EWS ネットワークによる動的問題の並列有限要素解析*

大石 篤 哉^{*1}, 山田 勝 稔^{*1}
吉村 忍^{*2}, 矢川 元 基^{*3}

A Parallel Finite-Element Analysis of Dynamic Problems Using an EWS Network

Atsuya OISHI, Katsutoshi YAMADA,
Shinobu YOSHIMURA and Genki YAGAWA

Computer simulations are increasingly replacing experiments in various fields, and the required scale of the models to be analyzed has become extremely large. To solve such large scale problems, various parallel FEM algorithms have been investigated to date. Among them, the domain decomposition method (DDM) is suitable for parallel computing because of small memory requirement, arbitrary decomposition of the domain, and easy implementation on parallel systems. In this study, we applied a DDM algorithm to dynamic FEM analyses, and implemented it on a network which consists of several engineering workstations. It is well demonstrated through numerical examples that the present DDM based dynamic FEM code shows very high parallel efficiency and fault tolerance.

Key Words: Finite Element Method, Computational Mechanics, Parallel Processing, Domain Decomposition, Dynamic Problem, Network Computing

1. 緒 言

近年の計算機性能の飛躍的向上により、従来模型や実機を用いていた実験を有限要素法などの数値解析手法によるシミュレーションに置き換える動きが盛んになっている。これに伴い、要求される解析の規模とスピードは計算機の進歩をさらに上回る速度で増大している。

こうした大規模な問題を許容しうる速さで解くために、これまでスーパーコンピュータ⁽¹⁾が使われてきた。スーパーコンピュータは、パイプラインプロセッサによりベクトル処理性能を飛躍的に高めたコンピュータである。スーパーコンピュータ用に特化し、その演算能力を最大限に利用するよう設計された高速計算プログラムも数多く開発され成功をおさめてきた^{(2)~(4)}。しかし、これまで性能向上を支えてきたデバイス素子の性能向上が限界に近づきつつあり、計算速度を高めるための新しい手法として、同時に複数の計算機を用いて処理を行う並列処理が注目を集めている⁽⁵⁾。

並列処理環境としては、nCUBE社のnCUBE 2やCray社のT3Dなどの超並列機と呼ばれる専用機、数個のCPUをもつマルチプロセッサ機などのほかに、ネットワークで結合された数台から数十台のEWS (Engineering WorkStation)などがある。こうした並列処理用計算機を有効に利用し高速な処理性能を得るためには、並列処理に適したアルゴリズムが必要とされる。これまでに数々のアルゴリズムが発表されている⁽⁶⁾⁽⁷⁾が、特定の並列処理環境に特化し汎用性に欠けたり、高い並列処理効率を得られないものも多い。

並列処理は、SIMD (Single Instruction stream, Multiple Data stream)とMIMD (Multiple Instruction stream, Multiple Data stream)に分類される⁽⁸⁾。SIMDは特定の問題に対しては実装も比較的容易であり効率もよいが、対象が限定される。MIMDは同期を考慮した効率的な実装が難しいものの、対象に限定されずに汎用性がある。有限要素解析の並列処理手法は、処理単位の観点から、節点(または要素)を単位とするものと領域(複数の節点や要素)を単位とするものに分類できる。このうち、節点を単位とするものは、処理単位が小さく非常に多くのCPUを用いるSIMD向きであり、領域を単位とするものは比較的処理単位が大きく、MIMDに適しているといえる。

* 原稿受付 1995年5月10日。

^{*1} 正員、徳島大学工学部 (〒770 徳島市南常三島町2-1)。^{*2} 正員、東京大学大理工学研究センター (〒113 東京都文京区本郷7-3-1)。^{*3} 正員、東京大学工学部。

本研究では、汎用性に優れた並列処理手法である領域分割法⁽⁹⁾を動的問題の有限要素解析に適用した。領域分割法は、大規模な問題を解くのに適し、ハードウェアに依存せず、高い並列処理効率が得られる。このため、静弾性問題についてはすでに各種の並列環境上で動作し、良好な結果が報告され⁽¹⁰⁾⁽¹¹⁾、さらに静弾塑性問題への適用も行われている⁽¹²⁾⁽¹³⁾。

著者はすでに領域分割法を動的問題に適用する際の定式化を示し、1 CPU 上における計算により精度、速度ともに優れた方法であることを報告した⁽¹⁴⁾。本論文では、領域分割法による動的有限要素解析コードを並列処理環境に実装し並列処理効率の検討を行った結果を報告する。

