong to this tree too, otherwise it belongs to the RH-Tree. To minimize the replicated data, the intersection among a private H-Tree and the RH-Tree includes the roots of the private H-Trees only. In the AMM, each hnode belongs to the private H-Tree of a p-node, because all hnodes are paired with a load.

Due to the body evolution in the NBP and to the grid refinement in the AMM, the initial allocation could result in an unbalance at a later iteration. The mapping is updated if the largest difference between *average_load* and the current workload of a process is larger than a tolerance threshold $T > 0$. Let us suppose that the load of P_h is *average_load* + C, $C > T$, while that of P_k, $h \neq k$, is *average_load* - C. To preserve the range property, the spaces are shifted among all the processes P_i in-between P_h and P_k. Let us define $Prec_i$ as the set $[P_0...P_{i-1}]$ and $Succ_i$ as the set $[P_{i+1}...P_{np}]$. Furthermore, $Sbil(PS)$ is the global load unbalances of the set PS. If $Sbil(Prec_i) = C > T$, i.e. processes in $Prec_i$ are overloaded, P_i receives from P_{i-1} a segment S where $= (S, C)$. If, instead, $Sbil(Prec_i) = C < -T$, P_i sends to P_{i-1} a segment S where $= (S, C)$. The same procedure is applied to $Sbil(Succ_i)$, but the hnodes are either sent to or received from P_{i+1}. To preserve the range property, if Do_i= $[N_q....N_r]$, then P_i sends to P_{i-1} a segment $[N_q....N_s]$, while it sends to P_{i+1} a segment $[N_t....N_r]$, with $q \leq t, s \leq r$.

3 Fault Prevention

To allow P_h to compute the properties of elements in Do_h whose neighbors have been mapped onto other p-nodes, we have defined the **fault prevention** strategy. The fault prevention strategy allows P_h to receive the properties of the neighbors of elements in Do_h without requesting them. Besides reducing the number of communications, this simplifies the applications of some optimization strategies such as messages merging. For each space A in Do_k, P_k determines, through the neighborhood stencil, which processes require the data of A and sends to these processes the data, without any explicit request. To determine the data needed by P_h, P_k exploits the information on Do_h in the RH-Tree. In general, P_k approximates these data because the RH-Tree records a partial information only. The approximation is always *safe*, i.e. it includes any data P_h needs, but, if it is not accurate, most data is useless. To improve the approximation, the processes may exchange some information about their private H-Trees before the fault prevention phase (**informed fault prevention**).

In the NBP, the neighborhood stencil of a body b is defined by the "Multipole Acceptability Criterium" (MAC), that determines, for each hnode N, whether the interaction between b and the bodies in *space(N)* can be approximated. A widely adopted definition of the MAC [2] is $\frac{l}{d} < \theta$, where l is the length of the side of *space(N)*, d is the distance between b and the center of gravity of the bodies in *space(N)* and θ is an user defined approximation coefficient. P_k computes the *influence space*, *is(N)*, for each hnode N that is not a leaf of PH-Tree(P_k). *is(N)* is a sphere with radius $\frac{l}{\theta}$ centered in the center of gravity recorded in N. Then, P_k visits PH-Tree(P_k) in anticipated way and, for each hnode N that is not a leaf, it computes $J(N, R) = is(N) \cap space(R)$ where R is the root of PH-Tree(P_h), $\forall h \neq k$. If $J(N, R) \neq \emptyset$, it may include one body d, and the approximation cannot be applied by P_h when computing the forces on d. Hence, P_h needs the information recorded in the sons of N in the PH-Tree(P_k). To guarantee the safeness of fault prevention, P_k assumes that $J(N, R)$ always includes a body, and it sends to P_h the sons of N. P_h uses these data iff $J(N, R)$ includes at least one body. If $J(N, R) = \emptyset$ then, for each body in Do_h, P_h approximates the interaction with N and it does not need the hnodes in *Sub(N)*.

