物理探査　第55巻第3号（2002）207-221頁　　BUTSURI-TANSA, Vol. 55 No. 3 (2002) pp. 207-221

重合前時間マイグレーションにおける並列処理に関する検討

松島　潤*・六川修一**
横田俊之*・大久保泰邦*

Parallel processing of prestack time migration

Jun Matsushima*, Shuichi Rokugawa,
Toshiyuki Yokota* and Yasukuni Okubo***

ABSTRACT

We have developed Kirchhoff summation prestack time migration (PSTM) code for parallel computer systems and evaluated a parallel implementation with two types of hardware architectures and also with two types of communication methods. The parallel implementation of an algorithm involves the division of total workload into a number of smaller tasks, which can be assigned to different processors and executed concurrently. Our PSTM algorithm uses data decomposition (to partition input data traces across processors) and the tasks communicate via MPI (Message Passing Interface) calls.

The algorithm is tested for four different numerical data sizes (11, 44, 171, 681 Mbyte). An impulse problem is used to demonstrate the computational performance on a parallel machine using up to 256 processors. We show that the efficiency and speed of the algorithm depends on the types of hardware architecture and data size. Its dependency is related to the share of communication time.

We also measure both computation and communication speed as a function of data size and based on these results derive the equation which predicts the total computation time. We show the availability of the equation for prediction of total computation time.

Key words: parallel processing, prestack time migration, MPI (Message Passing Interface)

1. 序　論

近年の計算機の進歩は目覚ましく，過去10年間でマイクロプロセッサの場合で数百倍，スーパーコンピュータで1000倍の性能向上が得られている（妹尾・左近，2000）。さらに先端的なスーパーコンピュータはベクトル型からベクトル・並列型へと移行し，現在では高性能計算機あるいはHPC（High-Performance Computer）と呼ばれその性能向上を続けている。その一方で，PCクラスタに代表されるような，安価な汎用チップを搭載したパソコンを複数台繋いで，安価な高性能計算機を構築することも行われている。

計算機に使用されるCPUの処理能力は上述のように年々大幅に進歩しているが，単一のCPUの処理速度向上に期待するのは限界がある。そこで互いに依存しない計算を複数のCPUに配分して同時に実行させることにより，単位時間当たりの計算量を向上させる並列計算処理が必要とされ，ネットワークの高速化・低価格化がそれを可能にしてきている。また，並列処理の利点は，このような計算時間の短縮効果ばかりでなく，大規模なメモリを必要とする計算を実施する場合には，並列処理することにより，1つのプロセッサ当たりに必要とされる

2001年9月13日原稿受付；2002年5月1日受理
Manuscript received September 13, 2001; Accepted May 1, 2002
* 産業技術総合研究所
〒305-8567 茨城県つくば市東1-1-1　中央第7
** 東京大学大学院工学系研究科
〒113-8656つ東京都文京区本郷7-3-1
* National Institute of Advanced Industrial Science and Technology Tsukuba central 7, 1-1-1, Higashi, Tsukuba, Ibaraki 305-8567, Japan
** Graduate School of Engineering, The University of Tokyo 7-3-1, Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

メモリ数を少なくできる点も大きな利点である。

反射法地震探査は計算機の発展とともに進歩してきた経緯があり（French, 1992），高性能計算機をいかに有効に利用していくかは，今後の反射法地震探査の展開にとって重要であると考えられる。特に，従来より行われてきた処理をより高速に実行できるという消極的な利用でなく，従来では現実的に行えなかった処理を実行可能にするような積極的な利用が望まれる。従来型のCMP重合法の適用においては単体のワークステーション上で実施するのが一般的であるが，ある程度規模以上のデータに対して重合前マイグレーションを適用する場合には，並列計算機あるいはPC（あるいはワークステーション）クラスタ等を用いた並列データ処理が現状では不可欠であると考えられる。

