THE SCIENCE AND ENGINEERING REVIEW OF DOSHISHA UNIVERSITY, VOL. 40, NO. 1 APRIL 1999

The Installation and Performance Evaluation of Parallel Virtual Machine for an Object-Oriented Computation Model

Mitsunori MIKI*, and Masayuki KASAI**

(Received November 4, 1998)

Recently, many applications have been implemented on WS・PC cluster systems using message-passing libraries, such as PVM and MPI. This paper addresses the performance evaluations of cluster computing for an object-oriented computation model by implementing truss analysis code on a PC cluster system, which consists of 8 PCs, Ethernet, and PVM. The effects of granularity, problem size, process mapping, and communication to the efficiency of parallel processing are investigated. From several numerical experiments, the effective granularity and computation model for cluster computing can be shown.

Key words：parallel processing, cluster computing, PVM, analysis of truss structure, granularity

キーワード：並列処理，クラスタコンピューティング，PVM，トラス構造物解析，粒度

PVMによる仮想並列計算機の構築と オブジェクト指向型計算モデルにおける性能評価

三 木 光 範・笠 井 誠 之

1. はじめに

近年，多数のWSやPCをネットワークで接続しクラスタ化したものを，並列計算機として利用するクラスタコンピューティングの研究が盛んになってきた．またそのためのメッセージ通信ライブラリとして，PVM[1)]やMPI[2)]などが開発され，利用されてきている．しかし現状では，メッセージ通信におけるオーバーヘッドが大きく，わずか数台で処理速度の向上が止まり，WS・PCクラスタの規模に見合うだけの処理性能が得られておらず，実行性能の改善が求められている[3)]．

実行性能を改善するには，WS・PCクラスタの性能を計測することによって，アプリケーションやクラスタを構築する際の設計指針を見つけなければならない．そこで，これまでPVMの性能評価方法には，PVMのメッセージ通信関数そのものの性能測定を行ったもの[3)4)]や，行列演算やソーティングといった一般的なベンチマークプログラムを用いる方法[3)]がある．また実用的なアプリケーションに特定した性能評価としては，並列GAなどの比較的粒度の大きいもので行われてきた[5)]．しかしながら，粒度の細かい実用的アプリケーションを用いた性能評価は行われていない．そこで，そのようなアプリケーションにクラスタコンピューティングを適用し，その際の問題点について研究することは，今後の並列アルゴリズムの設計に大いに意義のあることであると考えた．

* Department of Knowledge Engineering and Computer Science, Doshisha University, Kyoto
Telephone:0774-65-6434, Fax:0774-65-6796 E-mail:mmiki@mail.doshisha.ac.jp
** Graduate Student, Department of Knowledge Engineering and Computer Science, Doshisha University, Kyoto
Telephone/Fax:0774-65-6716 E-mail:mkasai@mikilab.doshisha.ac.jp

本研究では，オブジェクト指向型の計算モデルを基礎とするトラス構造物の解析[6]の並列化という粒度の小さいアプリケーションに，PVMによる並列処理を適用し，その並列処理能力の評価を行った．そのための実験環境として，8台の既存の計算機(WS・PC)と10Base-TのEthernet，そしてPVMを用いてヘテロジニアスなクラスタを構築した．本稿ではその実験結果として，プロセスのマッピング方法と粒度のチューニングを行った結果，そして，通信方法を変更した結果について報告する．

2. 仮想並列計算環境PVM

2.1 ＰＶＭシステム

PVMは，テネシー大学とオークリッジ国立研究所で開発されたソフトウェアであり，ネットワークに接続された異機種コンピュータ群を，単一のメッセージパシング型並列計算機として利用することを可能にするものである．これによって，多数のコンピュータの持つ計算パワーを，1つの大規模計算問題に結集して利用することができる．1989年にVersion 1がリリースされ，現在の最新バージョンは1997年12月31日にリリースされたVersion 3.4 beta 6(1998年6月現在)である．これらは，WWW上のサイトhttp://www.epm.ornl.gov/pvm/からフリーで入手できる[1)7)]．

2.2 ＰＶＭを構成するソフトウェア

PVMは2つの部分で構成される．PVMデーモンとＰＶＭユーザーインタフェースライブラリである．PVMデーモンはpvmdと呼ばれる．デーモンの機能は，プロセス間通信の中継，メッセージのルーティング(経路制御)，プロセス管理，エラー検出である．デーモンは，バーチャルマシンを構成する全てのコンピュータ上に常駐し，アイドル時にお互いの状況を監視している．ユーザの並列アプリケーションがクラッシュしたときでも，デーモンは実行され続ける[1)]．

PVMユーザインタフェースライブラリは，PVMの機能をユーザアプリケーションから呼び出すための基本命令を集めたものであり，ユーザアプリケーションのデーモンへの及び他のプロセスへのインタフェースである．ユーザアプリケーションをこのライブラリとリンクさせることで，PVMが提供する並列処理のための機能を利用することができる．このライブラリの機能は，FORTRAN，C及びC++から利用することができる[1)]．