2. 領域分割法による 動的有限要素解析の定式化

2.1 基礎式および弱形式 簡単のために、図1のような2領域分割の場合の動弾性問題を考える。

領域 Ω 内部において体積力 $\bar{B}$ 、境界 Γ_σ 上で表面力 $\bar{T}$ 、境界 Γ_u 上で変位 $\bar{u}$ が、それぞれ境界条件として与えられているとする。これを2領域に分割すると、

$$\Omega = \sum_{k=1}^2 \Omega^{(k)} \dots\dots\dots (1)$$

となり、各領域の境界は、

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

となる。ここで、 γ_{12} は二つの領域間の境界を表す。

このとき、各部分領域における基礎式は、

$$\varepsilon_{ij}^{(k)} := \frac{1}{2} [u_{i,j}^{(k)} + u_{j,i}^{(k)}] \quad \text{in } \Omega^{(k)} \dots\dots\dots (3)$$

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

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

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

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

であり、さらに、領域間の境界 γ_{12} において、力の釣合いと変位の連続性、すなわち、

$$\tau_{ij}^{(k)} \nu_j^{(k)} + \tau_{ij}^{(l)} \nu_j^{(l)} = 0 \quad \text{on } \gamma_{12} \dots\dots\dots (8)$$

$$u_i^{(k)} = u_i^{(l)} = \mu_i \quad \text{on } \gamma_{12} \dots\dots\dots (9)$$

ただし、

Fig.1 Elastic body split into two subdomains

$$l = k + (-1)^{k-1} \dots\dots\dots (10)$$

が成り立つ。ここで、 u_i , ε_{ij} , τ_{ij} はそれぞれ変位ベクトル、ひずみテンソル、応力テンソルであり、 C_{ijmn} はフックの法則の係数テンソルである。 ν_j は境界における外向き単位法線ベクトルであり、

$$\nu_j^{(k)} = -\nu_j^{(l)} \quad \text{on } \gamma_{12} \dots\dots\dots (11)$$

が成り立つ。 $\dot{u}_i$, $\ddot{u}_i$ はそれぞれ速度・加速度ベクトル、 ρ は質量密度、 c は減衰の係数を意味する、 $(\cdot)_{,j}$ は座標系の j 方向に関する微分を、右肩の添字 (k) , (l) は、注目している部分領域番号(ここでは、1 または 2)を示す。

以上の方程式に重み付き残差法を適用し、弱形式に定式化する。弱形式での定式化に際して、式(9)を厳密に満足させ、式(8)を積分形式で満足させる変位法を採用する。変位法では領域間境界につねに変位拘束が付加されるため任意の領域分割が可能となる。

変位法では、式(3), (4), (7), (9)を厳密に満足させ、式(5), (6), (8)を積分形式で満足させる。すなわち、

$$\sum_{k=1}^2 \left[- \int_{\Omega^{(k)}} (\tau_{ij,j}^{(k)} + \bar{B}_i^{(k)} - \rho \ddot{u}_i^{(k)} - c \dot{u}_i^{(k)}) \delta u_i^{(k)} d\Omega \right. \\ \left. + \int_{\Gamma_\sigma^{(k)}} (\tau_{ij}^{(k)} \nu_j^{(k)} - \bar{T}_i^{(k)}) \delta u_i^{(k)} d\Gamma \right. \\ \left. + \int_{\gamma_{12}} (\tau_{ij}^{(k)} \nu_j^{(k)} + \tau_{ij}^{(l)} \nu_j^{(l)}) \delta u_i^{(k)} d\Gamma \right] = 0 \dots\dots\dots (12)$$

を式(3), (4), (7), (9)のもとで評価すればよい。ここで、式(12)の第1項に発散定理を用い、さらに式(9)に注意すると、

$$\sum_{k=1}^2 \left[\int_{\Omega^{(k)}} \tau_{ij}^{(k)} \delta \varepsilon_{ij}^{(k)} d\Omega + \int_{\Omega^{(k)}} \rho \ddot{u}_i^{(k)} \delta u_i^{(k)} d\Omega \right. \\ \left. + \int_{\Omega^{(k)}} c \dot{u}_i^{(k)} \delta u_i^{(k)} d\Omega - \int_{\Gamma_\sigma^{(k)}} \bar{B}_i^{(k)} \delta u_i^{(k)} d\Omega \right. \\ \left. - \int_{\Gamma_\sigma^{(k)}} \bar{T}_i^{(k)} \delta u_i^{(k)} d\Gamma \right] \\ + \int_{\gamma_{12}} (\tau_{ij}^{(1)} \nu_j^{(1)} + \tau_{ij}^{(2)} \nu_j^{(2)}) \delta \mu_i d\Gamma = 0 \dots\dots\dots (13)$$