国外では，並列計算機を用いた重合前マイグレーションが盛んに行われおり，大規模な並列処理としては，1824個のプロセッサを搭載した超並列計算機を用いて46.9ギガバイトの3次元海上データをキルヒホッフ型の3次元重合前深度マイグレーション処理した例がある（Chang et al., 1998）。また国内においては，並列計算機を用いた空間周波数領域の差分法型の2次元深度マイグレーション（佐藤ほか，1995）やワークステーションクラスタを用いたキルヒホッフ型の3次元時間マイグレーション（中島ほか，1997）の数例があるのみである。

本研究では，松島ほか（2001）が提案した重合前時間マイグレーション手法をMPI（Message Passing Interface）と呼ばれる通信ライブラリを用いたメッセージパッシング方式を採用して並列計算機に実装した。MPIは，現在業界標準であり幅広く使用されている通信ライブラリである（e.g., Gropp et al., 2000）。なお並列処理する際に，2種類のハードウェアアーキテクチャ（分散メモリ型と階層メモリ型）と2種類のプロセッサ間通信方法（グループ通信と1対1通信）を組み合わせて4種類の並列処理を比較する。

Amdahl（1967）によれば，N台のプロセッサを用いて並列処理を実行した場合に得られる速度向上は，N/(Ns+1−s）倍である（sは全体の処理における逐次処理の割合であり，0<=s<=1）。しかし，この法則には通信のオーバーヘッドが考慮されていないため，実際の速度向上を予想するには現実的な方法ではない。古村ほか（2000）は，音響波動場計算の並列処理において演算速度と通信速度を用いて並列化効率を論じている。本研究でも，演算速度と通信速度の2つのパラメータに着目し，使用する並列計算機の演算性能ならびに通信性能を予め評価し，それらの基本性能を基にして，PSTMの並列処理時間を推定する試みも行う。

2. 並列計算機アーキテクチャとプログラミングモデル

並列計算機はメモリモデルの違いにより，下記のように大きく3つに分類される。

- 共有メモリ型：複数のプロセッサがメモリバス/スイッチを介して，主記憶を共有する形態であり，SMP（Symmetrical Multiple Processors）と呼ばれている。プロセッサ間での通信を必要としないため，プログラム開発の容易性が利点である。
- 分散メモリ型：個々のプロセッサが独自の主記憶を有するシステム構成をしており，そのシステム同士がネットワークにより接続されている形態である。大規模なシステム構築が可能であるという利点があるものの，プロセッサ間通信をプログラム内において明示的に記述する必要があり，プログラム開発が一般的に困難である。
- 階層メモリ型：複数の共有メモリシステムをネットワークで接続する形態であり，分散メモリシステムと共有メモリシステムを組み合わせた階層システムである。

本研究で使用する並列計算機は，産業技術総合研究所先端情報計算センター（以下，TACCと呼ぶ）所有の並列ベクトル計算機HITACHI SR8000（以下，SR8Kと呼ぶ）であり，上記分類では階層メモリ型に分類される。SR8Kの基本スペックは，総合演算性能512ギガフロップス，メモリ512ギガバイトであり，プロセッサは250 MHzのPowerPCアーキテクチャを元に擬似ベクトル機構，拡張レジスタ，拡張命令等を追加したプロセッサを用いている。またHI-UX/MPPと呼ばれるUNIX系のOSが搭載されている。現在TACCに導入されているSR8Kは，Fig. 1に示すように，64個のノードから構成されており，各ノード間は2次元クロスバーネットワーク（日立社製独自規格）と呼ばれる高速なネットワークで結合されている（ピーク通信性能：1

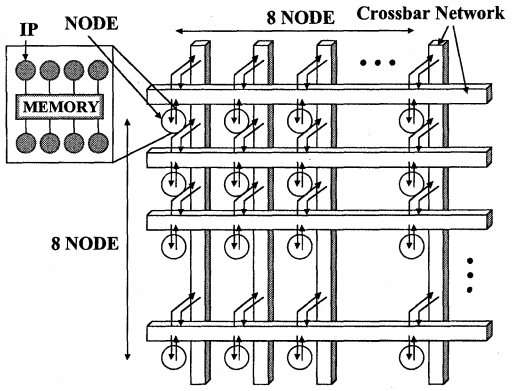

Fig. 1 Hardware configuration of the SR8K which is a parallel system with distributed memory consisting of 64 nodes. A node consists of 8 microprocessors that have shared memory. Data is transferred between each node via the two-dimensional crossbar network.