2.3 ＰＶＭアプリケーションの開発

ＰＶＭにおける並列性の単位はプロセスであり“PVMタスク”と呼ばれている．ユーザは，アプリケーションが複数のプロセスによって構成されるという概念に基づいて，アプリケーションプログラムを開発することになる．それぞれのプロセスが計算量を分担しており，それらのプロセスを，バーチャルマシンを構成する各計算機に配分することで，並列処理が行われる[1)]．

2.4 本研究で用いた計算機

本研究で用いた計算機をTable 1に示す．この表は，各計算機の名前(Machine name)，CPU，OS，クロック数(Clock)，そして，相対速度比(Relative Speed)を示す．なお，この相対速度比というのは，後述のトラス構造物解析プログラムをPvm00で計算したときの時間を，各計算機一台のみで計算したときの時間で割った値であり，各計算機がPvm00に対してどれだけ速いかを表す指標である．大きいほど高速であることを示している．仮想並列計算環境としては，これらの計算機を10Mbpsの10Base-TによってLAN接続とした環境を用いる．この環境は，各々の計算機の性能が異なっている．Table 1より，計算機の速度を決定するのは，CPUの種類，クロック数だけでなく，OSの種類も関わってくることが分かる．また，Pvm04とPvm05に関しては，同じCPUとOSでも異なる性能を示しているが，その理由は，Pvm04にPVMと無関係ないくつかのサーバプログラムが実行され

Table 1. Machines list

Machine name	OS	CPU type	Clock	Relative speed
Pvm00	Solaris 2.3	MicroSPARC	50MHz	1
Pvm01	SunOS 4.1.3	MicroSPARC	50MHz	1.33
Pvm02	SunOS 4.1.3	MicroSPARC	50MHz	1.34
Pvm03	SunOS 4.1.3	SuperSPARC	33MHz	2.75
Pvm04	Linux 2.0.29	intel Pentium	166MHz	7.97
Pvm05	Linux 2.0.30	intel Pentium	166MHz	9.34
Pvm06	Linux 2.0.29	intel Pentium with MMX	200MHz	10.09
Pvm07	Linux 2.0.29	intel Pentium Ⅱ	233MHz	13.63

ており，負荷が高くなっている．なお，Pvm04以外の計算機はPVM関係の計算を専用に行うようになっている．この環境では，並列アプリケーションの性能は最も性能の低いPvm00に影響を受ける．本研究では，負荷分散については考慮せず，Pvm00に大きな影響を受けている条件下で，全体としての処理効率を考えるものとした．

3. トラス構造物の並列解析

3.1 トラス構造物

本研究では，PVMによる並列処理を適用する問題としてトラス構造物の解析問題を採用した．トラス構造物とは直線部材を摩擦の無いピンで接合した構造物であり，例えば送電線の鉄塔や鉄道橋などで広く利用されている．実際のトラス構造物では多くの場合ピンではなく，ガセットプレートによって部材を接合するが，力学の対象としてのトラスではピン接合の仮定を用いて単純化する．Fig.1の左にモデル化されたトラスを示す．さらに，ピンの空間的な広がりは無視して点であると考え，節点あるいはノードと呼ぶ．

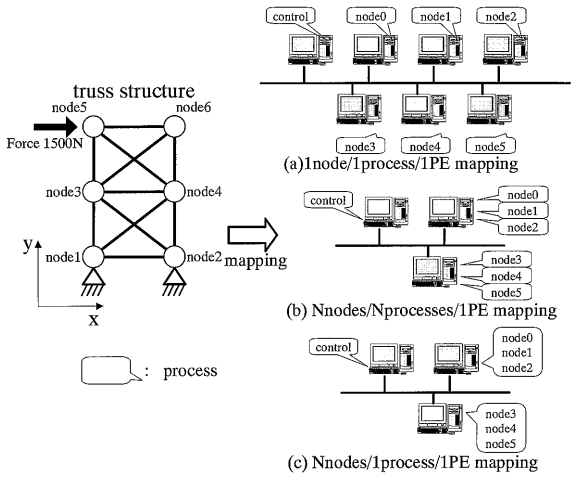

Fig. 1. Mapping truss nodes to processes

3.2 トラス構造物の解析

トラス構造物の構造解析とは，トラスに負荷を加えたときに，どのように変形するかを求める問題である．例えば，Fig.1では5番の節点に1500Nの力がx方向に負荷されており，その時の各節点の変位を計算することになる．

このようなトラス構造解析は一般的に有限要素法[8]などを用いて行われるが，三木はオブジェクト指向に基づく計算モデルを提案し，幾何学的非線形性や材料非線形性を容易に取り扱えるようにしたのみならず，部材を能動化したVGT(形状可変トラス)の形態解析と構造解析の統合化を行った[6]．

その方法を簡単に説明する．まず，トラス構造物に関する知識を要素レベルの知識に分散させる．トラス構造は，いくつかの節点と部材により構成される．すなわち，トラスをモデリングする際に必要となる知識は各節点と各部材の知識の集合となる．節点の知識はさらに，座標値，外力，接続されている部材，支持条件，不平衡力などの知識にわかれ，部材の知識は初期長さ，材料，断面形状，接続節点などの知識に分れる．問題解決のための分散アルゴリズムは，部分問題を節点における力の平衡と考える．