という弱形式表示が得られる。

式(13)は、各部分領域 $\Omega^{(k)}$ の一般的な弱形式表示の和に領域間境界 γ_{12} 上での制約条件が付加されたものである。領域分割法では、この制約条件の部分共役こう配法などの反復的な手法により満足させ、解を得る。

2.2 共役こう配法 共役こう配法を用いた領域分割法のアルゴリズムを示す。まず、対称・正定値である演算子 A

$$A(\mu_i) = -(\tau_{ij}^{(1)} \nu_j^{(1)} + \tau_{ij}^{(2)} \nu_j^{(2)})|_{\gamma_{12}} \dots\dots\dots (14)$$

を導入する⁽¹⁵⁾。この A に対して以下のように共役こう配法を用いる⁽¹⁶⁾。

Step 0

$$\mu_i^r = \mu_i^0 \quad (\mu_i^0: \text{constant}) \quad (15)$$

$$g_i^r = A(\mu_i^0) \quad (16)$$

$$w_i^r = g_i^r \quad (17)$$

$$r \leftarrow 0 \quad (18)$$

なお、式(16)の右辺は、部分領域境界間に適当な変位の初期値 μ_i^0 を与えて、有限要素解析を行い、式(14)より得られる。

Step 1

$$\tau_{ij}^{(k)} = \rho \ddot{u}_i^{(k)} + c \dot{u}_i^{(k)} \quad \text{in } \Omega^{(k)} \quad (19)$$

$$\tau_{ij}^{(k)} \nu_j^{(k)} = 0 \quad \text{on } \Gamma_\sigma^{(k)} \quad (20)$$

$$u_i^{(k)} = 0 \quad \text{on } \Gamma_u^{(k)} \quad (21)$$

$$u_i^{(k)} = w_i^r \quad \text{on } \gamma_{12} \quad (22)$$

を解いて得られた $\tau_{ij}^{(1)}, \tau_{ij}^{(2)}$ より、

$$A(w_i^r) = -(\tau_{ij}^{(1)} \nu_j^{(1)} + \tau_{ij}^{(2)} \nu_j^{(2)})|_{\gamma_{12}} \quad (23)$$

を計算する。

Step 2

$$\mu_i^{r+1} = \mu_i^r + \alpha_r w_i^r \quad (24)$$

$$g_i^{r+1} = g_i^r - \alpha_r A(w_i^r) \quad (25)$$

$$\alpha_r = \frac{w_i^r g_i^r}{w_i^r A(w_i^r)} \quad (26)$$

$$w_i^{r+1} = g_i^{r+1} + \beta_r w_i^r \quad (27)$$

$$\beta_r = -\frac{g_i^{r+1} A(w_i^r)}{w_i^r A(w_i^r)} \quad (28)$$

Step 3

収束(領域間境界で反力が釣り合うこと)すれば Step 4 へ。未収束の際は、 $r \rightarrow r+1$ として Step 1 へ戻る(具体的な収束判定条件については解析例の項で述べる)。

Step 4

領域間境界の変位の収束値 μ_i が得られたので、これを用いて再び式(3)(7)および式(9)を解くこと

により、全体の解が得られる(step 0-step 3では領域間境界の節点の解しか得られていないため、この Step が必要となる)。

本アルゴリズム中の $A(w)$ の計算に現れる、各領域における動的有限要素解析とは、後述する時間積分法を用いたものである。そこでは、直前の時間ステップにおける変位の値等のデータが境界条件として使われる。すなわち、Step 0における $A(\mu)$ の計算のときは各点の直前の変位等を用い、Step 1における $A(w)$ の計算のときは、それらはすべて零として扱う。

大規模な問題に対して収束速度向上のために前処理付き共役こう配法(PCG法)⁽¹⁷⁾が用いられることが多く領域分割形解法に適した前処理法もいくつか提案されているが⁽¹⁸⁾、今回は比較的小規模の問題を用いて並列環境における本アルゴリズムの基本性能の評価を行うため前処理は省略した。

2.3 空間離散化

式(13)の[]内を空間座標に

関して離散化し、マトリックス表示すると、

$$[M]\{\ddot{u}\} + [C]\{\dot{u}\} + [K]\{u\} = \{F\} \quad (29)$$

となる。ここで、 $[M]$ は質量マトリックス、 $[C]$ は減衰マトリックス、 $[K]$ は剛性マトリックス、 $\{F\}$ は外力ベクトルであり、 $\{\ddot{u}\}$ は加速度ベクトル、 $\{\dot{u}\}$ は速度ベクトル、 $\{u\}$ は変位ベクトルを表す。

