Computer Physics Communications 185 (2014) 798–808

Contents lists available at ScienceDirect

Computer Physics Communications

journal homepage: www.elsevier.com/locate/cpc

Parallel node placement method by bubble simulation

Yufeng Nie *, Weiwei Zhang, Nan Qi, Yiqiang Li

Department of Applied Mathematics, School of Science, Northwestern Polytechnical University, Xi'an 710129, PR China

ARTICLE INFO

Article history:
Received 19 January 2013
Received in revised form
14 November 2013
Accepted 18 November 2013
Available online 22 November 2013

Keywords:
Parallel
Node placement
Load balancing
Domain decomposition
Molecular dynamics simulation

ABSTRACT

An efficient Parallel Node Placement method by Bubble Simulation (PNPBS), employing METIS-based domain decomposition (DD) for an arbitrary number of processors is introduced. In accordance with the desired nodal density and Newton's Second Law of Motion, automatic generation of node sets by bubble simulation has been demonstrated in previous work. Since the interaction force between nodes is short-range, for two distant nodes, their positions and velocities can be updated simultaneously and independently during dynamic simulation, which indicates the inherent property of parallelism, it is quite suitable for parallel computing. In this PNPBS method, the METIS-based DD scheme has been investigated for uniform and non-uniform node sets, and dynamic load balancing is obtained by evenly distributing work among the processors. For the nodes near the common interface of two neighboring subdomains, there is no need for special treatment after dynamic simulation. These nodes have good geometrical properties and a smooth density distribution which is desirable in the numerical solution of partial differential equations (PDEs). The results of numerical examples show that quasi linear speedup in the number of processors and high efficiency are achieved.

1. Introduction

With the rapid development of parallel computers and the increasing scale of the finite element computing, parallel techniques for the finite element method (FEM) have been given unprecedented attention [1–8]. At present, the parallel aspects of FEM primarily focus on each function block such as mesh generation (referred to as pre-processing) [1–5], global stiffness matrix formation and solving systems of linear equations (referred to as main-processing) [6–8]. The structural analysis of FEM cannot be done until the completion of the FEM mesh generation, inevitably, this serial characteristic seriously restricts the parallel efficiency and becomes one of the bottlenecks in large-scale parallel FEM analysis [9–12].

In recent years, much effort has been devoted to improve the parallel efficiency of the FEM analysis based on nodes. The Free Mesh Method (FMM) and the Node-based Local Finite Element Method (NLFEM) have been developed by Yagawa et al. [9] and Nie et al. [12], respectively. The FMM and the NLFEM are parallel node-based finite element methods featuring node-based local mesh generation and node-based finite element calculation. These new parallel mechanisms achieve naturally the seamless link between pre-processing and main-processing, and get rid of the original serialization process. Nevertheless, it is worth noting that the FMM and the NLFEM begin by appropriately distributing the nodes in the analysis domain [9,12,13], i.e. node coordinates and nodal density information are given as input information. However, how to generate nodes in parallel is not mentioned.

In fact, the quality of the node set has a great influence on the accuracy and convergence properties of finite element solution for node-based parallel finite element methods. Therefore, how to generate node sets properly and efficiently is attracting much research interest, and some research results have already been reported in Refs. [14–18] and references therein. Li et al. [16] construct an advancing front-based sphere packing process. Based on a centroid Voronoi structure, Ju et al. [17] use probabilistic methods to generate centroidal voronoi tessellations (CVTs) and the parallel implementations are also presented. Zhang et al. [18] propose a node placement approach using Monte Carlo simulation to minimize system potential energy, and thereby to find a near-equilibrium configuration of nodes. Shimada et al. [19] describe a scheme to pack circles by defining proximity-based interacting forces among circles and finding a force-balancing configuration using dynamic simulation. However, the existing node placement methods mentioned above are serial algorithms except for Ju's method [17]. When performing the node-based parallel finite element calculation, these serial node distribution methods are usually not beneficial to improving the parallel efficiency. For the parallel CVTs method presented by Ju et al., the method may require special handling when generating boundary nodes of the whole domain. Furthermore, the number of iterations required is usually huge, and in an iterative process, global communication is needed between processors. This kind of communication will inevitably result in idle time, influencing the whole parallel efficiency.

* Corresponding author.
E-mail address: yfnie@nwpu.edu.cn (Y. Nie).

0010-4655/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.cpc.2013.11.010

Recently, the Node Placement method by Bubble Simulation (NPBS) has been developed [20–22], and it has also been successfully applied to node-based local mesh generation [23] and anisotropic triangular meshing problems [24]. In the NPBS method, due to the short-range interaction force between bubbles, for two relatively distant bubbles, their positions and velocities can be updated simultaneously and independently during simulation, making the NPBS method suitable for parallel environments.

Furthermore, domain decomposition (DD) techniques have been employed in parallel node placement methods in order to decompose a large, complex problem into many simpler subproblems which can be solved in parallel. Within the context of parallel mesh generation, the DD method, such as the Medial Axis Domain Decomposition (MADD) [25], the coarse-grained parallel harness method [26] and the METIS-based DD technique [27,28] have been developed in recent years. Within the context of parallel molecular dynamics simulations, there are a large variety of DD methods proposed in the literature, see Refs. [29–32] and references therein.