トラスの各節点では，以上の知識を基に式(1)で定義される節点の不平衡力が発生する．

$$\vec{f}^{*} = \begin{cases} \vec{0} & \text{(rotational support)} \\ \vec{f}_e + \sum_{k=1}^{n} \vec{f}_k & \text{(no support)} \end{cases} \qquad (1)$$

ここで，$\vec{f}^{*}$ は節点にかかる不平衡力，$\vec{f}_e$ は作用外力，nは節点につながっている部材数，$\vec{f}_k$ はk番目の部材の部材力を表す．力が釣り合っている状態では節点の不平衡力はすべての節点で0となる．節点に外力が作用すると，その節点で不平衡力が生じ，その節点は不平衡力をなくす方向および距離を求め，その位置に移動する．このとき，他の節点はすべて固定されているとして移動位置を求める．次に，この節点の移動によりその節点に接続している部材長が変化し，ひずみを誘起し，そのひずみが材料内部で変換され応力となる．こうして1つの節点の移動が次々に他の節点の移動を誘発し，これが繰り返されることによって，最終的にすべての節点で力の平

衡条件が満たされる状態が得られる．

このアルゴリズムを具体的に説明する．節点の移動距離は，節点の微小変動に対して節点力の感度により計算される．移動変位は式(2)により得られる．

$$[d] = [s][f] \tag{2}$$

ただし，

$$[\mathrm{d}]=\begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}, [s]=\begin{bmatrix} f_{x,x} f_{x,y} f_{x,z} \\ f_{y,x} f_{y,y} f_{y,z} \\ f_{z,x} f_{z,y} f_{z,z} \end{bmatrix}^{-1}, [\mathrm{f}]=\begin{bmatrix} f_x \\ f_y \\ f_z \end{bmatrix} \tag{3}$$

ここで, [d]は移動変位ベクトル, [s]は節点の微小変位ベクトル(Δx, Δy, Δz)に対しての節点力の感度マトリックスの逆行列, [f]は節点力の不平衡力ベクトルである．

また，収束判定は極めて容易で，各節点における平衡方程式の残差が指標になる．

このように，分散化された知識を並列計算機上に割り当てることによる分散アルゴリズムの有効性は，三木及び小板[9]によって明らかにされており，並列処理による計算時間の大幅な短縮が可能となっている．

4. 並列解析プログラムの実装と性能評価

4.1 実装方法

トラス構造物解析プログラムをPVMに実装する方法としては，節点の移動変位を計算する部分を並列化する方法を採用した．つまり，トラスの節点をプロセス(PVMタスク)に割り当て，それぞれのプロセスが，受け持つ節点だけの節点力，移動変位を求めるということである．Fig.1には，節点をプロセスに割り当てるときの3つの実装方法を示しており，それぞれについては次節以降で説明する．この解析プログラムは以前，三木及び小板[9]によって，米nCUBE社の超並列計算機nCUBE2上に実現されていたものであった．そこで，PVMへの実装は，プロセッサ間通信のコードの変更による移植という形で行った．

トラスの構造解析における局所制約条件は節点における力の平衡であり，これを満足するように節点が移動する．このとき，各節点に関する節点剛性マトリックスは保持しておらず，その都度評価する．得られた節点剛性マトリックスをもとに各節点で，力の平衡方程式を満足するようその節点の変位を求め，更新された節点の座標がプロセス間で交換され，その座標をもとに新たに節点の変位を求める．こうした，局所制約条件を全節点で並列に繰り返し解くことにより，トラス構造の並列解析が可能となる．

トラスの節点情報を保持するプロセスがお互いの更新された節点座標情報を交換する際の方法として，三木及び小板[9]は同期型と非同期型の方法を提案している．同期型の方法とは，全てのプロセスが各々受け持っている節点の不平衡力を計算して，節点の変位を求めるまでを1ステップとし，全プロセスがお互いの更新された全情報を交換するまで計算が進行しない方法である．これは，全情報交換が解析の同期を取っていることになる．一方，非同期型とは，節点座標の計算が終わるとその計算結果を必要とするプロセスにのみ，情報の送信を行う．自らの節点の計算も関連する他の節点情報が他のプロセスから届いていれば，それを用いて節点座標の計算を行い，情報が届いていない場合でも1ステップ前の情報用いて計算を行う．

現在，同期型は，同期をとるためのオーバーヘッドがあり，非同期型より計算時間が大きいが，非同期型と異なって確実に計算が収束することが確認されている．非同期型は高速であるが，収束は保証されない[9]．そこで，本研究では同期型解析を採用し，解析の1ステップごとに全情報の交換を行う通信アルゴリズムを用いる．その方法は，節点計算用プロセスの計算が終了すると，その更新された節点座標を一度，収束判定を行う管理プロセスに集める．計算用プロセスから管理用プロセスへはPVMの通信関数pvm_send()を用いている．そして，全節点の情報が集まりしだい，収束判定を行った管理プロセスが全節点情報を収束判定結果とともに全ての節点計算用プロセスに送信する．この際の送信関数としては，ＰＶＭの複数プロセスへの同報通信関数であるpvm_mcast()を用いた．上述の通信のタイミングをFig.2のチャートに示す．