2.4 時間離散化(直接時間積分法:陰解法)

直接時間積分法にはいくつかの方法があり、ここでは陰解法の例として Newmark- β 法による定式化を示す。

Newmark- β 法は、 δ, β をパラメータとして、

$$\begin{aligned} & \left([K] + \frac{\delta}{\beta \Delta t_n} [C] + \frac{1}{\beta (\Delta t_n)^2} [M] \right) \{u\}_{n+1} \\ &= \{F\}_{n+1} + [M] \left\{ \left(\frac{1}{2\beta} - 1 \right) \{\ddot{u}\}_n \right. \\ & \quad + \frac{1}{\beta \Delta t_n} \{\dot{u}\}_n + \frac{1}{\beta (\Delta t_n)^2} \{u\}_n \left. \right\} \\ & \quad + [C] \left\{ \left(\frac{\delta}{2\beta} - 1 \right) \Delta t_n \{\ddot{u}\}_n + \left(\frac{\delta}{\beta} - 1 \right) \{\dot{u}\}_n \right. \\ & \quad \left. + \frac{\delta}{\beta \Delta t_n} \{u\}_n \right\} \quad (30) \end{aligned}$$

となる。ここで、 $\{u\}_n, \{u\}_{n+1}, \{\dot{u}\}_n, \{\dot{u}\}_{n+1}, \{\ddot{u}\}_n, \{\ddot{u}\}_{n+1}$ はそれぞれ、時刻 t_n, t_{n+1} における変位および速度、加速度ベクトルである。このように、Newmark- β 法は各部分領域での有限要素解析において、系の左辺マトリックスに剛性マトリックス $[K]$ を含むため、マトリックスの反転が必要な陰解法である。

陰解法(例: Newmark- β 法)の領域分割法のアルゴリズムをまとめると次のようになる。

(0) $n \leftarrow 0$

(1) 共役こう配法(Step 0 から Step 4)

Fig. 2 Flow chart of the dynamic domain decomposition method

- (2) $n+1$ 時間ステップの解の出力
- (3) $n \leftarrow n+1$
- (4) 最終時間ステップでなければ(1)へ戻る。
- (5) 終了

共役こう配法の Step 0 で行う各領域の有限要素解析は式(30)であり、Step 1 における各領域の有限要素解析は式(30)の右辺を零とおいた式である。

図 2 にフローチャートを示した。時間積分のループの内側に共役こう配法のループを有している。

2.5 時間離散化(直接時間積分法：陽解法) 陽解法の例として中心差分法による定式化を示す。中心差分法は、

$$\begin{aligned} & \left(\frac{1}{(\Delta t)^2} [M] + \frac{1}{2\Delta t} [C] \right) \{u\}_{n+1} \\ &= \{F\}_n - \left([K] - \frac{2}{(\Delta t)^2} [M] \right) \{u\}_n \\ & - \left(\frac{1}{(\Delta t)^2} [M] - \frac{1}{2\Delta t} [C] \right) \{u\}_{n-1} \dots\dots\dots (31) \end{aligned}$$

となる。 $\{u\}_n, \{u\}_{n+1}$ は、時刻 t_n, t_{n+1} における変位である。中心差分法は、集中質量マトリックスを用い、減衰マトリックスが対角マトリックスであれば、左辺マトリックスが対角マトリックスとなり、マトリックスの反転操作が不要な陽解法となる。ただし、陽解法は条件安定(それに対し一般に陰解法は絶対安定)であり、適用には制限がある。

陽解法の領域分割法においては、式(31)から明らかに $\{u\}_{n+1}$ の各成分は独立に求めることができる。このため、さきに示した共役こう配法のアルゴリズムの Step 0 で得られた初期の不釣り合い量から直接に領域間境界上の変位を求めることができる(当然、共役こう配法を用いた、陰解法と同じ定式化もできる)。

陽解法の領域分割法アルゴリズムは次のようになる。演算子 A は式(14)で定義されているとする。

Step 0 E

$$\mu_i = \mu_i^0 \quad (\mu_i^0 : \text{arbitrary}) \dots\dots\dots (32)$$

$$g_i = A(\mu_i^0) \dots\dots\dots (33)$$

式(33)の右辺は先と同じく、部分領域間境界の変位を μ_i^0 として有限要素解析を行い、式(14)より求める。このとき、領域間境界以外の節点については解が決まる。