In the AMM, P_h applies the multigrid operators, in the order stated by the V-cycle, to the points in Do_h [3, 4]. We denote by Bo_h the boundary of Do_h, i.e. the sets of the spaces in Do_h such that one of their neighbors does not belong to Do_h. Bo_h depends upon the neighborhood stencil of the operator *op* that is considered. Let us define $I_{h,op,liv}$ as the set of spaces not belonging to Do_h and including the points required by P_h to apply *op* to the points in the spaces at level *liv* of Bo_h. $\forall h \neq k$, P_k exploits the information in the RH-Tree about Do_h to determine the spaces in Do_k that belongs to $I_{h,op,liv}$. Hence, it computes and sends to P_h a set $A_k I_{h,op,liv}$ that approximates $I_{h,op,liv} \cap Do_k$. The values of points in $A_k I_{h,op,liv}$ are trasmitted just before the application of *op*, because they are updated by the previous operators in the V-cycle. To improve the approximation, we adopt *informed fault prevention*. If a space in Do_k belongs to $I_{h,op,liv}$, $k \neq h$, P_h sends to P_k, at the beginning of the V-cycle and before the fault prevention phase, the level of each space in Bo_h that could share a side with the one in Do_k. If the load balancing procedure has been applied, P_h sends the level of all the spaces in Bo_h, otherwise, since spaces are never pruned, P_h sends the level of the new spaces only.

4 Experimental Results

To evaluate the generality of our methodology, we have implemented the NBP on the Meiko CS 1 with OCCAM II as programming language and the AMM on a Cray T3E with C and MPI primitives. The data set for the NBP is generated according to [1]. The AMM solves the *Poisson's problem* in two dimensions subject to two different boundary conditions, denoted by $h1$ and $h2$:

$$h1(x, y) = 10 \qquad\qquad h2(x, y) = 10 \cos(2\pi(x - y)) \frac{\sinh(2\pi(x + y + 2))}{\sinh(8\pi)}$$

To evaluate the fault prevention strategy, we consider the ratio of the amount of data sent against those that are really needed. This ratio is less than 1.1 in the

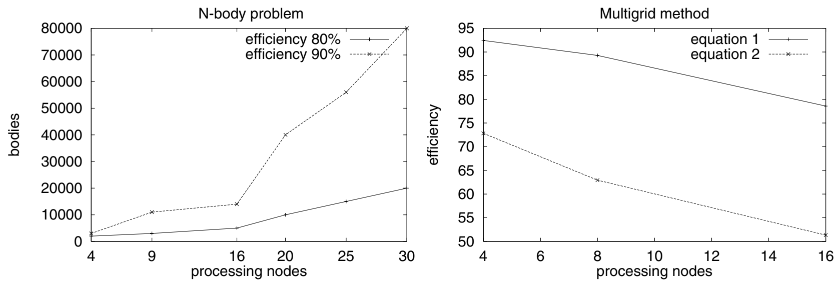

Fig. 1. Efficiency

NBP and less than 1.24 in the AMM. In the AMM, informed fault prevention reduces the ratio to 1.04.

In both problems, the balancing procedure reduces the total execution time but the optimal value of T has to be determined. In the NBP, the execution time is nearly proportional to difference between the adopted value of T and the optimal one. In the AMM, the optimal value of T also depends upon the considered equation, that determines the structure of the H-Tree. In this case, the relative difference between the execution time of a well balanced execution and that of an unbalanced one can be larger than 25%.

Fig. 1 shows the efficiency of the two implementations. For the NBP, the lowest number of bodies to achieve a given efficiency is shown. For the AMM we show the results for the two equations, for a fixed number of initial points, 16.000, and the same maximum depth of the H-Tree, 12. The larger granularity of the NBP results in a better efficiency. In fact, after each fault prevention phase, the computation is executed on the whole private H-Tree in the NBP while in AMM it is executed on one level of this tree.

References

[1] S. J. Aarset, M. Henon, and R. Wielen. Numerical methods for the study of star cluster dynamics. *Astronomy and Astrophysics*, 37(2), 1974.

[2] J.E. Barnes and P. Hut. A hierarchical O(nlogn) force calculation algorithm. *Nature*, 324, 1986.

[3] M. Berger and J. Oliger. Adaptive mesh refinement for hyperbolic partial differential equations. *J. Comp. Physics*, 53, 1984.

[4] W. Briggs. *A multigrid tutorial.* SIAM, 1987.

[5] P. Hanrahan, D. Salzman, and L. Aupperle. A rapid hierarchical radiosity algorithm. *Computer Graphics (SIGGRAPH '91 Proceedings)*, 25(4), 1991.

[6] J.R. Pilkington and S.B. Baden. Dynamic partitioning of non–uniform structured workloads with space filling curves. *IEEE TOPDS*, 7(3), 1996.

[7] J.K. Salmon. *Parallel Hierarchical N-body Methods.* PhD thesis, California Institute of Technology, 1990.

[8] J.P. Singh. *Parallel Hierarchical N-body Methods and their Implications for Multiprocessors.* PhD thesis, Stanford University, 1993.