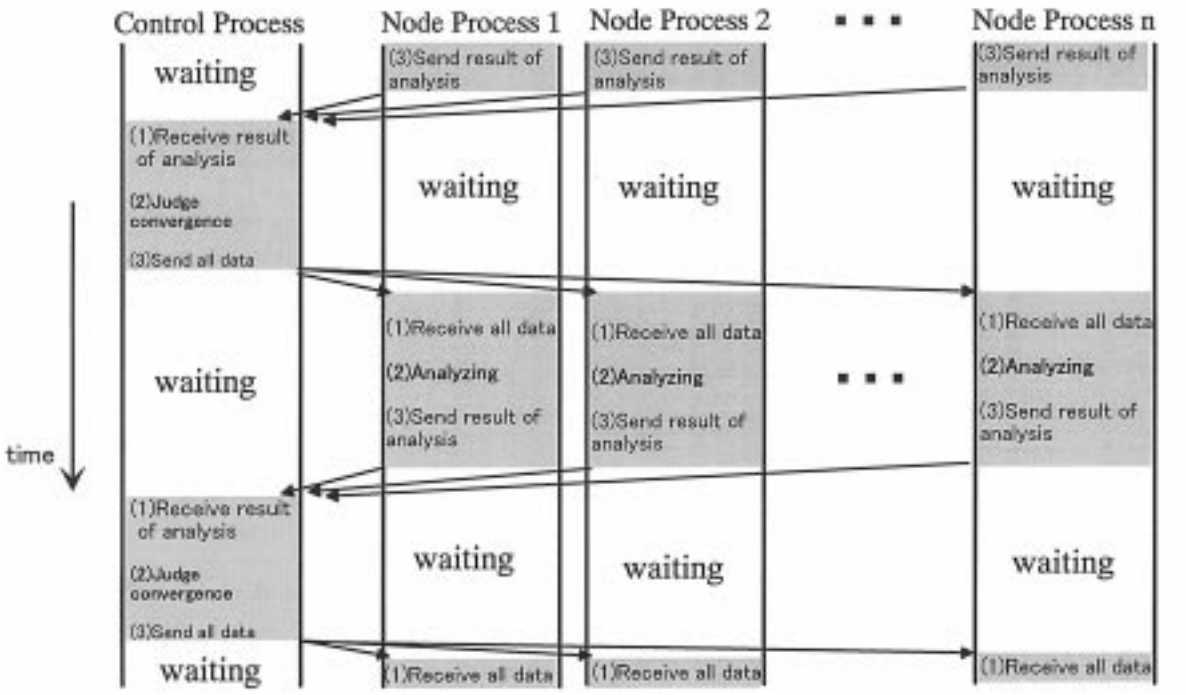

Fig. 2. Chart for the communication timing

4.2 粒度と計算時間

4.2.1 1ノード/1プロセス/1 PEの割り当て

バーチャルマシンにおける5台の計算機を用いた4節点トラスの反復計算1サイクルにおける平均処理時間と，7台の計算機を用いた6節点トラスの反復計算1サイクルにおける平均処理時間をFig.3に示す．このグラフは，バーチャルマシンの中で最も性能の低いPvm00の1台における計算時間とPVMの計算時間との比較を表す．このときのPVM環境の各マシンは，トラスの1節点の計算を行うもので，6ノードトラスの場合，6台の節点計算プロセス用計算機と，1台の管理プロセス用計算機からバーチャルマシンが構成される．その様子を，Fig.1(a)に示す．

Fig.3において，4節点，6節点ともにPVMによる並列処理のほうがPvm00の1台における計算時間より大きいことがわかる．このことは，トラスの1節点を保持するだけでは，並列処理による計算速度の向上は見られないことを表している．またFig.4は，6節点トラスの解析における反復計算1サイクルの中の計算時間と通信時間の平均を，各マシンについて測った結果である．(この時の管理プロセスは，Pvm01上で実行されている．以後同様である．)それによると，計算速度の向上が見られない理由は，管

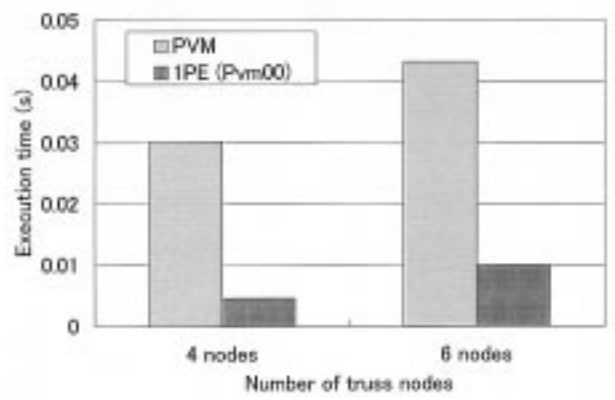

Fig. 3. Comparison of PVM (8 machines) and Pvm00 (stand-alone): average time of each iteration