ギガバイト/秒）。さらに各ノード内には8個のプロセッサ（Fig. 1ではIPと表示している）が搭載され，8ギガバイトの主記憶を共有しており，プロセッサあたりの演算性能は1ギガフロップスで，ノードあたりの総合演算性能は8ギガフロップスである（これらの値は実測ではなく，ハードウェアから規定される値である）。

以上の性質により，ノードを単位としてシステムは分散メモリマシン，個々のプロセッサを単位としてノードは共有メモリマシンである。また，各ノード内（主記憶8ギガバイト）に存在する8個のプロセッサを独立に使用できる（個々のプロセッサは主記憶1ギガバイトを有する）分散メモリシステムも可能であり，この場合は最高512個のプロセッサを使用した並列計算が可能になる。なおこの場合のプロセッサ間通信性能はノード間通信性能と同様にピーク通信性能1ギガバイト/秒である。

このようなSR8Kの計算機アーキテクチャの特質により，本研究では以下の2種類のハードウェアアーキテクチャに基づいた並列処理を適用する。

(1) ノード内プロセッサを単位とした分散メモリアーキテクチャ

SR8Kは標準的にはノードを単位とした分散メモリシステムであるが，ノード内プロセッサを単位とした分散メモリシステムとして使用し，プロセッサ同士はメッセージ交換ライブラリを用いて通信を行う。前述したように，TACCに導入されているSR8Kでは最高512個の独立したプロセッサを使用できるが，本研究では256個を最大として使用する。

(2) ノード内共有メモリ並列・ノード間分散メモリ並列による階層的なアーキテクチャ

ノードを単位とした分散メモリシステムとして並列化し，ノード間はメッセージ交換ライブラリを用いて通信を行う。またノード内においてはコンパイラによる8個のプロセッサの並列化（以降ではこれをノード内自動並列と呼ぶ）を行い，階層的な並列処理を実施する。前述したように，TACCに導入されているSR8Kでは最高64ノードを使用できるが，本研究では32ノードを最大使用とする。

並列化のためのプログラミングモデルとしては一般的にSPMD（Single Program Multiple Data）とMPMD（Multiple Program Multiple Data）の2種類がある。本研究ではSPMDモデルを採用した。SPMDモデルにおいては，1つの実行モジュールは各プロセッサにロードされ，各プロセッサに応じた振る舞いをするように命令されている。これに対して，MPMDモデルでは異なるロードモジュールが各プロセッサにロードされ実行される。MPMDモデルは一般的には，異なる性質の複数の処理を扱うプログラムの並列化に用いられる。

3. PSTMの並列アルゴリズム

本研究で用いるPSTMのデータ処理の概念と手順は松島ほか（2001）で述べているので詳細については省略する。基本的にはこの手法はキルヒホッフ型PSTMに分類される。キルヒホッフ型重合前マイグレーションは各種重合前マイグレーションの中でも並列化効率が非常に良いとされている（Epili and McMechan, 1996）。Zhang（1997）は16個のプロセッサを用いてキルヒホッフ型の3次元重合前深度マイグレーションを実行させた結果，並列化効率は95%以上だったことを報告している。

キルヒホッフ型マイグレーションにおける並列化の方法としては，2つの方法が考えられる。1つはイメージ領域（最終出力断面）を分割する方法であり，もう1つはデータ領域（入力データ）を分割する方法である（Epili and McMechan, 1996）。本研究ではデータ領域を分割する方法を採用した。データ分割法の利点は，大きなデータを処理する際にデータを分割することにより，1つのプロセッサあたりに必要とされるメモリを少なくできる点である。以下にデータ分割法を採用する際のデータ処理の概略を説明する。

