Annual Report of
ADVENTURE Project
ADV-99-1 (1999)

LARGE-SCALE DYNAMIC ANALYSES WITH CONTACT-IMPACT USING THE HIERARCHICAL DOMAIN DECOMPOSITION METHOD

Atsuya Oishi
Department of Mechanical Engineering, Faculty of Engineering
The University of Tokushima
e-mail: oishi@me.tokushima-u.ac.jp

Key words: Domain Decomposition, Parallel Processing, Dynamic Problem, Contact-Impact

1 INTRODUCTION

Concomitant with the remarkable progress in the field of microelectronics, the performance of computers has increased dramatically. This has allowed the increasing replacement of time-consuming and highly expensive experimentation with such computer simulations as the finite element method (FEM). Computer simulation is particularly useful in the case of large complex problems that elude traditional experimentation, including dynamic time-dependent problems such as determinations of vehicle crashworthiness, nuclear reactor vibration in an earthquake, or ultrasonic wave propagation in solids with cracks. Computer simulations have come to play a more and more important role because of their low cost, capability for parametric evaluation and applicability to problems that cannot be tested experimentally.

As computer simulation is increasingly utilized as a tool in design and analysis, models of increasingly larger scale will need to be simulated. The parallel processing technique, which concurrently utilizes anywhere from a few to a few thousand processors, has been recognized as the key technology for dealing with large-scale simulations in a reasonable computation time. Although parallel computers of various types have been developed and tested, it has become recognized that distributed local memory parallel computers combined with an MIMD (Multiple Instruction, Multiple Data) processing model are the most promising for high performance computing.

To achieve good scalability in a distributed local memory parallel processing environment, large granularity of parallel tasks and a well-balanced workload distribution are key issues, where granularity is a measure of the amount of computation which can be performed on each processor without any data communication among processors. A number of researchers have been studying various finite element algorithms for parallel computers, most of which

take into account the node-wise, the element-wise, and the domain-wise concurrency. Among them, parallel algorithms based on the domain-wise concurrency tend to feature the largest granularity of parallel tasks. The domain-wise concurrency is found in the parallel substructure equation solvers and in some domain decomposition methods [Malone, 1988; Carter et al., 1989; Farhat and Roux, 1991; Farhat et al., 1994; Farhat and Roux, 1994; Farhat et al., 1995].

We have been studying the domain decomposition method (DDM) combined with an iterative solver as a parallel numerical algorithm for finite element analysis [Yagawa et al., 1991; Oishi et al., 1992; Yagawa et al., 1993; Yagawa and Shioya, 1993; Oishi et al., 1996]. In this method, a whole domain to be analyzed is fictitiously divided into a number of subdomains without overlapping, and the finite element analyses of these subdomains are performed in parallel under the constraint of both displacement continuity and force equivalence among subdomains, which is satisfied through iterative calculations such as the Conjugate Gradient (CG) method. This algorithm was formulated for elastostatic problems and implemented on various parallel computing environments [Yagawa et al., 1991; Yagawa et al., 1993; Yagawa and Shioya, 1993]. It was also formulated for use in elastodynamic problems [Oishi et al., 1992], such as analyses of stress wave propagation in a solid, and implemented on a workstation cluster [Oishi et al., 1996].

For the analysis of some dynamic problem, such as vehicle crashworthiness and metal forming, contact-impact phenomena must inevitably be taken into account. There have been many researches on the analysis of contact-impact phenomena with the finite element method [Zhong and Mackerle, 1994]. The use of contact-impact algorithms within finite element analyses can be divided into two phases: the contact-searching algorithm phase and the contact interface algorithm phase. First, the place where contact occurs is located by contact-searching algorithm. Second, contact forces are applied between contact surfaces according to the contact interface algorithm. Contact-searching algorithms have a primary impact on the performance in the parallel processing environment due to their intrinsic requirement for global access.

In the present paper, the above DDM-based parallel finite element algorithm for dynamic problems and its implementation in parallel processing environments are explained. Then, after reviewing contact-impact algorithms and their implementation in parallel computers, a parallel contact algorithm in the ADVENTURE project and its implementation in the DDM-based parallel explicit dynamic code are explained. Finally, the performance of this algorithm within a parallel environment is demonstrated through sample analyses of a

problem of one million dofs size.

2 FUNDAMENTAL DDM ALGORITHM FOR ANALYSES OF DYNAMIC PROBLEMS

2.1 Fundamental formulation

2.1.1 Fundamental equations and weak formulation

Let us take into consideration an elastodynamic problem concerning a domain Ω, which is fictitiously divided into two non-overlapping subdomains $\Omega^{(1)}$ and $\Omega^{(2)}$ with Γ_{12} being the inter-subdomain boundary between $\Omega^{(1)}$ and $\Omega^{(2)}$. Here, $\overline{B}_i$ is the body force applied in the domain Ω, $\overline{T}_i$ the traction force applied on the boundary Γ_σ, and $\overline{u}_i$ the prescribed displacement on the boundary Γ_u. The entire domain Ω is expressed with the following equation:

$$\Omega = \Omega^{(1)} + \Omega^{(2)} \tag{1}$$

while the entire boundary of the subdomain $\Omega^{(k)}$ is expressed as :

$$\Gamma^{(k)} = \Gamma_\sigma^{(k)} + \Gamma_u^{(k)} + \Gamma_{12} \ . \tag{2}$$

The fundamental equations of the present elastodynamic problem in an infinitesimal deformation mode are summarized as follows :

$$\varepsilon_{ij}^{(k)} = \frac{1}{2}\left(u_{i,j}^{(k)} + u_{j,i}^{(k)}\right) \quad \text{in} \quad \Omega^{(k)} \tag{3}$$

$$\tau_{ij}^{(k)} = C_{ijmn}^{(k)} \varepsilon_{mn}^{(k)} \quad \text{in} \quad \Omega^{(k)} \tag{4}$$

$$\tau_{ij,j}^{(k)} + \overline{B}_i^{(k)} = \rho \ddot{u}_i^{(k)} + c\dot{u}_i^{(k)} \quad \text{in} \quad \Omega^{(k)} \tag{5}$$

$$\tau_{ij}^{(k)} \nu_j^{(k)} - \overline{T}_i^{(k)} = 0 \quad \text{on} \quad \Gamma_\sigma^{(k)} \tag{6}$$

