Transactions of the 15th International Conference on
Structural Mechanics in Reactor Technology (SMiRT-15)
Seoul, Korea, August 15-20, 1999

Large Scale Analysis of Pressure Vessel Problem Using Iterative Domain Decomposition Method

Ryuji Shioya[1)], Tomoshi Miyamura[2)] and Genki Yagawa[2)]

1) Kyushu University, Japan
2) University of Tokyo, Japan

ABSTRACT This paper describes the parallel finite elements for a MIMD type massively parallel computer. In order to solve the issue of memory shortage and calculation time, the developed system employs a dynamic load balancing and hierarchical distributed data management technique. The present system is successfully applied to large FEM analyses of over ten million degrees of freedom with high parallel performance over 90% on a massively parallel computer, SR2201 consisting of 1,024 processing units.

1 INTRODUCTION

With the increase of the size and complexity of numerical simulation problems when using the finite element method (FEM), more processing power and memory of computer are required. Using single processor computers, we encounter their physical limits. To save calculation time and memory, it is well-known that the parallel computers, particularly Multiple Instruction Multiple Data (MIMD) type computers including clustered workstation computers seem promising. A MIMD type computer has many processors with local memory, and can reduce calculation time by distributing tasks among processors. However, we need special algorithms for parallel computing to solve problems with high performance using this kind of computer.

In order to achieve a high performance of parallel algorithms with a MIMD type computer, a large granularity of tasks and a well-balanced workload distribution are key issues, where the granularity means a measure of the amount of computation which can be performed on each processor without any data communication among processors. A variety of parallel computing algorithms for large scale finite element analyses have been studied by several researchers, most of which take into account the node- or element-wise, the column-wise, the domain-wise concurrency, or their combination [1, 2, 3]. However, these are known result in small granularity in general.

As a parallel numerical algorithm for the finite element analysis, the present authors have proposed the domain decomposition method (DDM) combined with an iterative solver [4, 5, 6, 7]. 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 each subdomain 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.

In the present study, the above method was implemented on massively parallel computer SR2201 consisting of 1,024 processing units applied to large FEM analyses of a pressure vessel with nozzles model with high parallel performance over 90%.

2 DOMAIN DECOMPOSITION METHOD (DDM)

A variety of parallel FEM algorithms for a large scale problem have been studied by several researchers, most of which take into account the node- or the element-wise, the column-wise, the domain-wise [2] concurrency, or their combination. The domain-wise concurrency is found in the parallel substructure equation solvers [2, 1, 8, 9, 3] or in some domain decomposition methods [3, 5, 6, 7, 10, 11, 12, 4, 13, 14]. It is well known that the parallel algorithm with the domain-wise concurrency has, in general, a large granularity of parallel tasks.

The Domain Decomposition Method, on which the present study is based, is originated in the well-known Schwarz method for solving elliptic problems. Although the original method is substantially of a sequential type, Glowinski et al. extended it to a parallel algorithm [10]. Their formulation, however, involves fully Neumann type calculations, which often generate floating domains that do not have enough prescribed displacements to eliminate the local rigid body modes.

In this section, a domain decomposition formulation based on a displacement-based weighted residual method [5, 6, 7], which can avoid the fully Neumann type calculation, is described.

2.1 Basic Equations Using Lagrange Multipliers

The present DDM is summarized in the following. To explain its theory, let us consider an elastic problem concerning a domain Ω, as shown in Figure 1. Here, $\overline{\mathbf{T}}$ is the traction force applied on the boundary $\Gamma_{\mathbf{T}}$, $\overline{\mathbf{B}}$ the body force applied in the domain Ω, and $\overline{\mathbf{u}}$ the prescribed displacement on the boundary $\Gamma_{\mathbf{u}}$.

Fundamental equations of this elastic problem in an infinitesimal displacement mode are given by:

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

$$\sigma_{ij} = C_{ijmn}\varepsilon_{mn} \quad \text{in } \Omega \tag{2}$$

$$\sigma_{ij,j} + \overline{B}_i = 0 \quad \text{in } \Omega \tag{3}$$

$$\sigma_{ij}\nu_j - \overline{T}_i = 0 \quad \text{on } \Gamma_{\mathbf{T}} \tag{4}$$

$$u_i = \overline{u}_i \quad \text{on } \Gamma_{\mathbf{u}} \tag{5}$$

where i,j take the value 1 to 3, u_i is the displacement vector, ε_{ij} strain tensor, σ_{ij} stress tensor, C_{ijmn} coefficient tensor of the Hooke's law and ν_j the outer normal vector on the boundary Γ. $()_{,j}$ denotes the first order derivative with respect to coordinate x_j.