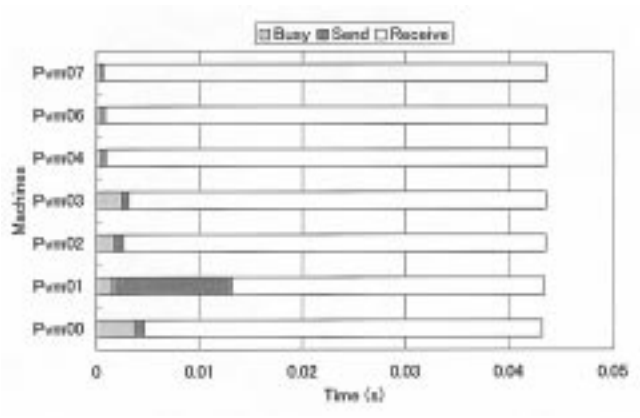

Fig. 4. Calculation and communication time of each machines for 6 node truss analysis

理プロセスに節点座標情報を集め，管理プロセスがそれを他の全計算用プロセスに送信するという処理と，そのための待ち時間が計算用プロセスの計算量と比べて大きいことがその原因として考えられる．

管理プロセスを媒体とした全情報交換は，この並列処理における同期操作となっている．そして同期操作の時間が，計算処理に比べて無視できないほど大きい場合は，並列処理による速度向上が難しい[10]．このとき，一般的にとられる方法としては，同期操作やオーバーヘッド等の時間を削減できない以上，その他の計算量を増加させ，個々のプロセッサの通信量に対する計算量の比率，つまり粒度を大きくする方法がある．粒度を大きくする方法には，問題の規模を大きくする方法と，並列処理の並列度(問題をどれだけの数に分解するかを表す度合)を下げる方法の2つがある．ここでは，問題の規模を大きくしていくことにした．

4.2.2 Nノード/Nプロセス/1PEの割り当て

粒度を上げるために，問題の規模を大きくするならば，各計算機の計算量を増加させなければならない．そこで，各計算機に節点計算用プロセスを複数実行させることで計算量の増加を試みた．これはFig.1(b)のように実行される．この図は，6節点トラスを3台で計算する図であるが，実際の実験では8台の計算機を用いる．そこで，管理プロセスをPvm01に担当させ，それ以外の7台に複数の計算用プロセスを実行させた．プロセス数の増加は，おのおのの計算機内で節点の数が2節点ずつ増加するさせるようにして，問題規模を42節点まで大きくした．その時の結果をFig.5に示す．この図によると，並列処理の効果はまったく現れていない．その理由は，Pvm00における反復1回の中の通信時間と計算時間の比率を表すFig.6を見ても明らかなように，計算粒度が大きくなっていないからである．計算粒度が大きくなっていないことは，この並列アプリケーションの情報交換の方法にその原因があることが考えられる．本研究での情報交換方式は，反復一回ごとに管理プロセスを介して全情報交換を行う．各プロセスは反復一回のうちで必ず1回の通信を行う．そこで，1回の通信を行うプロセスを1台の計算機に複数

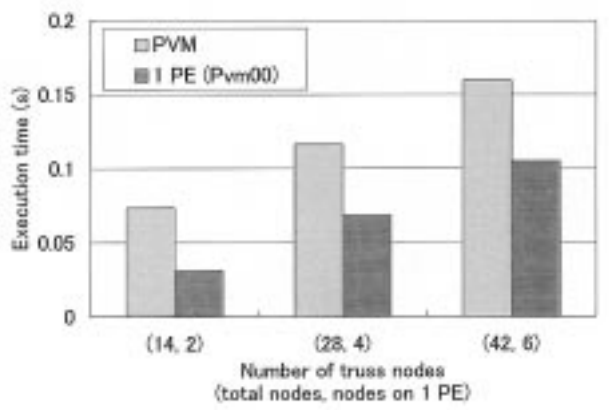

Fig. 5. Comparison of PVM (N nodes / N processes / 1 PE type algorithm) and Pvm00 (stand-alone): average time on each iteration

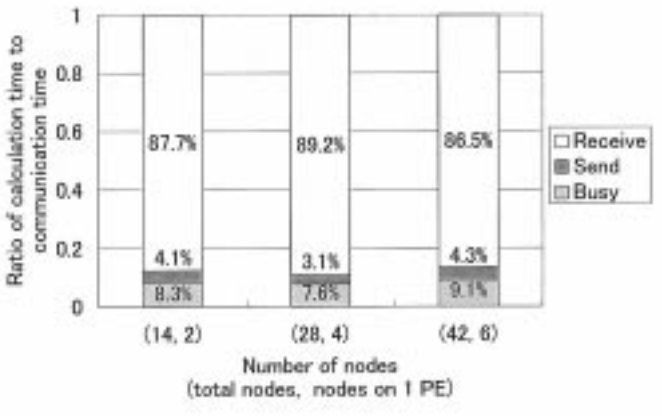

Fig. 6. Granularity of N nodes / N processes / 1 PE type algorithm on Pvm00

個実行させると，1台の計算機の通信も複数回起こることになる．そのため，計算時間の増加とともに通信回数の増加が起こり，1つの計算機上で複数のプロセスを実行する方法では，並列処理の効果が得られないのである．