$$u_i^{(k)} = \overline{u}_i^{(k)} \quad \text{on} \quad \Gamma_u^{(k)} \tag{7}$$

$$\tau_{ij}^{(1)} \nu_j^{(1)} + \tau_{ij}^{(2)} \nu_j^{(2)} = 0 \quad \text{on} \quad \Gamma_{12} \tag{8}$$

$$u_i^{(1)} = u_i^{(2)} = \mu_i \quad \text{on} \quad \Gamma_{12} \ , \tag{9}$$

where u_i is the displacement vector, $\dot{u}_i$ the velocity vector, $\ddot{u}_i$ the acceleration vector, ρ

the mass density, c the damping coefficient, ε_{mn} the strain tensor, τ_{ij} the stress tensor and C_{ijmn} the coefficient tensor of the Hooke's law and ν_j the outer normal vector applied on the boundary Γ, respectively. $(\ )_{,j}$ denotes the first order derivative with respect to the coordinate x_j. The superscript (k) denote variables defined in the subdomains $\Omega^{(k)}$.

Employing the displacement-based weighted residual formulation, equations (5) and (6) are satisfied approximately, while equations (3), (4) and (7) are satisfied exactly. As for the inter-subdomain boundary conditions of equations (8) and (9), equation (9) is satisfied exactly, while equation (8) is satisfied approximately in a weak sense. The weighted residual formulation is derived as follows :

$$\sum_{k=1}^{2}\Big[\int_{\Omega^{(k)}}\tau_{ij}^{(k)}\delta\varepsilon_{ij}^{(k)}d\Omega+\int_{\Omega^{(k)}}\rho\ddot{u}_i^{(k)}\delta u_i^{(k)}d\Omega+\int_{\Omega^{(k)}}c\dot{u}_i^{(k)}\delta u_i^{(k)}d\Omega \\ -\int_{\Omega^{(k)}}\overline{B}_i^{(k)}\delta u_i^{(k)}d\Omega-\int_{\Gamma_\sigma^{(k)}}\overline{T}_i^{(k)}\delta u_i^{(k)}d\Gamma\ \Big]+\int_{\Gamma_{12}}\left(\tau_{ij}^{(1)}\nu_j^{(1)}+\tau_{ij}^{(2)}\nu_j^{(2)}\right)\delta\mu_i d\Gamma=0 \tag{10}$$

where $\delta\varepsilon_{ij}^{(k)}$, $\delta u_i^{(k)}$ and $\delta\mu_i$ are kinematically admissible sets. Equation (10) consists of usual weighted residual terms for the subdomain $\Omega^{(k)}$ and the constraint term derived from the force equilibrium and the displacement continuity between the subdomains. To solve equation (10), the Conjugate Gradient (CG) method is employed here.

2.1.2 Conjugate Gradient (CG) algorithm for DDM

A positive definite and symmetric function A is first defined as :

$$A(\mu_i)=-\left(\tau_{ij}^{(1)}\tau_j^{(1)}+\tau_{ij}^{(2)}\nu_j^{(2)}\right)\Big|_{\Gamma_{12}} \quad . \tag{11}$$

Then, the CG algorithm for solving equation (11) is summarized as follows :

Step 0 : Initialization (r = 0)

$$\mu_i^r=\mu_i^0 \qquad (\mu_i^0 : \text{constant}) \tag{12}$$

$$g_i^r=A\left(\mu_i^0\right) \tag{13}$$

$$w_i^r=g_i^0 \tag{14}$$

The r.h.s. of equation (13) is calculated using the traction forces on the inter-subdomain boundary Γ_{12}, i.e. $\tau_{ij}^{(1)}\nu_j^{(1)}$ and $\tau_{ij}^{(2)}\nu_j^{(2)}$, which are obtained by solving equations (3)-(7) with the following constraint :

$$u_i^{(1)} = u_i^{(2)} = \mu_i^0 \qquad \text{on } \Gamma_{12}. \tag{15}$$

Step 1: Calculation of $A\left(w_i^r\right)$

$$A\left(w_i^r\right) = -\left(\tau_{ij}^{(1)}\tau_j^{(1)} + \tau_{ij}^{(2)}\nu_j^{(2)}\right)\Big|_{\Gamma_{12}} , \tag{16}$$

where the traction forces $\tau_{ij}^{(1)}\tau_j^{(1)}$ and $\tau_{ij}^{(2)}\nu_j^{(2)}$ on Γ_{12} are calculated by solving the following equations :

$$\tau_{ij,j}^{(k)} = \rho\ddot{u}_i^{(k)} + c\dot{u}_i^{(k)} \qquad \text{in} \qquad \Omega^{(k)} \tag{17}$$

$$\tau_{ij}^{(k)}\nu_j^{(k)} = 0 \qquad \text{on} \qquad \Gamma_\sigma^{(k)} \tag{18}$$

$$u_i^{(k)} = 0 \qquad \text{on} \qquad \Gamma_u^{(k)} \tag{19}$$

$$u_i^{(k)} = w_i^r \qquad \text{on} \qquad \Gamma_{12} . \tag{20}$$

Step 2: Update of boundary value

$$\mu_i^{r+1} = \mu_i^r + \alpha_r w_i^r \tag{21}$$

$$g_i^{r+1} = g_i^r - \alpha_r A\left(w_i^r\right) \tag{22}$$

$$w_i^{r+1} = g_i^{r+1} + \beta_r w_i^r \qquad , \tag{23}$$

where

$$\alpha_r = \frac{w_i^r g_i^r}{w_i^r A\left(w_i^r\right)} , \quad \beta_r = \frac{g_i^{r+1} A\left(w_i^r\right)}{w_i^r A\left(w_i^r\right)} \qquad . \tag{24}$$

Step 3: Judgement of convergence

If μ_i^r has not converged yet, return to Step 1 by setting r=r+1.

Step 4: Calculation of solutions

After converged values of the displacements on the inter-subdomain boundaries are determined, equations (3)-(7) and (9) are solved subdomain by subdomain.

In Steps 1-4, the dynamic finite element calculation is performed with some direct time integration schemes. Figure 1 shows the analysis flow of the implicit time integration scheme combined with the DDM algorithm. The inner CG iteration loop can be processed in parallel.