Step 1 E

$$\mu_i^{\text{solution}} = \mu_i - \frac{g_i}{s_i} \dots\dots\dots (34)$$

ここで

$$s_i = \frac{1}{(\Delta t)^2} \sum_{k=1}^2 m_i^{(k)} + \frac{1}{2\Delta t} \sum_{k=1}^2 c_i^{(k)} \dots\dots\dots (35)$$

ただし、 $m_i^{(k)}, c_i^{(k)}$ はそれぞれ、領域間境界の i 番めの自由度に関する、領域 k の質量マトリックス $[M]$ 、減衰マトリックス $[C]$ の成分である。これで領域間境界の節点変位が決まるので、Step 0 E で求めた他の節点変位と合わせて全節点の変位が決まる。

陽解法の領域分割法のアルゴリズムをまとめると次のようになる。

- (0) $n \leftarrow 0$
- (1.1) Step 0 E
- (1.2) Step 1 E
- (2) $n+1$ 時間ステップの解の出力
- (3) $n \leftarrow n+1$
- (4) 最終時間ステップでなければ(1)へ戻る。
- (5) 終了

3. EWS ネットワーク上での並列処理

3.1 EWS ネットワーク 動的領域分割法のアルゴリズムは MIMD 形並列処理環境に容易に実装できる。MIMD 形並列処理環境としては、nCUBE、T3D に代表される超並列機、ネットワークでつながれた EWS 群(多くは Unix OS を備えている)などがある。ここでは、EWS ネットワーク上に動的領域分割法アルゴリズムを実装する。EWS ネットワークは、広く普及しており、現在最も身近な並列処理環境といえる。4.2 BSD 以降の Unix OS は“socket”と呼ばれるプロセス間通信機構⁽¹⁹⁾をもち、異なる EWS 間でのデータのやりとりが容易に行える。この機構を利用することにより、EWS ネットワーク上で並列処理を行うことが可能となる。

EWS ネットワークを並列処理環境として用いる場合、次の点に注意する必要がある。まず、すべてのプロセッサが同じ演算能力を有する超並列機とは異なり、EWS ネットワークでは、さまざまなプロセッサが混在しており各プロセッサ(EWS)の演算能力に差がある。さらに、同等のプロセッサ(EWS)においても、演算処理性能は変動する。これは、Unix がマルチタスクの OS であり、各 EWS は通常複数のユーザのプロセスを処理しているためである。したがって、EWS ネットワーク上で高い並列処理効率を達成するには、各 EWS に演算負荷を適切かつ動的に配分することが重要である。

次に、現在の EWS ネットワークの多くは伝送路として Ethernet を用いている。Ethernet は伝送速度が 10 Mbps (10⁶ bits per second) であり、この値は急激に向上したプロセッサのスピードに比べてかなり遅いものと言わざるを得ない。このため、EWS ネットワーク

上で用いる並列処理アルゴリズムは、各プロセッサで実行される演算に対するプロセッサ間データ通信の比率を小さくすることが要求される。これは、並列計算の単位(粒度)を大きくすることに相当する。

3.2 Client-Server モデル 図3に動的領域分割法アルゴリズムのEWS ネットワークへの実装形式を示す。

Controllerと名付けられた1台のEWSと、Analyzerと呼ばれる複数台のEWSにより並列処理が行われる。ControllerおよびAnalyzerの役割は次のようにまとめることができる。Controllerは各部分領域のメッシュデータ、境界条件、さらに、部分領域間の接続情報など、有限要素解析に必要なすべてのデータを管理し、Analyzerからの要求に応じて、ネットワークを通じて必要なデータを各Analyzerへ供給する。また、ControllerはAnalyzerから部分領域の有限要素解析の結果を受け取る。

Client-Serverモデル⁽²⁰⁾を構成するために、Controllerでは次の三つのプロセスが同時に動作している。DCC (Domain Connectivity Controller)は主たるプロセスであり、すべてのデータを管理し、共役この配法による部分領域間境界の変位の更新を行う。

Fig. 3 Client-Server model

Fig. 4 Flow chart of the Domain Connectivity Controller

Sending ServerはAnalyzerの要求に応じて、部分領域の有限要素解析に必要なデータをAnalyzerへ引き渡す。Receiving ServerはAnalyzerから、部分領域の有限要素解析の結果を受け取る。