4.2.3 Nノード/1プロセス/1PEの割り当て

次に，各計算機には1つのプロセスだけを実行させ，その1つのプロセスの中に複数の節点計算を割り当てられるようにした．Fig.1(c)のように実行する方法である．このとき，複数の節点の計算はおのおののプロセスの中で逐次的に処理される．そのためそれぞれのプロセスの処理時間が割り当てられた節点の数だけ大きくなる．そして，各計算プロセスに割り当てる節点数を2節点ずつ増加させ，問題規模を56節点まで大きくした．その時の結果をFig.7に示す．この図において，14節点トラス(計算機1台につき2節点の割り当て)の解析時には，並列処理の効果

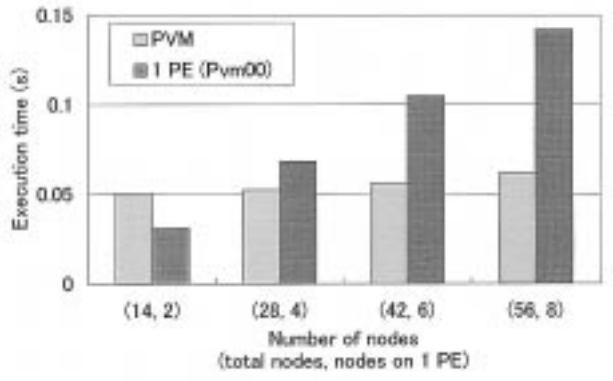

Fig. 7. Comparison of PVM (N nodes / 1 process / 1 PE type algorithm) and Pvm00 (stand-alone): average time on each iteration

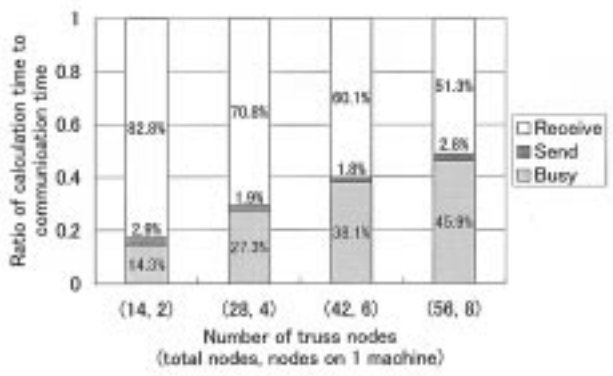

Fig. 8. Granularity of N nodes / 1 process / 1 PE type algorithm on Pvm00

は現れていないが，28節点(計算機1台につき4節点の割り当て)以降に関しては，問題規模の増加に伴って，並列処理による計算時間の短縮が見られる．Pvm00における計算粒度を示すFig.8を見ると，計算粒度も確実に大きくなってきていることがわかる．また，問題規模の増加に対する計算時間の増加については，1台の計算機における増加の割合に比べて，PVMによる計算は増加の割合が低い．そこで，この方法で粒度を非常に大規模な問題まで上げて行けば，有効な処理効率が得られることが期待できる．

4.3 マルチキャストとダイレクトリンクについて

PVMのような逐次計算機ネットワークを並列計算機化するソフトウェアには，そのソフトウェアそのものが複雑な処理をしている場合が多く，そのために処理時間が長くなることがある．そのような例を以下に示す．

マルチキャストとは，PVMで用いられている1対Nプロセス間通信を目的とする通信方法である．この方法はPVMに用意されているpvm_mcast()という通信関数によって実現される．マルチキャストは，指定された複数の宛先に一度にデータを送信するための関数である．前節までのプログラムでは，管理プロセスの送信時にこのマルチキャスト機能を使用していた．

あるプロセスでマルチキャストが要求されると，まず最初に送信したいメッセージがローカルのPVMデーモンに渡される．そのローカルデーモンは，送信先に指定されたプロセスを持つPVMデーモンを決定する．その後，それらのデーモンにデータを送信し，データを受け取ったそれらのデーモンが，そのデータを各々デーモンが管理するローカルプロセスに送信する[1)]．Fig.9にその様子を示す．

また，PVMシステムの通常の設定では，プロセス間通信はPVMデーモンを介して行われる．つまり，PVMデーモンがあるプロセスから受け取ったメッセージを他のプロセスにフォワーディングするオーバヘッドが存在しているということである．そこで，このオーバーヘッドをなくすためのダイレクトリンク機能というものがある．ダイレクトリンク機能は，ユーザプロセス同士をTCPソケットによって直接接続させ，デーモンのフォワーディングを無くすための機能である．なお，ダイレクトリンクを使用しているときでも，マルチキャストが指定されると通信はデーモン経由となり，デーモン同士のUDPソケットによる通信となる[1)]．