The above variational form is equivalent to the following minimization problem which finds the displacement function u, a stationary point of the energy functional:

$$J(v) = \frac{1}{2}\int_\Omega \sigma_{ij}\varepsilon_{ij}d\Omega - \int_\Omega \overline{B}_i v_i d\Omega - \int_\Gamma \overline{T}_i v_i d\Gamma \tag{6}$$

As shown in Figure 2, after dividing domain Ω into N_d subdomains, $(\Omega^{(d)})_{1\le d\le N_d}$ with γ_{pq} being the interface between $\Omega^{(p)}$ and $\Omega^{(q)}$, solving the above problem becomes equivalent to finding the displacement functions $u^{(d)}$, which represents a stationary point of the energy functional:

$$J'(v^{(1)},\ldots,v^{(N_d)}) = J^{(1)}(v^{(1)}) + J^{(2)}(v^{(2)}) + \cdots + J^{(N_d)}(v^{(N_d)}) \tag{7}$$

with additional conditions on the interface boundary γ_{pq}:

$$u^{(p)} = u^{(q)} \qquad \text{on } \gamma_{pq} \tag{8}$$

$$\sigma_{ij}^{(p)}\nu_j^{(p)} + \sigma_{ij}^{(q)}\nu_i^{(q)} = 0 \quad \text{on } \gamma_{pq} \tag{9}$$

where the superscript $()^{(d)}$ designates a variable defined in subdomain $\Omega^{(d)}$.

Depending on the treatment of additional interface boundary conditions of equations (8) and (9), two approaches are available. In the first approach, equation (8) is satisfied exactly, while equation (9) is approximately satisfied, and vice versa in the second approach.

Although both the approaches are valid in principle, the first formulation involves fully Neumann type calculations, which often generate floating domains that do not have enough prescribed displacements to eliminate the local rigid body modes. The second approach is thus thought to be more appropriate.

With the use of Lagrange multiplier method, the equation (7) with the subsidiary condition (9) can be equivalently replaced by finding the saddle-point of the Lagrangian functional:

$$\mathcal{L}(v^{(1)},\ldots,v^{(N_d)},\mu^{(1)},\ldots,\mu^{(N_i)}) = \sum_{d}^{N_d} J^{(d)}(v^{(d)}) + \sum_{p,q}^{N_d}\int_{\gamma_{pq}} \mu^T C(v^{(p)},v^{(q)})d\gamma \tag{10}$$

where

$$C(v^{(p)},v^{(q)}) = \sigma_{ij}^{(p)}\nu_j^{(p)} + \sigma_{ij}^{(q)}\nu_i^{(q)} \tag{11}$$

and N_i is the total d.o.f. on interface γ_{pq}. The above problem can be equivalently converted to finding the displacement functions $u^{(d)}$ and the interface Lagrange multipliers $\lambda^{(i)}$ that satisfy:

$$\begin{aligned} \mathcal{L}(u^{(1)},\ldots,u^{(N_d)},\mu^{(1)},\ldots,\mu^{(N_i)}) &\le \mathcal{L}(u^{(1)},\ldots,u^{(N_d)},\lambda^{(1)},\ldots,\lambda^{(N_i)}) \\ &\le \mathcal{L}(v^{(1)},\ldots,v^{(N_d)},\lambda^{(1)},\ldots,\lambda^{(N_i)}) \end{aligned} \tag{12}$$

for any admissible $(v^{(d)})_{1\le d\le N_d}$ and $(\mu^{(i)})_{1\le i\le N_i}$.

We can solve this saddle-point problem (12) by a saddle-point solver such as Uzawa's algorithm or its CG variants. Let us describe the CG method to solve the equation (12) in next section.