地下のある一点を松島ほか（2001）の方法に基づいたPSTMによりイメージすることは，その地点において励起される散乱波の振幅を加算することに相当する。この作業は以下の(1)式で行われる。

$$Image\_Value=\sum_{i=1}^{S}\sum_{j=1}^{R}\frac{Amp(i,j,t_{ij})}{S\cdot R} \qquad (1)$$

ここで$Amp(i, j, t_{ij})$はi番目の震源で発震しj番目の受振器で観測されるトレースの時刻t_{ij}における振幅値である。また，SとRはそれぞれ発震点および受振点総数であり，全入力トレース数はS*R本となる。この入力トレースに対して，散乱波パターンに沿って(1)式に基づいた加算を行い，加算された振幅を重合数（この場合は$S*R$）で割ることにより正規化を行う。(1)式に基づいた操作は$S*R$回の加算を実行することに相当し，この操作をn個のプロセッサを用いて並列的に処理するには以下のように考える。$S*R$本の入力全トレースをn個のプロセッサにできるだけ均等になるように分配し，それぞれのプロセッサにおいて部分和を計算する。それぞれの部分和はプロセッサ間通信することによりそれらの総和を求める。

以上のことについてFig. 2を用いて具体的な並列処理の流れを述べる。Fig. 2の中央の図には，処理全体の流れの概略を示しており，その左側にはデータ入力に関する説明図，右側にはイメージングに関する説明図を示している。データ入力については，全データをできるだけ均等に分割し，分割したそれぞれのデータに対して，担当するプロセッサを割り当てる。各々のプロセッサは

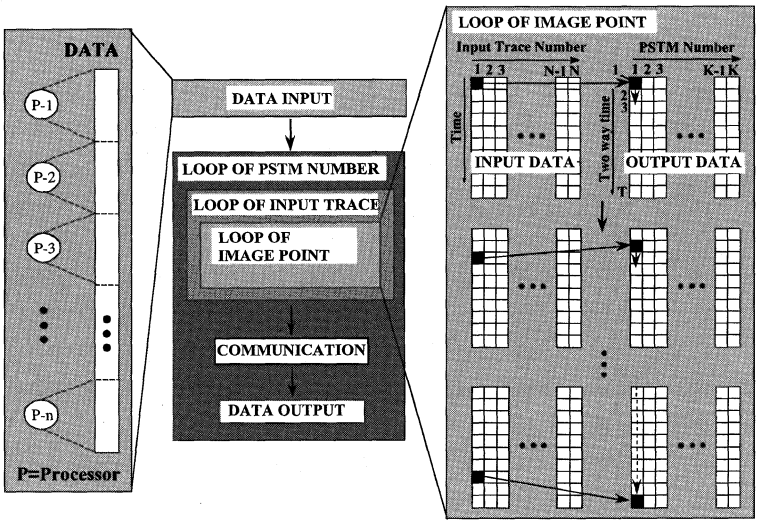

Fig. 2 Flow chart of parallel processing of PSTM (Pre-Stack Time Migration). Left figure shows the details of "DATA INPUT". Right figure shows the details of "LOOP OF IMAGE POINT".

割り当てられたデータ部分のみを読み込む。解析部分では，各々のプロセッサについて以下の手順で実施する。

手順1：PSTMトレースを作成する位置を指定する。例として，Fig. 2の右側の図（「LOOP OF IMAGE POINT」と表示された枠内）に「OUTPUT DATA」と示されたデータ領域において，PSTM番号が1～Kまで表示されている。この番号を1に指定する。この作業は繰り返しのループになっており，Fig. 2の中央の図において「LOOP OF PSTM NUMBER」と示されている。

手順2：PSTMトレースを作成するための入力トレースを指定する。例として，Fig. 2の右側の図に「INPUT DATA」と示されたデータ領域において，入力トレース番号を1に指定する。この作業は繰り返しのループになっており，Fig. 2の中央の図において「LOOP OF INPUT TRACE」と示されている。コンパイラによるノード内自動並列を実施する場合はこのループを並列処理する（全ループを分割し，8個のプロセッサにそれぞれ分担させる）。