各部分領域の有限要素解析に必要なデータはDCCが用意し、Sending Serverを通してAnalyzerへ送られる。いずれかのAnalyzerで部分領域の解析が終わると、その解析結果はReceiving Serverを通してDCCへ送られる。二つのサーバプロセス(Sending ServerとReceiving Server)は要求に応じて自分自身の複製を作り増殖することができる。このため、同時に複数のAnalyzerとのデータ通信が可能である。増殖した各サーバプロセスはそれぞれが独立してAnalyzerとの通信を行うため、Analyzerはcontrollerから遅滞なく必要なデータを受け取ることができ、各Analyzerを最大限に利用することができる。図4に、DCCのフローチャートを示す。

DCCはイベントドリブン機構をもち、Sending ServerやReceiving Serverのデータ転送要求にスムーズに応じることができる。Analyzerでは、DA (Domain Analyzer)がControllerから部分領域の有限要素解析に必要なデータを受け取り、有限要素解析を行い、解析結果をControllerに送り、新たな領域の解析のためのデータをControllerに要求する。図5にDAのフローチャートを示す。

ControllerとAnalyzerの間では、メッシュデータや境界条件といった少量のデータがやりとりされるだけである。

3.3 計算負荷の均等化 高い並列処理効率を達成するには、各プロセッサ間で計算負荷を均等化する必要がある。動的領域分割法アルゴリズムをEWS ネットワークに実装する際、通常、部分領域の数はAnalyzerの数よりもかなり大きいと考えてよい。これは、数台から十数台程度のEWSを用いて大規模な問題を解く場合などに相当する。

Client-Server機構により、ControllerはAnalyzer

Fig. 5 Flow chart of the Domain Analyzer

の要求に対し遅滞なくデータを渡すことができ、何も処理を行っていない Analyzer はすみやかに新しい解析のためのデータが渡される。こうして、処理の終わった EWS に順次新しい処理を行わせることで、各 EWS 間での計算負荷が動的かつ自動的に均等化され、各 EWS の能力を最大限に利用することが可能となる。この手法によれば、各 EWS 間の演算性能のバラツキによる並列処理効率の低下は最小限に抑えられる。さらに、この手法は、次のように容易に fault tolerant 性をもたせることができる。すなわち、ある部分領域の解析を行っていた Analyzer が極端に性能が劣化したり、なんらかの原因でシステムダウンしたような場合、Controller は障害を検知し、他の Analyzer にその部分領域の解析を行わせることができるのである。

4. EWS ネットワーク上での動的領域分割法

4.1 問題の設定 動的領域分割法アルゴリズムを EWS ネットワーク上に実装し、二次元弾性体の衝撃応答問題を解き、並列処理効率を検証した。図 6 は要素分割および領域分割を示している。

各要素は 4 節点アイソパラメトリック平面ひずみ要素であり、全要素数は 1600 (40×40) である。部分領域への分割として、部分領域数 16, 25, 64, 100 の 4 種類について検討した。

材料定数は、ヤング率 200 GPa、ポアソン比 0.3、質

Fig. 6 Domain decompositions of a 2D elastic square plate

Fig. 7 Configuration of the EWS network used

量密度 $7.88 \times 10^3 \text{ kg/m}^3$ とした。荷重は上端の片側半分に 5 MPa のステップ荷重をかけた。集中質量マトリックスを用い、減衰はないものとした。時間積分法は、陰解法としては Newmark- β 法(時間刻み 1×10^{-7} 秒)、陽解法としては中心差分法(時間刻み 5×10^{-8} 秒)を用いた。なお、Newmark- β 法の二つの定数 δ , β はそれぞれ 0.0, 0.25 とした。

4.2 EWS ネットワーク 使用した EWS ネットワークの構成を図 7 に示す。リング状に接続された 5 台の Apollo 社製 EWS を使用した。Controller として DN 10000 を 1 台 (node_0) 使用し、Analyzer として DN 3500 を 3 台 (node_1, node_2, node_3) と DN 3000 を 1 台 (node_4) 使用した。伝送路は Ethernet と同等の性能をもつ Apollo Token Ring (12 Mbps) である。プロセス間通信には "socket" を用いた。

EWS の実効性能は機種ごとに異なり、さらには同一機種でも同時に動作するプロセスによって実効性能は変化する。本解析の際は、5 台の EWS を占有したため、他のユーザに起因する実効性能の変化は無視できる。さらに、伝送路もほぼ占有状態であり、並列処理性能の測定条件は統一されていると考えられる。

Table 1 Power ratio (PR) of EWSs

	node_1	node_2	node_3	node_4
PR	1.00	1.00	1.00	0.45

Table 2 Four different parallel computing environment

	CASE 1	CASE 2	CASE 3	CASE 4
EWS	node_1	node_1 node_2	node_1 node_2 node_3	node_1 node_2 node_3 node_4
Total PR	1.00	2.00	3.00	3.45