2.2 Conjugate Gradient Algorithm for DDM

Defining the positive definite and symmetric operator $\mathcal{A}$:

$$\mathcal{A}\mu^{(i)} = C(u^{(p)}(\mu),u^{(q)}(\mu)) \tag{13}$$

where

$$u^{(p)}(\mu) = u^{(q)}(\mu) = \mu^{(i)} \text{ on } \gamma_{pq} \tag{14}$$

the CG algorithm for solving equation (12) is summarized as follows:

Step 0: Initialization

$$\mu^{(i)^0} \quad : \quad \text{arbitrarily given} \tag{15}$$

$$g^{(i)^0} = \mathcal{A}\mu^{(i)^0} \tag{16}$$

$$w^{(i)^0} = g^{(i)^0} \tag{17}$$

The $g^{(i)^0}$ of equation (16) is obtained from the traction forces on γ_{pq} which are calculated by solving equations (1)-(5) in each subdomains with the constraint:

$$u^{(p)} = u^{(q)} = \mu^{(i)^0} \text{ on } \gamma_{pq} \tag{18}$$

Step 1: Steepest descent

$$\mu^{(i)^{n+1}} = \mu^{(i)^n} - \rho^n w^{(i)^n} \tag{19}$$

where

$$\rho^n = \frac{\sum_i^{N_i} g^{(i)^n} g^{(i)^n}}{\sum_i^{N_i} w^{(i)^n} \mathcal{A} w^{(i)^n}} \tag{20}$$

Step 2: Calculation of the new descent direction

$$g^{(i)^{n+1}} = g^{(i)^n} - \rho^n \mathcal{A} w^{(i)^n} \tag{21}$$

$$w^{(i)^{n+1}} = g^{(i)^{n+1}} + \kappa^n w^{(i)^n} \tag{22}$$

where

$$\kappa = \frac{\sum_i^{N_i} g^{(i)^{n+1}} g^{(i)^{n+1}}}{\sum_i^{N_i} g^{(i)^n} g^{(i)^n}} \tag{23}$$

The $\mathcal{A}w^{(i)^n}$ of equations (20) and (21) is obtained from the traction forces on γ_{pq} which are calculated by solving the equations:

$$\sigma_{ij,j}{}^{(d)} = 0 \quad \text{in } \Omega^{(d)} \tag{24}$$

$$\sigma_{ij}{}^{(d)} \nu_j{}^{(d)} = 0 \quad \text{on } \Gamma_{\mathbf{T}}{}^{(d)} \tag{25}$$

$$u_i{}^{(d)} = 0 \quad \text{on } \Gamma_{\mathbf{u}}{}^{(d)} \tag{26}$$

$$u_i{}^{(d)} = w^n \quad \text{on } \gamma_{pq} \tag{27}$$

Step 3: Judgment of convergence

If $\mu^{(i)^n}$ has not converged yet, return to Step 1 by setting n to be $n+1$. Here the convergence criterion is defined as:

$$\frac{\max_i |g^{(i)^n}|}{\max_i |g^{(i)^0}|} < Err \tag{28}$$

in which the maximum component of force imbalance along the interface boundary, i.e. residual value, is monitored.

The flow chart of the present DDM algorithm is illustrated in Figure 3. It should be noted here that the finite element analysis of each subdomain can be performed without any data communication among subdomains. Namely, the finite element analyses of subdomains can be performed in parallel, once the displacement values on the inter-subdomain boundaries are given. Since the workload for each finite element calculation is much larger than those of other tasks including data communication and modification of boundary values, the so-called overhead due to parallel calculation is estimated to be very small. In addition, owing to the decomposition of a large scale finite element system into a number of smaller sub-systems, only small computation storage is needed for each finite element calculation.

3 PARALLEL IMPLEMENTATION

While a large granularity of tasks can be obtained through the DDM, a technique to balance workload well among processors is demanded to achieve high performance of parallel algorithms. This means that a technique to allocate subdomain data to processors suitably is needed. With a SIMD(Single Instruction-stream Multiple Data-stream) type of MPP, which can perform the same program on all processors, we must divide a domain of concern and provide uniformly distributed data to all processors in a regular fashion, although it is not easy for a complicated problem. On the other hand, MIMD(Multiple Instruction-stream Multiple Data-stream) type of MPP, which can perform different programs on each processors, enables dynamic data allocation as not only data but also instructions are distributed among processors.

This fact thus motivated us to use the MIMD type of MPP and in addition, to avoid limitation from the memory size of each processor for a large size problem, the hierarchical data management technique is developed and implemented on MPP.