2.2 Implementation to Parallel Processing Environment

2.2.1 Hierarchical Domain Decomposition Technique

Figure 2 illustrates the practical implementation of the present dynamic DDM on a parallel processing environment. Here, processors are hierarchically classified as either Analyzers (children) or a Controller (Parent). Inter-processor communication (message passing) occurs only between the Controller and the Analyzers. The main roles of the Controller and the Analyzers are summarized below.

The Controller prepares all the data necessary for the finite element analyses of subdomains, including displacements on the inter-subdomain boundaries, mesh data and material properties of subdomains, and then provides these data to any idling Analyzer through a network. The Controller also receives analysis results of subdomains from the Analyzers, which perform the finite element analyses of subdomains. The following three processes work simultaneously on the Controller : The main process called Domain Connectivity Controller (DCC) manages all the data of a whole domain to be analyzed and performs the renewal of the values of displacements on the inter-subdomain boundaries based on the CG method. The process called Sending Server sends the data necessary for the finite element analyses of subdomains to the Analyzers. The process called Receiving Server receives the results of the finite element analyses of subdomains from the Analyzers. The input data for the finite element analyses of subdomains are prepared by the DCC and sent to the Analyzers through the Sending Server. After any of the Analyzers finishes the finite element analysis of a subdomain, its analysis results are sent back to the Controller through the Receiving Server. As long as input data to be analyzed exist, all the Analyzers keep working without any idling.

On each of the Analyzers, the Domain Analyzer (DA) receives the data for the finite element analysis of a subdomain from the Controller, performs the finite element analysis of the subdomain, sends the analysis results to the Controller and sends a request to the Controller for the data necessary for the finite element analysis of another subdomain.

In the case in which hundreds or thousands of Analyzers are utilized, the Controller suffers from the intensely concentrated data communication with the Analyzers, and this deteriorates

scalability. In order to avoid this, a hierarchy consisting of approximately one to one hundred Analyzers (children) and a single Controller (Parent) can be extended to a hierarchy consisting of hundreds or thousands of Analyzers (children), several SubControllers (Parents) and a single Controller (Grand Parent) [Yagawa and Shioya, 1993].

As for a message passing library, the PVM (Parallel Virtual Machine) [Geist et al., 1994] is adopted.

2.2.2 Workload balancing

Rigorous workload balancing among processors is required to accomplish good scalability, i.e., high parallel efficiency. When implementing the present dynamic DDM on a parallel-processing environment, we assume that the number of subdomains is much greater than that of the Analyzers employed. This assumption becomes reasonable in the case of solving a very large-scale problem.

Owing to the hierarchical domain decomposition model, the Controller distributes the data for the finite element analysis of one subdomain to an idling Analyzer as soon as possible: a dynamic scheduling [Lewis and El-Rewini, 1992]. Appropriate workload balancing among Analyzers is then realized dynamically as well as automatically. Neither the non-homogeneity of calculation powers among processors nor load unbalance among subdomains has more than a slight influence on scalability.

3 PARALLEL CONTACT-IMPACT ALGORITHM

3.1 Review of Contact-Impact Algorithms

Among the various contact-searching algorithms proposed, the sliding interface algorithm [Goudreau and Hallquist, 1982; Hallquist et al., 1985] is the most popular, and has been chosen for implementation in the DYNA3D, an explicit three-dimensional finite element code for analyzing large deformation dynamic response of inelastic solids and structures [Hallquist, 1983]. In the sliding interface algorithm, any two surfaces that may come into contact must be specified prior to the analysis. One of the two surfaces is designated the master surface and the other the slave surface. Contact searching need only be performed between slave nodes, i.e., the nodes on the slave surface, and the master segments, i.e., the facets of the elements on the master surface. While this algorithm is simple and very efficient for some problems, it does have limitations in applicability to other problems in which contacts occur

within one single surface and it is impossible to specify which parts of the boundary are to be in contact prior to the solution of the problem. To solve this type of contact problem, a single surface algorithm has been proposed by Benson and Hallquist (1990).

A general contact searching algorithm with contact hierarchies, which is generally referred to as the hierarchy-territory algorithm, or HITA, has been proposed by Zhong and Nilsson (1989). In the HITA algorithm, a contact system is decomposed into contact surfaces, a contact surface into contact segments, a contact segment into contact edges, and a contact edge into contact nodes, and these constituent surfaces, segments, etc. form contact hierarchies. Territories are then defined for contact hierarchies to give approximate but efficient representations of the domains occupied by these contact hierarchies. Contact searching is then performed from higher level contact hierarchies to lower level ones. If the territories of two contact hierarchies do not intersect each other, no contact searching is performed between lower level hierarchies of the two hierarchies. The position code algorithm (POCA) [Oldenburg and Nilsson, 1994] is a variant of the general HITA algorithm. One important feature of this algorithm is the use of position codes for efficient sorting of contact nodes. The inside-outside algorithm [Wang and Nakamachi, 1997] adopts the POCA for global search and an improved sliding interface algorithm using area coordinates for local search.

Belytschko proposed another contact searching algorithm, the pinball contact algorithm [Belytschko and Neal, 1991]. The pinball algorithm transforms the general contacts between contact nodes and their targets into contacts between pinballs, i.e., spheres embedded in surface elements. In the pinball algorithm, the basic task for contact searching can be reduced to finding pinballs between which the distance is smaller than the sum of their radii.

Because these contact searching algorithms have been developed without taking parallel processing into account, they tend to result in poor scalability in the distributed memory MIMD parallel environments. Plaskacz (1995) reported that the use of a pinball algorithm on a workstation cluster, which consists of four workstations, resulted in a speed enhancement of only about 70% in comparison to the sequential computation using a single processor. Use of a volume-checking algorithm with tree-like inter-processor communication on a 128 processor Symult 2010 shows 70% parallel efficiency for some problems, but also shows as little as 28% parallel efficiency in worst-case scenarios [Malone and Johnson, 1994a; Malone and Johnson, 1994b]. Another contact-searching algorithm, the multi-level approach, has been proposed and implemented on distributed memory MIMD parallel computers [Elsner et al., 1996]. Use of a multi-level approach with static load balancing on an IBM SP2 resulted

in a speed enhancement of 10 times for a practical crashworthiness problem solved with 32 processors.