Fig. 8 Time charts of working states of processors (CASE3)

並列処理効率の測定の前に、Analyzer 1 台だけで部分領域数 16 の場合の解析を行い Analyzer の性能評価を行った。node_1 の EWS (DN 3500) を基準 EWS とし、解析に要した時間から各 EWS の性能指標 “PR” を次のように求め、表 1 に示した。

$$PR = \frac{\text{基準 EWS での所要時間}}{\text{対象 EWS での所要時間}} \quad (36)$$

本研究では CASE 1 から CASE 4 まで、Analyzer の数に応じて 4 種類の並列処理環境でテストした。表 2 にそれぞれの構成と、その環境における “PR” 値の和を示した。

並列処理効率は次式で定義した。

$$PE_n^k = \frac{T_n^1}{T_n^k \times N^k} \times 100 \quad [\%] \quad (37)$$

ここで、

PE_n^k : CASE k における部分領域数 n の場合の並列処理効率

T_n^1 : CASE 1 における部分領域数 n の場合の実行時間

T_n^k : CASE k における部分領域数 n の場合の実行時間

N^k : CASE k における “PR” の合計

である。

4.3 陰解法における並列処理効率 まず、並列処理環境における動的領域分割法アルゴリズムによる

処理の進行状況を検証する。図 8 は部分領域数 16、並列処理環境 CASE 3 における各 Analyzer の処理の進行状況を示したものである。

横軸に時間を取り、小さな矩形一つ一つが部分領域 1 個の有限要素解析に相当する。点線は共役こう配法の 1 回の反復の開始と終了を表している。この図から、Controller-Analyzer 間の通信によるオーバーヘッドが無視できるほどに小さいことがわかる。

共役こう配法の各反復の最終段階では、いくつかの Analyzer が処理を行っていない状態が発生する。これは、その Analyzer に割り当てられるべき処理がなくなったために起きる。また、共役こう配法の一反復が終了してから次の反復が始まるまで、短時間ながら、すべての Analyzer が何も処理を行っていない時間が存在する。これは、Analyzer が Controller における部分領域間境界変位の更新を待たなければならないために起きる。共役こう配法が収束するまで、各反復において同様な進行状況が繰返し見られる。

表 3 にすべての条件での並列処理効率を示した。ほとんどの条件で 90% 以上の高い並列処理効率を得られていることがわかる。

図 9 は CASE 4 における部分領域数 25 および 64 の二つの場合について、各 Analyzer の解析の進行状況を示したものである。表 3 によると、部分領域数 25 の場合 81% という比較的低い並列処理効率しか得られていないのに対し、部分領域数 64 の場合は 96% という高い並列処理効率を得られている。図 9 から、部分領域数 25 における低い並列処理効率が共役こう配

Table 3 Measured parallel efficiency (Implicit method)

Number of Subdomains	CASE1		CASE2		CASE3		CASE4	
	Time	PE	Time	PE	Time	PE	Time	PE
16	1.00	—	0.50	99	0.38	88	0.32	91
25	1.00	—	0.52	96	0.36	92	0.36	81
64	1.00	—	0.51	99	0.34	97	0.30	96
100	1.00	—	0.51	98	0.34	98	0.30	96

Time : Relative Calculation Time
PE : Parallel Efficiency (%)

Fig. 9 Time charts of working states of processors (CASE3)

Table 4 Measured parallel efficiencies (Explicit method)

Number of Subdomains	CASE1		CASE2		CASE3		CASE4	
	Time	PE	Time	PE	Time	PE	Time	PE
16	1.00	—	0.50	99	0.37	89	0.31	93
25	1.00	—	0.52	96	0.37	91	0.35	82
64	1.00	—	0.50	99	0.35	95	0.31	95
100	1.00	—	0.51	98	0.35	96	0.31	93

Time : Relative Calculation Time
PE : Parallel Efficiency (%)

Fig. 10 Dynamic work load balancing among processors

法の各反復の最終段階で比較的長い時間いくつかの Analyzer に処理が割り振られないことによるものであることがわかる。

本研究で採用した動的領域分割法のアルゴリズムで高い並列処理効率を得るためには、共役こう配法の各反復においてアイドル状態の Analyzer が生じる時間の割合を最小限にする必要がある。これは、部分領域の数が Analyzer よりもはるかに多いとき、例えば大規模な解析を行うときには自動的に満足され、高い並列処理効率を得られる。