3.1 Dynamic Data Allocation

In each sub-structuring or domain decomposition method, a physical problem, i.e. a domain to be analyzed, is fictitiously divided into a number of subdomains. There are two approaches in allocating parallel tasks, i.e. calculations of subdomains, on multiple processors. One approach is the so-called "Static Workload Balancing" and the other is the "Dynamic Workload Balancing", which is used in the developed system.

In static workload balancing, task allocation is performed *a priori* considering possible workload balance among processors. For simplicity, let us consider a problem which is divided into nine subdomains and allocated to nine processors as shown in Figure 4. The simplest idea is to allocate one subdomain data to each processor when the numbers of subdomains and processors coincide. With this method, when all subdomain's data are equal in size and the abilities of all the processors are equivalent, such as MPP, all processors can iteratively perform the analysis of each subdomain at the same. In such a case, as shown in Figure 5 wherein the vertical and horizontal axes denote the number of processors and time, respectively, well workload balancing can be obtained easily. When each size of subdomains are irregular (Figure 6) or processor's abilities are different, workload balance cannot be obtained similarly and there exists imbalance as shown in

Figure 7. When the calculation time of each subdomain can be estimated prior to the calculation, task allocation can be performed beforehand statically, but usually it is not easy to estimate accurately and sometime it is changed during calculation.

On the other hand, in dynamic workload balancing, task allocation among processors is performed dynamically during calculation. As data of another subdomain is provided as soon as the analysis of one subdomain is finished, an efficient workload is obtained if the whole domain is decomposed into a large number of subdomains in comparison to the number of processors. In Figure 8, twenty domains are analyzed with nine processors where each processor deals with two or three subdomains.

3.2 Roles of Processors

To implement the DDM which incorporates the dynamic workload balancing on the MIMD type of MPP, some processors must be used to allocate the data of subdomains to the other processors. That is, we need manager processors and analyzer processors. With the MIMD of MPP, such tasks can be coordinated among processors.

Up to now two types of management system, 'Father-Child Model' where each processor consists of either a manager and analyzers set or one chief manager and 'Grand-Father-Child Model' which has several manager processors, have been proposed by the authors. The latter is required for a large scale model which can not be managed by only one manager processor.

In more detail, Figure 9 shows the schematic data flow among processors of the 'Father-Child Model' system wherein the role of the 'Father' is to manage data and that of the 'Child' is to execute finite element analysis of each subdomain.

Figures 10 and 11 show the flowchart of 'Father' and 'Child' processors, respectively. 'Father' first reads mesh data which are previously divided into some subdomains and prepares initial values on the interface boundaries. 'Father' then provides subdomain's data to any idling 'Child'. The 'Child' which receives subdomain's data from a 'Father' executes the finite element analysis of the subdomain and, after the analysis, sends its result to 'Father'. Based on the result, 'Father' finally adjusts the values on the interface boundaries so as to hold the balance among subdomains. These operations are iterated until the convergence is achieved.

This system may come across a limitation in analysis due to the lack of memory of Father processor. To handle this problem, in 'Grand-Father-Child Model' system, one processor is set as a chief manager termed 'Grand', some 'Father' processors and a number of 'Child' processors. The actual role of 'Grand' is to manage values on the interface boundaries and status of 'Father' processors. Meanwhile 'Father' stores and manages data while 'Child' executes the finite element analysis of each subdomain.

Figure 12 shows the schematic data flow among processors of the present system. 'Grand' first prepares initial values on the interface boundaries. Each 'Father' then reads a part of mesh data equally divided and each data part is divided into some subdomains. 'Father' provides subdomain data to any idling 'Child'. The 'Child' which receives subdomain's data from a 'Father' executes the finite element analysis of the subdomain and, after the analysis, sends its result to the 'Father'.

Table 1: Mesh sizes for cubic model

Model	Elements	Nodes	Total DOFs	Subdomains	Parts
Vessel	228,513	359,213	1,077,639,	1,800	6
Cubic	3,375,000	3,442,951	10,328,853	15,625	5

Table 2: Calculation times of 2 CG iterations and parallel performances

PEs	Time for 2CG(sec)	Efficiency
64	636	94.2
128	293	97.6
256	144	98.0
512	76	94.0

4 LARGE SCALE ANALYSIS

The present system was applied to the finite element elasticity analysis of a pressure vessel with nozzles model subject to pressure as shown in Figure 13 and a cubic structure subject to uniform tension in these z-direction as a larger model.