3.2 Parallel Contact-Impact Algorithm in the ADVENTURE project

3.2.1 Fundamental Concepts

ADVENTURE project researchers have developed a new parallel contact-impact algorithm that utilizes a sliding interface algorithm for the local search process and that is suitable for use in distributed memory parallel processing environments. In the sliding interface contact algorithm, contact search is performed between master and slave surfaces. Domain decomposition type parallelization of the sliding interface algorithm has two variants from the viewpoint of mesh partitioning. One decomposes a contact surface into several subdomains and the other does not. In the latter parallelization, subdomains that include contact surfaces may have many more dofs than subdomains that do not include contact surfaces, and this may lead to poor load balancing among processors. Moreover, special techniques are required to suppress decomposition of contact surfaces into several subdomains in the mesh partitioning process. Use of the latter parallelization is therefore restricted to cases with small contact surfaces. So decomposition of a contact surface into subdomains is essential.

In the case of decomposing a contact surface into several subdomains, the contact-surface data can be arranged in one of several ways :

(1) Whole data of a contact surface can be distributed to every processor that analyzes subdomains that include the contact surface.

(2) Only that portion of contact-surface data that is absolutely essential for contact search can be distributed to processors that analyze subdomains that include the contact surface.

(3) A contact surface can be divided into subdomains in a manner different from that of the subdivision for a finite element deformation [Hoover, et al., 1997]. In this case, additional processors other than those for the usual deformation analysis are required for contact calculation.

In the case (1), one big data set is placed at every processor that analyzes subdomains that include the contact surface. This suggests a fast, broadcast-type data communication. However, in the case of a very large contact surface or a single-surface contact, the data size to be transferred may become too large to keep the communication overhead negligible. In the case (2), the size of the contact surface data to be transferred to each processor is small, but the data varies from processor to processor. In the case (3), poor scalability may result

from not only the difficulty of achieving load balancing among processors, but also the difficulty of determining the appropriate location of contact data among processors. In consideration of the above factors, the contact surface data in this study are distributed and located among processors using the second method for reason of its flexibility for any contact surface in mesh partitioning, its scalability and its easy application to parallel processing environments.

Figure 3 illustrates the adopted contact data location in the present parallel contact algorithm using the domain decomposition method. Among domains containing a master surface, domains M1, M2 and M3 are considered to be neighbor domains for the S2 domain that contains a slave surface. Therefore, the processor that perform the analysis of the S2 domain is also provided with the contact data of the M1, M2, and M3 domains.

3.2.2 Procedures

The parallel contact algorithm suitable for use in the domain decomposition method, which is newly developed in this study, consists of the following six parts.

Part I:

Contact surfaces are numbered sequentially from 0 to N, N must be an odd number, and the k-th surface is assumed to contact with the $k+(-1)^k$ th surface. Contact can happen only between the pair of surfaces designated as master and slave. Hereafter the set of nodes on the n-th contact surface is written as S^n, and the subset of S^n which belongs to the i-th domain D_i is written as S_i^n. A territory, the smallest rectangular parallelepiped containing S_i^n, is defined as follows.

$$V_i^m = \{P(x,y,z) \mid X_{\min} < x < X_{\max}, Y_{\min} < y < Y_{\max}, Z_{\min} < z < Z_{\max}\} \quad (25)$$

Here $X_{\min}$, $Y_{\min}$, and $Z_{\min}$ are the minimum coordinates of nodes in S_i^n, and $X_{\max}$, $Y_{\max}$, and $Z_{\max}$ are the maximum coordinates. $\overline{V}_i^m$ is also defined as the expansion of V_i^m along each axis by the length d:

$$\overline{V}_i^m = \{P(x,y,z) \mid X_{\min} - d < x < X_{\max} + d, Y_{\min} - d < y < Y_{\max} + d, Z_{\min} - d < z < Z_{\max} + d\} \quad (26)$$

Part II:

For each node in S_i^n, the nearest neighbor node on the opposite surface must be found first. The search area for a node in the S_i^n is limited within $S_j^m \ (j \in J)$. Here, $m = n + (-1)^n$ and J is defined as follows.

$$J = \left\{ j \mid \overline{V}_i^n \cap \overline{V}_j^m \neq \phi \right\} \tag{27}$$

Part III:

T_i^n, a set of nodes related to the contact searching for S_i^n, is defined as follows.

$$T_i^n = S_i^n \cup \left(\bigcup_{j \in J} S_j^m \right) \tag{28}$$

Nodes in the T_i^n are partitioned into buckets, or sets of rectangular parallelepipeds, according to their coordinates, as follows:

(1) Find the range of coordinates of node T_i^n: $X_{\max}, Y_{\max}, Z_{\max}$ and $X_{\min}, Y_{\min}, Z_{\min}$.

(2) Determine the size of buckets, bs_x, bs_y, bs_z, from the size of T_i^n and the appropriate number of buckets.

(3) Determine the bucket number (bn_x, bn_y, bn_z) for every node $P(x, y, z)$ in the T_i^n. The bucket number is determined as follows:

$$\begin{aligned} bn_x &= (\text{int}) \frac{(x - X_{\min})}{bs_x} \\ bn_y &= (\text{int}) \frac{(y - Y_{\min})}{bs_y} \\ bn_z &= (\text{int}) \frac{(z - Z_{\min})}{bs_z} \end{aligned} \tag{29}$$

Here, $(\text{int})a$ means the maximum integer number not exceeding a.

Part IV:

For every node in the S_i^n, find the nearest neighbor node among $S_j^m \ (j \in J)$. For the node P in the bucket (bn_x, bn_y, bn_z), the search area is limited within 27 buckets ranging from ($bn_x - 1, bn_y - 1, bn_z - 1$) to ($bn_x + 1, bn_y + 1, bn_z + 1$).

Part V:

If P_m is found to be the nearest node for the node P_s in the S_i^n, then the segment which P_s really hits is subsequently determined. Here a segment is a facet of an element on a

contact surface. P_m usually belongs to several segments. The segment that P_s really hits can be found by the following criterion [Hallquist, 1993].

$$(\vec{c}_1 \times \vec{r}) \cdot (\vec{c}_1 \times \vec{c}_2) > 0 \tag{30}$$

$$(\vec{c}_1 \times \vec{r}) \cdot (\vec{r} \times \vec{c}_2) > 0 \tag{31}$$

Here, as shown in the Figure 4, $\vec{c}_1$ and $\vec{c}_2$ are vectors drawn from P_m to the neighboring node on the segment counterclockwise or clockwise, respectively :

$$\vec{r} = \vec{t} - (\vec{t} \cdot \vec{m}) \cdot \vec{m} \tag{32}$$

$$\vec{m} = \frac{\vec{c}_1 \times \vec{c}_2}{|\vec{c}_1 \times \vec{c}_2|} \quad , \tag{33}$$

and $\vec{t}$ is a vector drawn from P_m to P_s. The segment that meets these criteria (Eqs. (30),(31)) is picked up as the target segment.