次に、Client-Server 機構による動的負荷分散の様子を示す。図 10 は並列処理環境 CASE 3、部分領域数 16 における各 Analyzer の解析進行状況を示したものである。ここでは、途中から 1 台の Analyzer (node_3) の処理能力を半分に落として、進行状況の変化を見た。図から、node_3 へ割り当てられる領域数が減少し、各 Analyzer の処理能力に応じて負荷が動的に分散されていることがわかる。

4.4 陽解法における並列処理効率 表 4 に陽解法における並列処理効率を示す。陰解法と同様に高い並列処理効率を得られていることがわかる。

4.5 最適な領域分割の基準 大規模解析の場合について、領域分割の際の指針として次のことに留意する必要がある。

- ・各部分領域は Analyzer の主記憶内で有限要素解析が可能である程度に小さい。
- ・部分領域の数は、Analyzer 数の数倍から数十倍程度以上である。

本研究では、規則的な要素分割の場合しか扱っていないが、不規則な要素分割、領域分割の場合でも、動的負荷分散機構により同様に高い並列処理効率が期待される。この手法では、「各部分領域の計算負荷を同等にする」という領域分割への要求が緩和されるため、要素分割や領域分割というプリプロセスの負担が軽減される。

なお、本研究で採用した共役こう配法は収束速度が演算子 A の条件数に依存する。領域分割の際にはこの点にも注意が必要である。

5. 結 言

動的領域分割法の特徴は次のようにまとめられる。

- (1) 有限要素解析を個々の部分領域に分割して行うため、記憶容量が少なく済み、大規模解析に適している。
- (2) 各部分領域の有限要素解析は独立しており、

粒度が大きく、高い並列効率を得られる。

(3) 各部分領域の有限要素解析は従来のコードが使えるので、資産の継承が可能であり、インプリメントも容易である。

(4) 領域数、分割方法が任意に選べるため、汎用性が高く、ハードウェア依存性が低い。

こうした特徴をもつ動的領域分割法を、MIMD 形並列処理環境として EWS ネットワーク上に実装し、陰解法、陽解法それぞれの場合について並列処理効率を測定し、

(5) 陰解法、陽解法ともに 90% 以上の高い並列処理効率を得られる。

(6) Client-Server 形の実装を行うことによりプロセッサの性能に応じて動的に負荷を分散させることができる。

文 献

- (1) 村田健郎・ほか 2 名, スーパーコンピュータ, (1985), 1-52, 丸善.
- (2) 津田孝夫, 数値処理プログラミング, (1988), 137-148, 岩波書店.
- (3) 三好俊郎・ほか 2 名, 第 3 回計算力学講演会講演論文集, No. 900-69 (1990-10), 127-128.
- (4) 矢川元基・赤星保浩, シミュレーション, 8 (1989), 49-55.
- (5) 高橋義造 (編), 並列処理機構, (1989), 1-66, 丸善.
- (6) Storaasli, O., ほか 2 名, *Comput. Struct.*, **26** (1987), 551-559.
- (7) Farhat, C. and Wilson, E., *Int. J. Num. Methods Eng.*, **24** (1987), 1771-1792.
- (8) Lewis, T. G. and El-Rewini, H., *Introduction to parallel computing*, (1992), 18-22, Prentice Hall.
- (9) Yagawa, G., ほか 2 名, *Comput. Struct.*, **38** (1991), 615-625.
- (10) Yagawa, G., ほか 2 名, *Comput. Struct.*, **47** (1993), 407-418.
- (11) Yagawa, G. and Shioya, R., *Comput. Systems Eng.*, **4** (1993), 495-503.
- (12) Soneda, N., ほか 2 名, *Trans. 11th SMiRT Conf.*, B (1991), 243-248.
- (13) 矢川元基・内田善彦, 機論, **60**-580, A (1994), 2737-2742.
- (14) 大石篤哉・ほか 3 名, 機論, **58**-552, A (1992), 1445-1452.
- (15) Glowinski, R., ほか 2 名, *Comput. Meth. Appl. Mech. Eng.*, **40** (1983), 27-109.
- (16) 戸川隼人, 共役こう配法, (1977), 127-133, 教育出版.
- (17) 村田健郎, ほか 2 名, 大型数値シミュレーション, (1990), 95-114, 岩波書店.
- (18) Tallec, P. Le, *Domain decomposition methods in computational mechanics*, (1994), 155-166, North-Holland.
- (19) Stevens, W. R., *Unix Network Programming*, (1990), 258-341, Prentice Hall.
- (20) Tanenbaum, A. S., *Computer Networks*, 2nd ed., (1988), 455-456, Prentice Hall.