The vessel model was expressed by 10-noded tetrahedral elements and The cubic model was isoparametric 8-noded elements. These sizes are shown in Table 1. Each model was divided into subdomains with parameters listed in Table 1 and divided again into parts to be read by Father processors.

These problems were solved by the commercial MPP, Hitachi SR2201 with 1,024 processors and to communicate between processors, PVM [15] library was utilized.

For cubic model, several number of processors, from 64 to 512, were used and these calculation time of 2 CG iterations and parallel performances are shown in Table 2. As shown in the table, high parallel performances over 90 % were achieved for all the cases.

For vessel model, the force imbalance measure at inter subdomain measure (residual value) against the number of CG iterations with and without diagonal preconditioning technique are shown in Figure 14. It is shown here that the residual values decrease almost monotonically with the increase of the number of iterations.

5 CONCLUSIONS

The parallel finite element system based on the DDM, which works on massively parallel computers were developed in the present study. This system can be applied to structural analyses of over ten million d.o.f. and a parallel performance of over 90% with 1,024 CPUs was obtained.

References

[1] O.O.Storaasli and P.Bergan, A nonlinear substructuring method for concurrent processing computers, *AIAA Journal* **25** (1987) 871-876.

[2] C.Farhat and L.Crivelli, A general approach to nonlinear FE computations on shared-memory multiprocessors, *Computer Methods in Applied Mechanics and Engineering* **72** (1989) 153-171.

[3] I.S.Doltsinis and S.Noelting, Studies on parallel processing for coupled field problems, *Computer Methods in Applied Mechanics and Engineering* **89** (1991) 497-521.

[4] G.Yagawa, Parallel techniques for computational mechanics, *Theoretical and Applied Mechanics* **39** (1990) 3-9.

[5] G.Yagawa, N.Soneda and S.Yoshimura, A large scale finite element analysis using domain decomposition method on parallel computer, *Computers and Structures* **38** (1991) 615-625.

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

[7] G.Yagawa and R.Shioya, Parallel finite elements on a massively parallel computer with domain decomposition, *Computing systems in Engineering* **4(4-6)** (1994) 495-503.

[8] C.Farhat, E.Wilson and G.Powell, Solution of finite element systems on concurrent processing computers, *Engineering with Computers* **2** (1987) 157-165.

[9] J.Hajjar and J.Abel, On the accuracy of some domain-by-domain algorithms for parallel processing of transient structural dynamics, *International Journal for Numerical Methods in Engineering* **28** (1989) 1855-1874.

[10] R.Glowinski, O.V.Dinh and J.Periaux, Domain decomposition methods for nonlinear problems in fluid dynamics, *Computer Methods in Applied Mechanics and Engineering* **40** (1983) 27-109.

[11] C.T.Sun and K.M.Mao, A global-local finite element method suitable for parallel computations, *Computers and Structures* **29** (1988) 309-315.

[12] A.K.Noor and J.M.Peters, A partitioning strategy for efficient nonlinear finite element dynamic analysis on multiprocessor computers, *Computers and Structures* **31** (1989) 795-810.

[13] J.C.Luo and M.B.Friedman, Implicit decomposition as a tool for solving large-scale structural systems in a parallel environment, *Computers and Structures* **35** (1990) 215-220.

[14] Y.Zhang and R.S.Harichandran, Implicit subdomain integration for dynamic analysis of large-scale structural systems, *Computer Methods in Applied Mechanics and Engineering* **81** (1990) 57-70.

[15] A.Beguelin, J.Dongarra, A. Geist, R. Mancheck and V.Sunderam, A user's guide to PVM : Parallel virtual machine, *Technical Report, Mathematical Sciences Section, Oak Ridge National Laboratory* **ORNL/TM-11826** (1991).