手順3：手順1で指定したPSTM番号（Fig. 2の場合は1）において，垂直往復走時方向の位置を指定する。例として，Fig. 2の右側の図の「OUTPUT DATA」と示されたデータ領域において，PSTM番号が1であり，なおかつ「Two way time」と示された方向において，番号1を指定する。この点において手順2で指定した入力トレースがイメージングに寄与するデータを取りだし，この点に加算する。この作業は繰り返しのループになっており，Fig. 2の中央の図において「LOOP OF IMAGE POINT」と示されている。

手順4：手順3で指定した垂直往復走時方向の位置番号を1つ進めて，位置番号がTになるまで繰り返す。

手順5：手順2で指定した入力トレース番号を1つ進めて，入力トレース番号がNになるまで手順3と手順4を繰り返していく。

手順6：手順5までの作業を行うことで，手順1で指定した位置でのPSTMトレースが1本作成される。しかし，これは各々のプロセッサが割り当てられたデータに対してのみ行った処理結果であるので，すべてのプロセッサで得られた結果を通信することによってすべて加算する（この値をいま*Amp*とする）。なお，このときに各イメージ点での重合数も一緒に送信して，その総和を計算し（この値をいま*Stack_num*とする），*Amp*/*Stack_num*を実行することにより正規化する。この正規化された値を出力ファイルに書き出す。

手順7：手順1で指定したPSTMトレースを作成する位置番号を1つ進めて，位置番号がKになるまで手順2，手順3，手順4，手順5，手順6を繰り返していく。

4. プロセッサ間通信

プロセッサ間通信に関して，本研究ではメッセージ通信ライブラリとしてMPIを採用した。MPIには様々な通信関数が用意されているが，本研究では以下に述べる2つの通信方法を採用する。

(1) 演算付グループ通信

この通信の特徴は，プロセッサ間通信の同期を自動的に行えることにより，ユーザが明示的にプロセス間通信の同期を取る必要がない点である。Fig. 3において，3つのプロセッサ（それぞれプロセッサ1，プロセッサ2，プロセッサ3と呼ぶ）を用いて並列処理を行うことを考える。Fig. 3において，3つのプロセッサの中でプロセッサ2が最も早く処理を終えているが，他の2つのプロセッサは処理を終えていない。このときプロセッサ2は待ち状態になる。そして，そのうちにプロセッサ1が処理を終えて待ち状態になり，最後にプロセッ

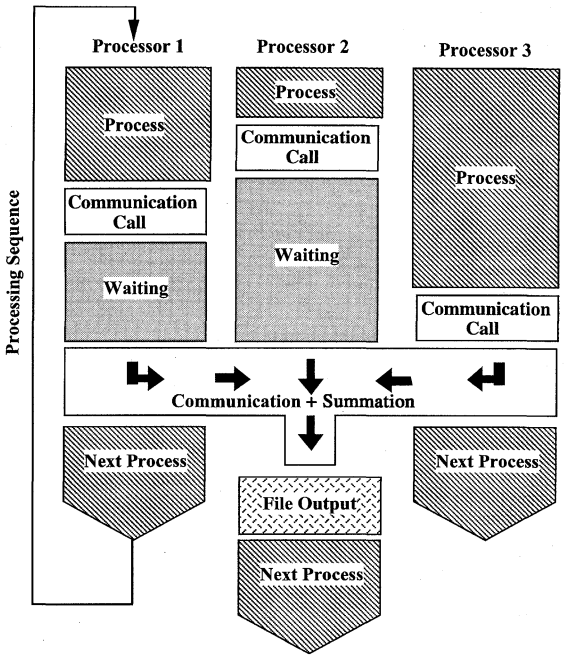

Fig. 3 Illustration of a group communication. A group communication is executed by having all processes in the group call. This call provides a global reduce operation such as summation across all the members of a group.