In this paper, we focus on developing a parallel, scalable, robust node placement method by bubble simulation (PNPBS) using MPI communication for an arbitrary number of processors, and the METIS-based DD technique is used. The METIS-based DD technique [27] addresses the issue of load imbalance among processors in the process of node placement. Furthermore, the average speed of bubbles could decrease quickly during dynamic simulation [21], thus the PNPBS method requires much fewer iterations for convergence. Communication is performed every k steps among geometrically neighboring processors, such that the proportion of the communication time in the total time is reduced greatly, so parallel efficiency can be improved effectively. Furthermore, an adjacency list related to each node is also provided which stores the information of neighboring nodes, this can be directly used for node-based local mesh generation [12,23] and node-based finite element calculation [9,10] when solving partial differential equations system in parallel.

The rest of this paper is organized as follows: the NPBS method is described briefly in Section 2, and its parallel features based on the METIS DD scheme are investigated in detail in Section 3. In Section 4, the numerical results of the PNPBS method are presented. Finally, conclusions are described in Section 5.

2. Outline of node placement method by bubble simulation

In this section we give a brief overview of the node placement method by bubble simulation (NPBS) (refer to Refs. [20,21] for more details). The main steps of the NPBS method are given as follows: First, an initial node set is positioned in the domain. It is important to obtain a good initial bubble configuration for speeding up the simulation. Then nodes are considered as the centers of bubbles, and bubbles are driven by their interacting forces, until a force-balancing configuration of bubbles is obtained. Finally the centers of bubbles will form a good-quality node distribution in the domain. The main steps are discussed in the following.

2.1. Initial bubble placement

The initial distribution of the bubbles is very important to the NPBS method. If the initial bubble configuration is very poor, then a large number of iterative steps will be required before achieving a stable node configuration. In the NPBS method, initial boundary nodes are placed with the sub-binary technique, while inner nodes are placed using rhombic cells with inside angles of 60^0 and 120^0 to realize a hexagonal arrangement of bubbles. Meanwhile, the desired node density is controlled by a node-spacing function $d(x, y)$, which is user-defined, or be changed with a priori and a posterior error estimates in the adaptive numerical computation [33].

2.2. Dynamic simulation

According to Newton's second law of motion, the motion equation of each bubble is a second order ordinary differential equation

$$m\ddot{x}_i + c\dot{x}_i = f_i, \quad i = 1 \ldots N \tag{1}$$

where m is the mass of the bubble, c is the damping coefficient, N is the number of bubbles, x_i is the center of bubble i, $-c\dot{x}_i$ is a viscous damping force from the system, which makes the bubble system converge to a stable configuration, and f_i is the resultant force exerted on bubble i by its surrounding bubbles.

For two neighboring bubbles, the interaction force tries to maintain the ideal distance between them by exerting a repelling force when they are too close, or an attracting force when they are too distant. It can be approximated by [19]:

$$f(w) = \begin{cases} k_0\left(1.25w^3 - 2.375w^2 + 1.125\right) & 0 \le w \le 1.5 \\ 0 & 1.5 < w \end{cases} \tag{2}$$

where w is the ratio of the real distance and the desired distance between two bubbles. For the system of second-order differential equations (1), a numerical method using the Euler predictor–corrector formula is used, and the fourth-order Runge–Kutta method is used in the last iteration. Because of the addition of the damping force in the motion Eq. (1), the average speed of bubbles tends to a final value of zero. That is to say, the bubble system could converge to a stable configuration, as discussed in Ref. [21].

2.3. Adjacency list

In principle, all interaction forces on bubble i resulting from all other bubbles have to be taken into account when computing the resultant force. In practice, because of the short-range interaction force between bubbles, an adjacency list for the bubble i including its neighbor bubbles located within the cutoff radius $r = 1.5\sigma$ is defined, where σ is the ideal distance between bubble i and its neighboring bubble. When calculating the resultant force, only the interaction forces exerted by adjacent bubbles from the adjacency list need to be considered. Since each adjacency list has to be updated at each time step, which is very time-consuming, the radius of this adjacency list is often extended to be 1.7σ to reduce the frequency of updating, and each adjacency list is usually updated every 5–8 time steps. This greatly reduces the time consumed in searching for neighboring nodes. In the initial establishment of the adjacency list, uniform bucket and multilayer bucket search methods can be used [13].

2.4. The adjustment of the bubble population

Usually, initial bubble placement cannot generate a proper number of bubbles, and the number of bubbles is adjusted by the overlap ratio [19] during dynamic simulation

$$\alpha_i = \frac{1}{r_i}\sum_{j=0}^{N}(2r_i + r_j - l_{ij}) \tag{3}$$

where r_i and r_j are the ideal radii of bubble i and bubble j, l_{ij} is the real distance between the centers of bubble i and bubble j. In the ideal case, the standard overlap ratios of nodes on a line, on a surface and in the internal volume are 2, 6 and 12 respectively. So by computing the overlap ratio, we delete the bubbles whose overlap ratios are too large, or add new bubbles near the bubbles whose overlap ratios are too small. Finally, the population of bubbles can be controlled dynamically.

Fig. 1 shows the final bubbles and the nodal distribution after the dynamic adjustment of the bubble population. Local refinement is implemented through controlling the node spacing