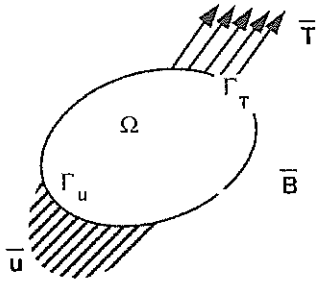

Figure 1: Analysis domain

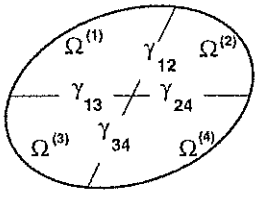

Figure 2: Analysis domain split into subdomains

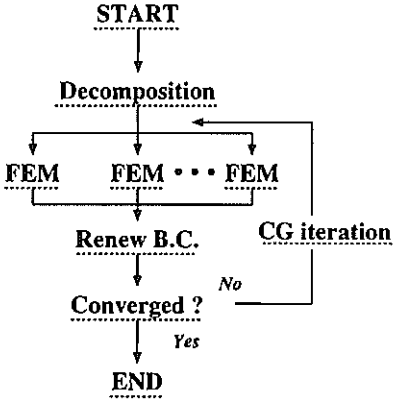

Figure 3: Flow chart of domain decomposition method

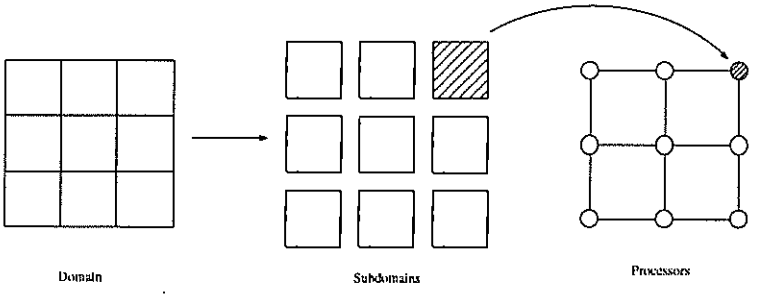

Figure 4: Static data allocation of regular subdomains

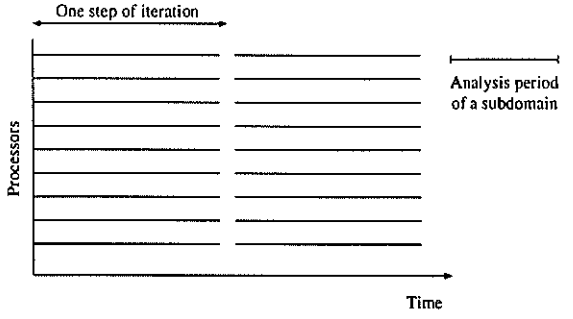

Figure 5: Regular workload balancing among processors

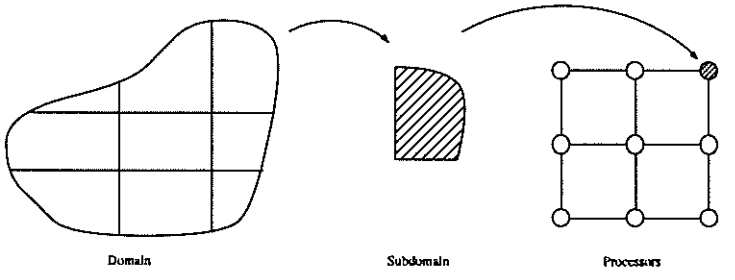

Figure 6: Static data allocation of irregular subdomains

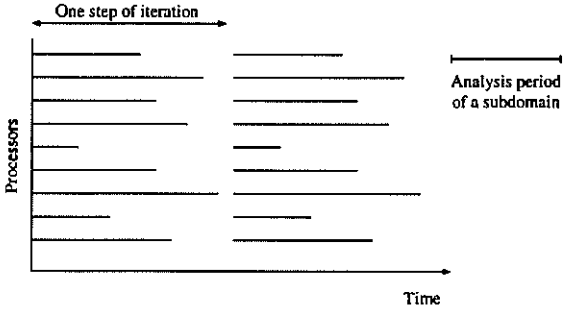

Figure 7: Irregular workload balancing among processors