After the target segment is specified, local coordinates (ξ, η) of the accurate contact point on the segment must be identified by solving the following set of equations using Newton's method.

$$\frac{\partial \vec{r}}{\partial \xi}(\xi_c, \eta_c) \cdot \{\vec{t} - \vec{r}(\xi_c, \eta_c)\} = 0 \tag{34}$$

$$\frac{\partial \vec{r}}{\partial \eta}(\xi_c, \eta_c) \cdot \{\vec{t} - \vec{r}(\xi_c, \eta_c)\} = 0 \tag{35}$$

Further, a penetration depth g is calculated as follows.

$$g = \vec{m} \cdot (\vec{t} - \vec{r}(\xi_c, \eta_c)) \tag{36}$$

$g > 0$ means that the node and the segment are not in contact but in proximity. If $g < 0$, contact forces for the slave node P_s and master nodes on the segment are calculated as

$$\vec{f}_s = -gk\vec{m} \tag{37}$$

$$\vec{f}_m^i = \phi_i(\xi_c, \eta_c) \cdot \vec{f}_s \quad , \tag{38}$$

where $\vec{f}_s$ is a contact force vector for the slave node P_s, $\vec{f}_m^i$ is a contact force assigned to the i-th node on the target segment, and k is a factor determined by geometry and material properties of the element including the target segment.

Part VI:
Contact forces calculated for every node on contact surfaces are assembled to define accurate contact forces. The accumulated contact force $\vec{f}_c$ consists of one $\vec{f}_s$ and several $\vec{f}_m$ from one or more domains.

$$\vec{f}_c = \vec{f}_s + \sum \vec{f}_m \tag{39}$$

3.2.3 Parallel Implementation

Part I and part III are performed subdomain by subdomain in parallel on Analyzers. Part II is performed sequentially on the Controller. Parts I through III do not have to be executed for each time step. Adopting the appropriate parameter d in part I and the appropriate bucket sizes in part III makes the recalculation of parts I to III unnecessary over approximately one to one hundred time steps. Part IV and part V are performed subdomain by subdomain in parallel on Analyzers. Part VI is performed sequentially on the Controller.

4 PARALLEL DYNAMIC CONTACT IMPACT ANALYSIS CODE

Finite element formulation of dynamic contact problems based on the penalty method is described as follows.

$$[M]\{\ddot{u}\} + [K]\{u\} = \{f\} + \{f_c\} \tag{40}$$

Here, $\vec{f}_c$ is a contact force based on the penalty method, which is calculated by equation (39).

In this study, the proposed parallel contact-impact algorithm was applied to the domain decomposition type parallel dynamic analysis code based on the explicit time integration scheme. This code inherited the following important features from the parallel dynamic analysis code developed in Chapter 2.

(1) The hierarchical domain decomposition method with an iterative solver is adopted as the fundamental algorithm for parallel processing.
(2) The number of subdomains is much greater than that of processors available for calculation.
(3) The mapping from a subdomain to a processor dynamically changes during analysis

due to a dynamic scheduling mechanism.

(4) The manner of mesh partitioning, including division of a contact surface, is not limited. Thus domain decomposition suitable for load balancing is possible.

Features (2),(3) and (4) suggest the possibility of high parallel efficiency and a scalability appropriate for extra large scale analyses.

In the developed parallel dynamic contact analysis code, however, another overhead arises due to synchronization for contact force accumulation. To cope with this overhead, the order of the subdomain to be transferred to Analyzers is controlled. Subdomains are classified into two groups, D1 and D2, with the former including and the latter not including contact surfaces. Analysis at each time step is made up of three phases as follows:

Phase1: The data for domains in D1 are transferred to Analyzers. Contact search and contact force evaluation are performed.

Phase2: The data for domains in D2 are transferred to Analyzers. Finite element deformation analyses are performed.

Phase3: The data for domains in D1 are again transferred to Analyzers. Finite element deformation analyses are performed using the accumulated contact forces.

Though Phase2 can start without waiting for the end of Phase1, Phase3 cannot start before the completion of Phase1. This may cause synchronization overhead when the number of D2 subdomains is too few to fill all Analyzers by Phase2 analyses before the completion of Phase1.

In the present code, whole time steps are also classified into three modes, NEW, NORMAL and UPDATE. These three modes are sequentially and repeatedly processed in this order. Works specific to each mode are as follows.

NEW: Construct $S_j^m \left(j \in J\right)$ for every S_i^n and reset the bucket number.

NORMAL: Only the localized contact search is performed by making use of data constructed at the last NEW mode. This mode continues for one to one hundred time steps.

UPDATE: Calculate a new $\overline{V}_i^n$ for each contact surface in a subdomain and prepare for the successive NEW mode.

Figure 5 shows a flowchart of an Analyzer in the present code.

5 SAMPLE ANALYSES

The present parallel dynamic contact analysis code was applied to the analyses of elastic wave propagation in a cubic structure model with cracks. Figure 6 shows three kinds of analysis domains subjected to vertical sinusoidal force on the center of the upper surface.

CASE1: There exists no contact surface.

CASE2: One pair of contact surfaces is located in the middle half of the analysis domain.

CASE3: Seven pairs of contact surfaces are located in the half of the analysis domain.

Parameters related to computational load are listed in Table 1. In the case of a parallel-processing environment, a PC cluster consisting of 9 PCs was used. Table 2 shows the configuration of the PC cluster. One PC is assigned to the Controller and the others are assigned to Analyzers.

Figure 7 shows the speed enhancement obtained by parallel processing. The horizontal axis represents the number of Analyzers used for calculation, and the vertical axis represents the speed enhancement by parallel processing. The ideal speed enhancement is also shown in Fig.7 for comparison. As can be seen in the figure, a greater than 5-fold speed enhancement was obtained using 8 Analyzers even in the most severe CASE3. This speed enhancement corresponds to a 68.3% parallel efficiency. This result is considered to be degraded by the poor communication speed (3.4 MB/sec) of the PC cluster tested. These results indicate that parallel efficiency will be improved in the parallel processing environment with faster inter-processor communication.