Fig.10は，上述のマルチキャストを用いず，またダイレクトリンクの指定を行ったアプリケーション

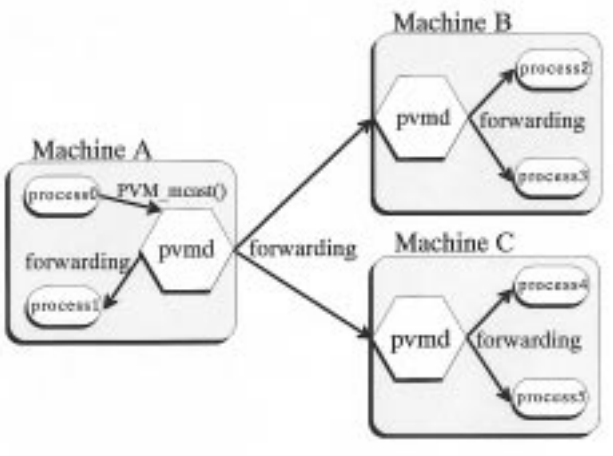

Fig. 9. Pvm multicast

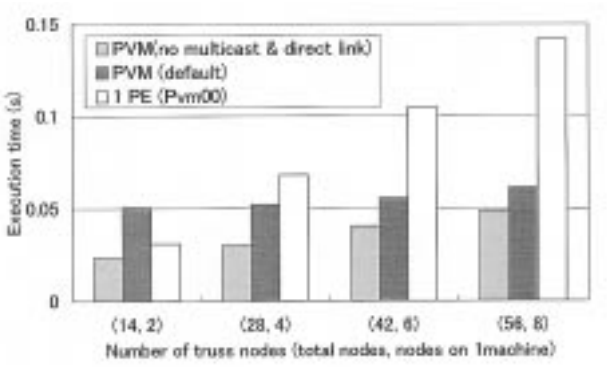

Fig. 10. Comparison of default configuration (using pvm_mcast()) and direct link configuration: average time on each iteration

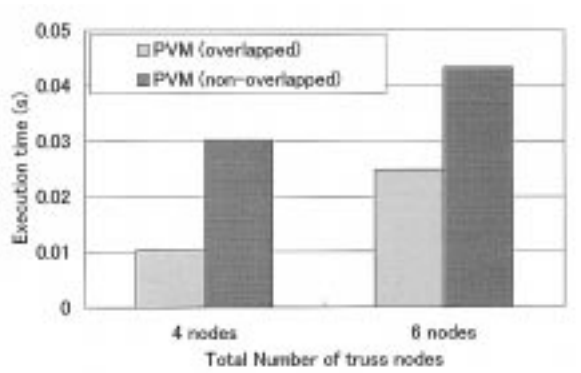

Fig. 11. Comparison of overlapped type algorithm and non-overlapped type algorithm: average time on each iteration

の性能を表す．通常のPVMアプリケーション(マルチキャスト使用)の実行時間よりも，短くなっていることがわかる．この通常の処理時間との差が，上述のような複雑な処理によるPVMそのもののオーバーヘッドであると考えられる．

4.4 通信と計算のオーバーラップ

これまでは，管理プロセスに通信を集中させるアルゴリズムについて，その性能を提示してきた．ここでは，今までと異なる通信アルゴリズムによる並列アプリケーションの性能について，考察する．

これまでのアルゴリズムは，全情報交換が一度に行われた場合に，通信路が混雑しないような方法として採用したものである．イーサネットが直列的通信方法のみを許していることから，並列アプリケーションそのものの通信方法も直列に行うようにしたものである．しかし，ここで複数のプロセスの全情報交換が，まったくの同時に起こるとは限らない可能性から，もう1つの通信アルゴリズムの効果を検討することができる．

Fig.11がそのもう１つのアルゴリズム(overlapped)と今までの，管理プロセス集中型(non-overlapped)の比較である．そのもう1つの方法は，多くの通信が複数のプロセスの計算とオーバーラップしている可能性があり，実際の通信回数通りのオーバヘッドが他のプロセスの計算によって隠蔽されている可能性がある．その具体的なアルゴリズムのタイムチャートをFig.12に示す．この図は，8つのプロセスがお互いの通信をオーバーラップさせて通信を行い，その結果全情報の交換が行われるという方法である．実際,

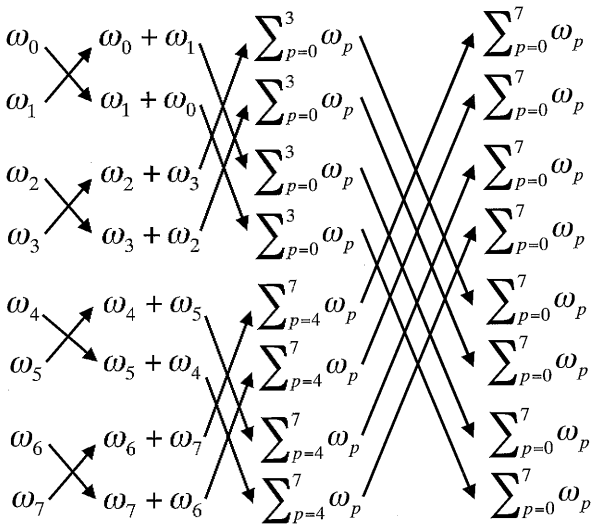

Fig. 12. Chart of an overlapped communication algorithm

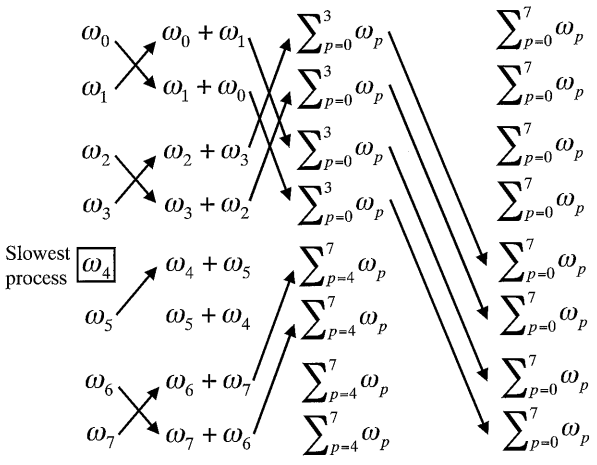

Fig. 13. Message sending which is finished during calculation of the slowest process