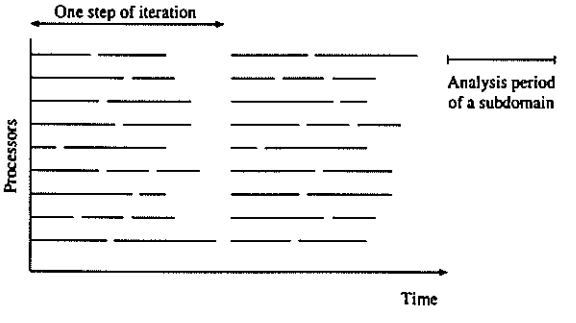

Figure 8: Dynamic workload balancing among processors

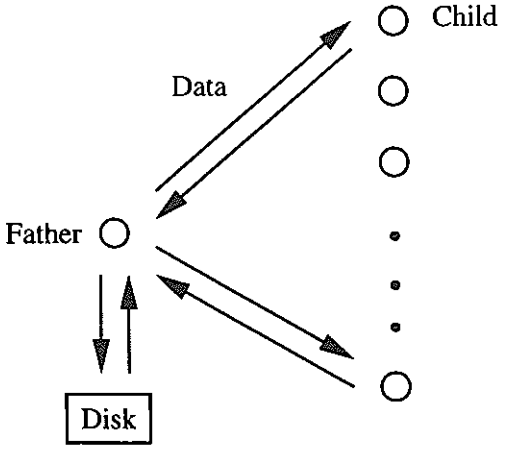

Figure 9: Schematic data flow among Father-Child processors

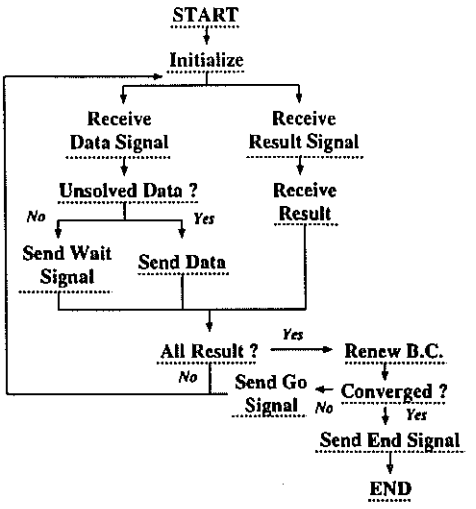

Figure 10: Flow chart of Father processor

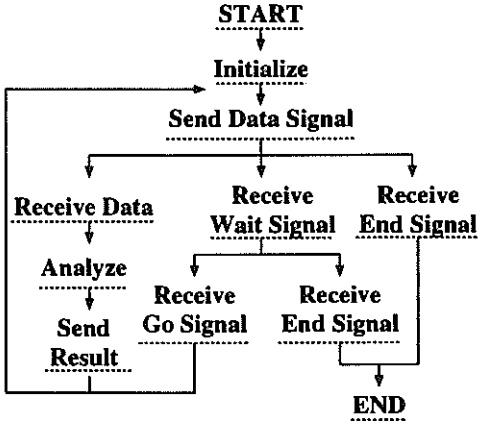

Figure 11: Flow chart of Child processor

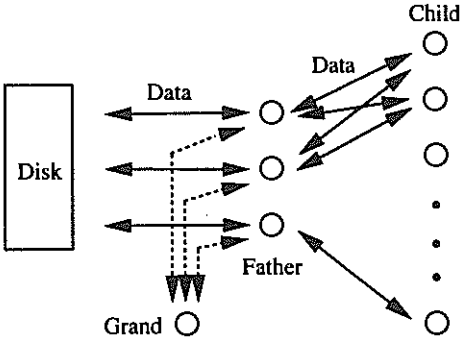

Figure 12: Schematic data flow among Grand-Father-Child processors

Figure 13: Pressure Vessel Model

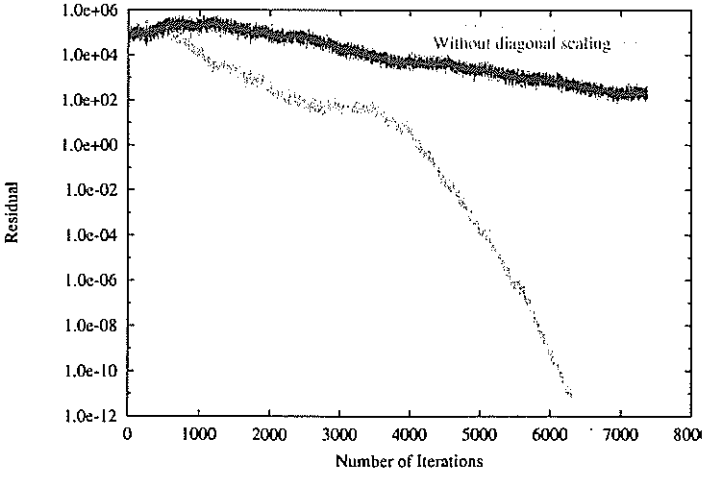

Figure 14: Residual vs number of iterations