6 CONCLUSIONS

In the present study, a parallel contact algorithm for finite element analyses was developed and applied to a parallel explicit dynamic analysis code based on the hierarchical domain decomposition method (DDM). Specific features of the code can be summarized as follows:

(1) A distributed memory MIMD parallel processing system with message passing, as in the case of several commercial massively parallel computers and WS/PC clusters, is the target environment.

(2) The hierarchical domain decomposition method is adopted as the fundamental algorithm for parallel processing.

(3) The number of subdomains is much greater than the number of processors available for calculation.

(4) The mapping from a subdomain to a processor dynamically changes during analysis due to a dynamic scheduling mechanism.

(5) Contact surfaces can be partitioned into subdomains.

The present code showed an approximately 70 % parallel efficiency for 8 CPUs on a PC cluster through sample analyses of a problem of one million dofs size.

REFERENCES

Belytschko, T. and Neal, M.O. (1991), Contact-impact by the pinball algorithm with penalty and Lagrangian methods, *International Journal for Numerical Methods in Engineering*, 31, pp.547-572.

Benson, D.J. and Hallquist, J.O. (1990), A single surface contact algorithm for the post-buckling analysis of shell structures, *Computer Methods in Applied Mechanics and Engineering*, 78, pp.141-163.

Carter, Jr, W.T., Sham, T.-L. and Law, K.H. (1989), A parallel finite element method and its prototype implementation on a hypercube, *Computers & Structures*, 31, pp.921-934.

Elsner, B., Galbas, H.-G., Gorg, B., Kolp, O. and Lonsdale, G. (1996), A parallel multilevel contact search algorithm in crashworthiness simulation, *Advances in Computational Structures Technology (Selected papers from the Third International Conference on Computational Structure Technology, Budapest, 21-23 August, 1996)*, Civil-Comp Press.

Farhat, C., Chen, P.-S. and Mandel, J. (1995), A scalable Lagrange multiplier based domain decomposition method for time-dependent problems, *International Journal for Numerical Methods in Engineering*, 38, pp.3831-3853.

Farhat, C., Crivelli, L. and Roux, F.-X. (1994), A transient FETI methodology for large-scale parallel implicit computations in structural mechanics, *International Journal for Numerical Methods in Engineering*, 37, pp.1945-1975.

Farhat, C. and Roux, F.-X. (1991), A method of finite element tearing and interconnecting and its parallel solution algorithm, *International Journal for Numerical Methods in Engineering*, 32, pp.1205-1227.

Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R. and Sunderam, V. (1994), *PVM : Parallel Virtual Machine*, The MIT Press.

Goudreau, G.L. and Hallquist, J.O. (1982), Recent developments in large-scale finite element Lagrangian hydrocode technology, *Computer Methods in Applied Mechanics and Engineering*, 33, pp.725-757.

Hallquist, J.O. (1983), *Theoretical Manual for DYNA3D*, UCID-19401, Lawrence Livermore National Laboratory.

Hallquist, J.O. (1993), *LS-DYNA3D Theoretical Manual*, Livermore Software Technology Corporation.

Hallquist, J.O., Goudreau, G.L. and Benson, D.J. (1985), Sliding interfaces with contact-impact in large-scale Lagrangian computation, *Computer Methods in Applied Mechanics and Engineering*, 51, pp.107-137.

Hoover, C.G., Badders, D.C., De Groot, A.J. and Sherwood, R.J. (1997), Parallel algorithm research for solid mechanics applications using finite element analysis, Thrust Area Report, UCRL-ID-125471, Lawrence Livermore National Laboratory.

Lewis, T.G. and El-Rewini, H. (1992), *Introduction to Parallel Computing*, Prentice-Hall.

Malone, J.G. (1988), Automated mesh decomposition and concurrent finite element analysis for hypercube multiprocessor computers, *Computer Methods in Applied Mechanics and Engineering*, 70, pp.27-58.

Malone, J.G. and Johnson, N.L. (1994a), A parallel finite element contact/impact algorithm for non-linear explicit transient analysis : Part I - The search algorithm and contact mechanics, *International Journal for Numerical Methods in Engineering*, 37, pp.559-590.

Malone, J.G. and Johnson, N.L. (1994b), A parallel finite element contact/impact algorithm for non-linear explicit transient analysis : Part II - Parallel implementation, *International Journal for Numerical Methods in Engineering*, 37, pp.591-603.

Oishi, A., Yamada, K., Yoshimura, S. and Yagawa, G. (1992), An application of domain decomposition method to dynamic FEM, *Transactions of the Japan Society of Mechanical Engineers*, 58A, pp.1445-1452, (in Japanese).

Oishi, A., Yamada, K., Yoshimura, S. and Yagawa, G. (1996), A parallel finite-element analysis of dynamic problems using an EWS network, *Transactions of the Japan Society of Mechanical Engineers*, 62A, pp.253-260, (in Japanese).

Oldenburg, M. and Nilsson, L. (1994), The position code algorithm for contact searching, *International Journal for Numerical Methods in Engineering*, 37, pp.359-386.

Plaskacz, E.J. (1995), On impact-contact algorithms for parallel distributed-memory computers, *Computational Mechanics '95 (Proceedings of the International Conference on Computational Engineering Science, July 30 – August 3, Hawaii, USA)*, pp.369-374.

Wang, S. P. and Nakamachi, E. (1997), The inside-outside contact search algorithm for finite element analysis, *International Journal for Numerical Methods in Engineering*, 40, pp.3665-3685.

Yagawa, G., Soneda, N. and Yoshimura, S. (1991), A large scale finite element analysis using domain decomposition method on a parallel computer, *Computers & Structures*, 38, pp.615-

625.

Yagawa, G. and Shioya, R. (1993), Parallel finite elements on a massively parallel computer with domain decomposition, *Computing Systems in Engineering*, 4, pp.495-503.

Yagawa, G., Yoshioka, A., Yoshimura, S. and Soneda, N. (1993), A parallel finite element method with a supercomputer network, *Computers & Structures*, 47, pp.407-418.

Zhong, Z.-H. and Mackerle, J. (1994), Contact-impact problems : A review with bibliography, *Applied Mechanics Review*, 47, pp.55-76.