イーサネットの場合は，この様に通信同士が並列にオーバラップされることはなく，このような順序の通信が少しずつずれて，直列に処理されることとなる．そのため，全情報交換にかける総通信回数は24回となり，管理プロセスに集中させる14回の通信より10回分も通信回数が多い．しかし，実際にはこの方法でFig.11のように，オーバーラップ型のほうが計算時間が短い．その理由については，各マシンの性能が異なるために，一度に全てのプロセスが通信可能状態になるわけではなく，早い段階でその状態(送信可能状態)になったプロセスから通信が始まっているということが考えられる．そうすれば，遅いプロセスの計算時間に速いプロセス同士の通信が隠されてしまう．例えば，Fig.13のように4番目のプロセスが最も遅いプロセスで他のプロセスを待たせることになっている状況でも，図中の矢印の送信処理がその4番目のプロセスの計算処理時間内に終了する可能性がある．また，この通信と計算処理のオーバーラップについては，計算性能がヘテロな環境に限ったことではないとも考えられる．イーサネットの通信の場合，通信は直列に処理されるために，個々ののプロセスの計算もその分だけタイムラグをもった形で終了していく．そのタイムラグが，通信と計算のオーバーラップを実現させる可能性がないわけではない．しかしながら，これについては，現状のヘテロな環境では検討するすべもなく，今後の課題となる．

5. おわりに

本研究では，PVMを用いることによって既存の計算資源から，8プロセッサの仮想的な並列計算機を構築した．そして，その環境で並列アプリケーションを実行し，以下のことが明らかとなった．

1) 同期操作や通信のオーバーヘッド等による時間が，並列化されている問題の本質部分の計算処理に比べて大きいときは，並列処理による速度向上が得られない．
2) 本研究で用いた並列処理方法は，問題規模が大きいときに有効であることが期待できる．
3) オブジェクト指向計算モデルは，それぞれのオブジェクトが独立した単位なので，並列プロセスに割り当てることが比較的容易である．
4) プロセス数とプロセッサ数は同じ数であるほうが，処理効率が良い．異なる場合には，並列処理の効果が見られない場合がある．
5) クラスタコンピューティング用ソフトウェア(逐次計算機並列化ライブラリ)は，通信のために複雑な処理を行うため，処理に時間がかかる．そのためユーザは，並列アプリケーションのオーバヘッド以外の，並列化ライブラリそのもののオーバーヘッドも考慮して，アプリケーションを開発しなければならない．
6) 通信回数を少なくすることは，通信にオーバーヘッドの時間を小さくすることにはならない可能性がある．
7) 個々の並列処理要素(プロセス，プロセッサ)の通信回数が多くても，通信オーバーヘッドの時間が大きいとは限らない．

また今後の課題として，以下にいくつか挙げてみる．

1) ネットーワークの負荷による並列アプリケーションへの影響を解析する．
2) 均一な環境におけるPVMシステムの構築と，ヘテロな環境との比較．
3) 同期型処理であるが，全情報の交換を行うのではなく，関連するプロセスとのみ通信を行い通信回数を減らすアルゴリズム(局所データ同期型)の検討を行う．

参 考 文 献

1) G.A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam, “PVM: Parallel Virtual Machine: A Users’ Guide and Tutorial for Networked Parallel Computing”, The MIT Press(1994)
2) http://www.mcs.anl.gov/mpi/
3) 奥野　弘之，堀口　進, “ ワークステーションクラスタにおけるメッセージ通信性能の評価 ”，情報処理学会ハイパフォーマンスコンピューティング研究会報告

HPC68-4(1997)
4) http://www.epm.ornl.gov/pvm/performance.html
5) F.J.Marin and O.Trelles-Salazar and F.Sandoval, “Genetic Algorithms on LAN-Message Passing Architectures Using PVM:Application to Routing Problem”, Parallel problem solving from nature — PPSNIII : International Conference on Evolutionary Computation the Third Conference on Parallel Problem Solving from Nature, October 9-14(1994)
6) M. Miki and Y. Murotsu, “Object-Oriented Approach to Modelling and Analysis of Truss Structures”, AIAA Journal, Vol. 33, No. 2, pp. 348-354(1994)
7) G.A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek, V. Sunderam著, 村田 英昭訳, “ PVM3ユーザズガイド &リファレンスマニュアル 日本語版 ”(1995)
8) 矢川　元基, 吉村　忍, “ 計算力学とCAEシリーズ1 有限要素法 ”, 培風館(1991)
9) M. Miki and T. Koita, “Parallel Computing for Analysis of Variable Geometry Trusses”, AIAA Journal, Vol. 34, No. 7, pp. 1468-1473(1996)
10) 富田 眞治, 末吉 敏則, “ 並列処理マシン ”, オーム社(1989)