Zhong, Z.-H. and Nilsson, L. (1989), A contact searching algorithm for general contact problems, *Computers & Structures*, 33, pp.197-209.

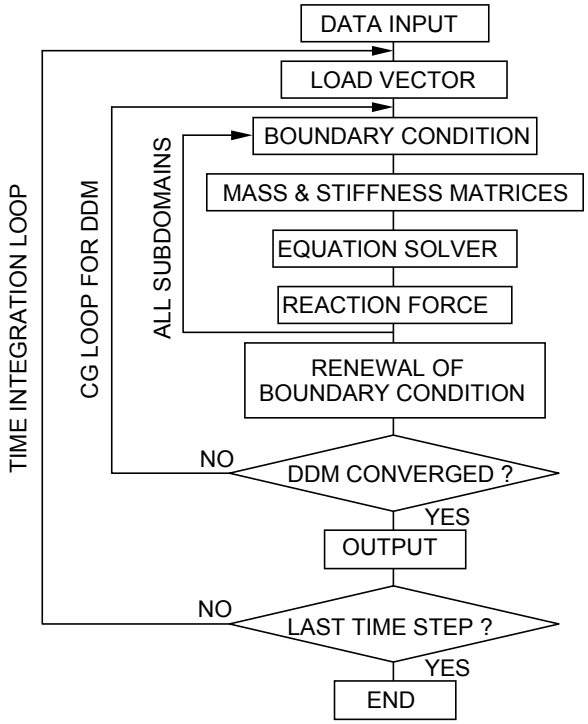

Figure 1 Flowchart of the dynamic domain decomposition method

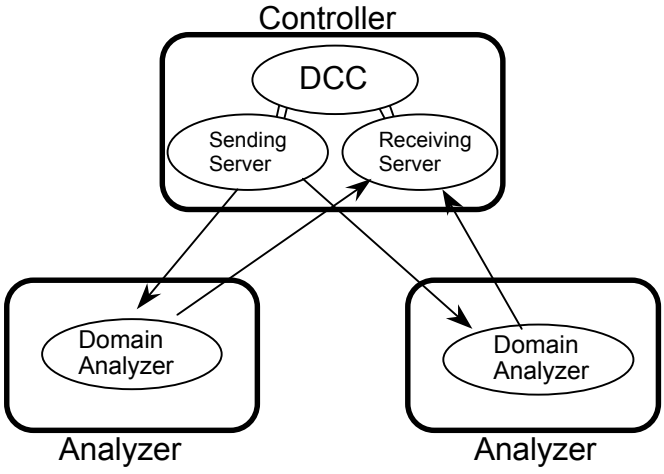

Figure 2 Schematic view of the hierarchical domain decomposition method combined with dynamic workload scheduling

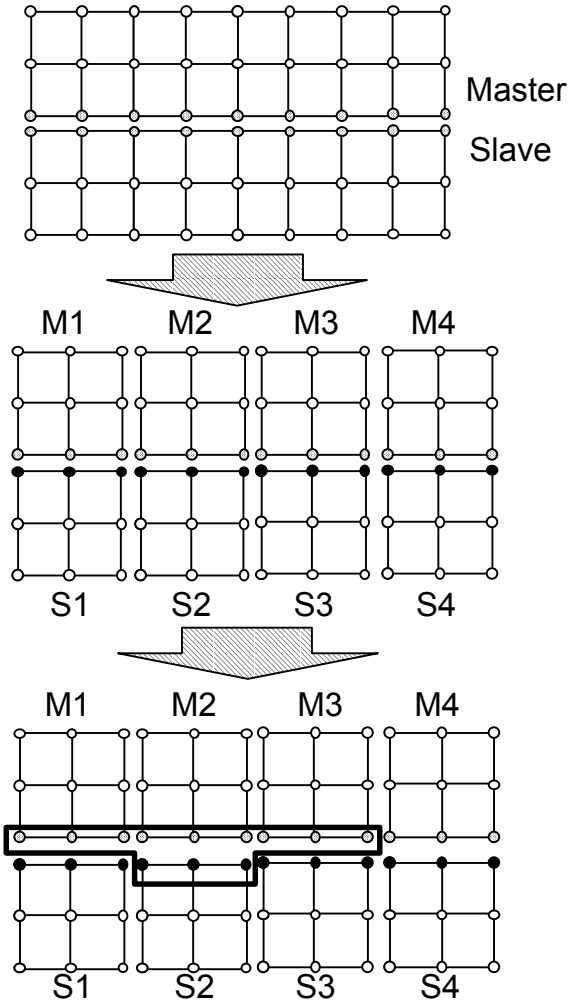

Figure 3 Decomposition and distribution of a sliding interface

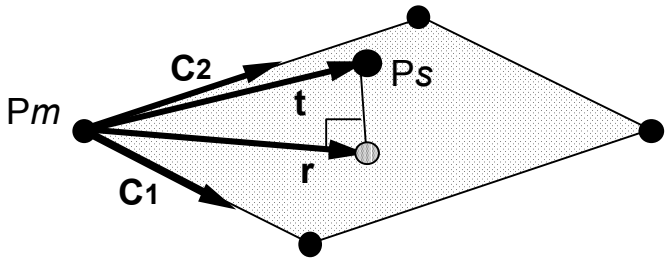

Figure 4 Schematic diagram to determine local coordinates of a contact point

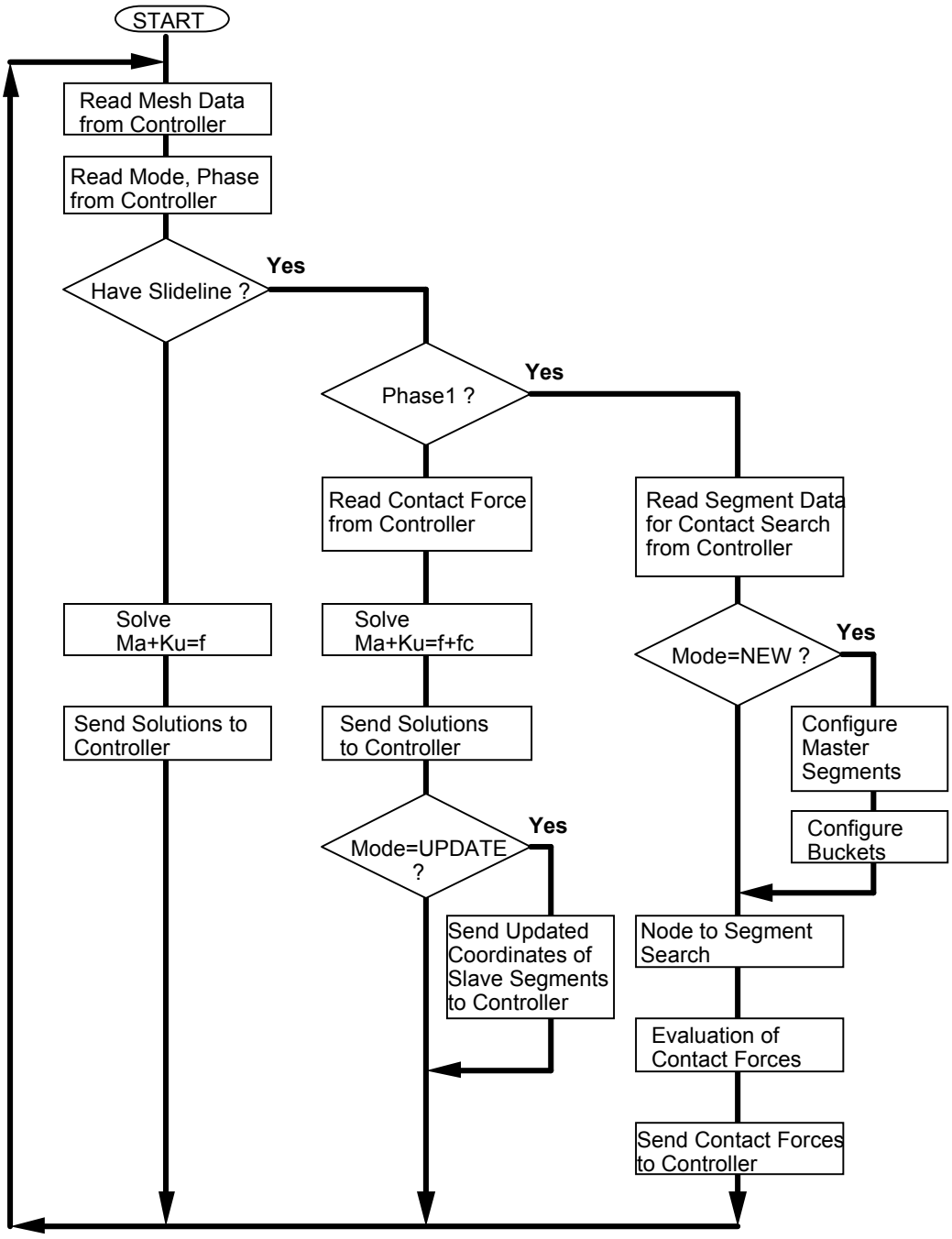

Figure 5 Flowchart of Analyzer

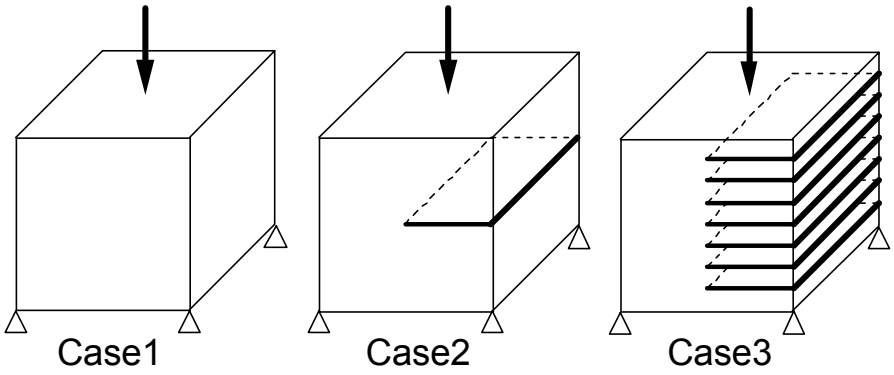

Figure 6 Three kinds of configurations of analysis domain

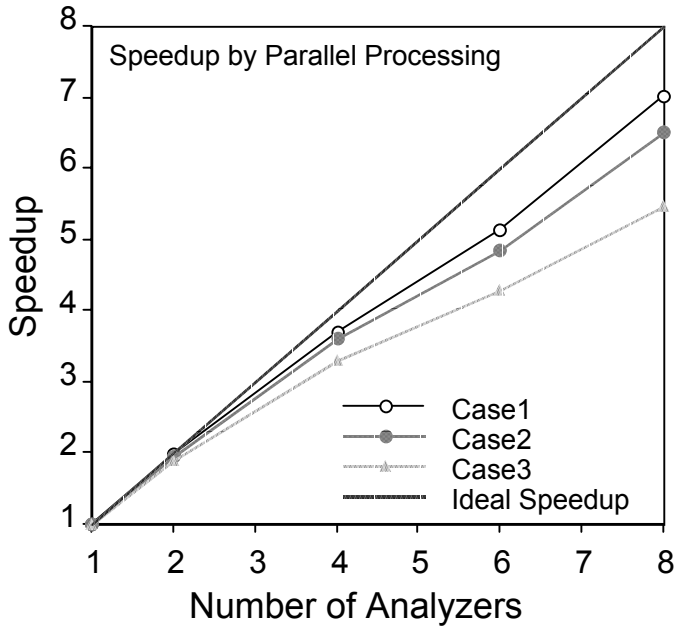

Figure 7 Speedup by parallel processing

Table 1 Parameters of three models

	CASE1	CASE2	CASE3
Number of total elements	373248	373248	373248
Number of total nodes	389017	391645	407413
Number of subdomains	512	512	512
Number of elements of one subdomain	729	729	729
Number of sliding interfaces	0	1	7
Number of subdomains including sliding interfaces	0	64	256
Number of total nodes on sliding interfaces	0	5402	37814

Table 2 Configuration of the PC cluster

	Controller	Analyzer 1-8
CPU	PentiumII-300	PentiumII-350
Memory	512MB	256MB
HDD	4GB	6GB
OS	FreeBSD-2.2.6-RELEASE	
Message-passing	pvm3.3.11	
Network	100BASE-TX Theory: 100Mbits/sec Measured: 27Mbits/sec over PVM	