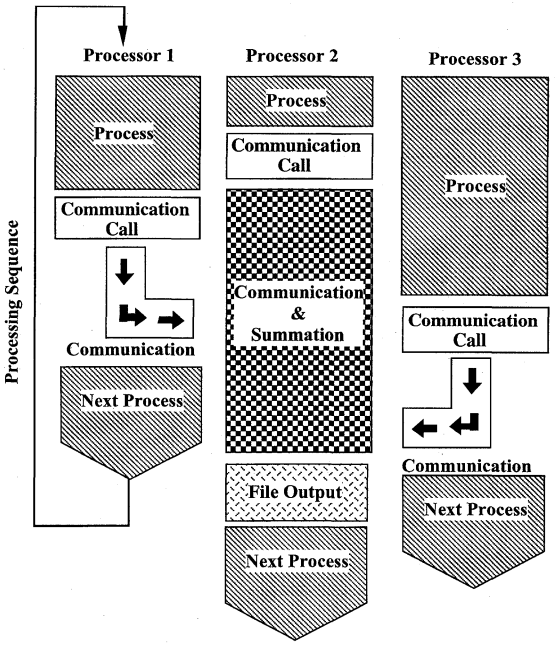

Fig. 4 Illustration of a point-to-point communication. One processor sends a message to another processor using the send operation function. Another processor receives this message with the receive operation function.

サ3が処理を終えることにより，通信を行うことができる。また，通信されているデータに対する演算も行うことができ，Fig. 3の場合には通信されているデータの総和を求め，その結果はプロセッサ2に送信され，プロセッサ2によりファイルに書きだされる。プロセッサの不要な待ち状態をなくすためには，各プロセッサ間の処理負荷を均等にすることが重要である。

(2) 一対一通信

この通信は，2つのプロセッサ間での通信を行い，明示的に送信側と受信側を設定する必要がある。この通信について，Fig. 4を用いて説明する。前述のように3つのプロセッサを用いて並列処理を行うことを考える。この場合，各プロセッサで計算された結果を1つのプロセッサに集中させる。すなわち，他のプロセッサから送信されてくるデータに対して，受信側プロセッサはその結果の総和を計算しファイルに書きだす操作を行う。Fig. 4において，プロセッサ1とプロセッサ3は送信側でプロセッサ2が受信側である。プロセッサ1とプロセッサ3では，担当した処理が終わると計算結果をプロセッサ2に送信し，送信が完了すると，次の処理に移行する。一方，プロセッサ2では，送信されてきた結果の総和を計算し，ファイルに書きだす作業を行う。このように，プロセッサ2において，送信されてくる結果の総和をとり，ファイルに書きだす作業があるので，処理量に関して他のプロセッサより負荷を軽減させておく。処理時間を短縮するためには負荷率の設定を最適化する必要がある。負荷率αの最適化については後述する。

以下では，演算付グループ通信と一対一通信を利用する場合において，上述の説明のうち通信部分のみの性能を評価する。前述したようにTACCのSR8Kのピーク通信性能は1ギガバイト/秒であるが，実際の通信性能は送信データ規模やプロセッサ数に依存する。前述の2種類の通信方法と前述の2種類のハードウェアアーキテクチャを組み合わせることにより，以下の4種類について調べた結果をFig. 5(a)～(d)にそれぞれ示す。

(a) ノード内プロセッサを単位とした分散メモリ並列化＋演算付グループ通信

(b) ノード内自動並列・ノード間分散メモリ並列による階層的な並列化＋演算付グループ通信

(c) ノード内プロセッサを単位とした分散メモリ並列化＋一対一通信

(d) ノード内自動並列・ノード間分散メモリ並列による階層的な並列化＋一対一通信

Fig. 5(a)～(d)において横軸は通信に用いたデータサイズ（単位：バイト）を示し，縦軸は通信のスループット（単位：メガバイト/秒）を示す。また，プロセッサ間通信の場合（上記の(a)ならびに(c)）は，プロセッサ数を8, 32, 64, 128, 256のそれぞれについて測定し，ノード間通信の場合（上記の(b)ならびに(d)）はノード数を4, 8, 16, 32のそれぞれについて測定した。なお，測定

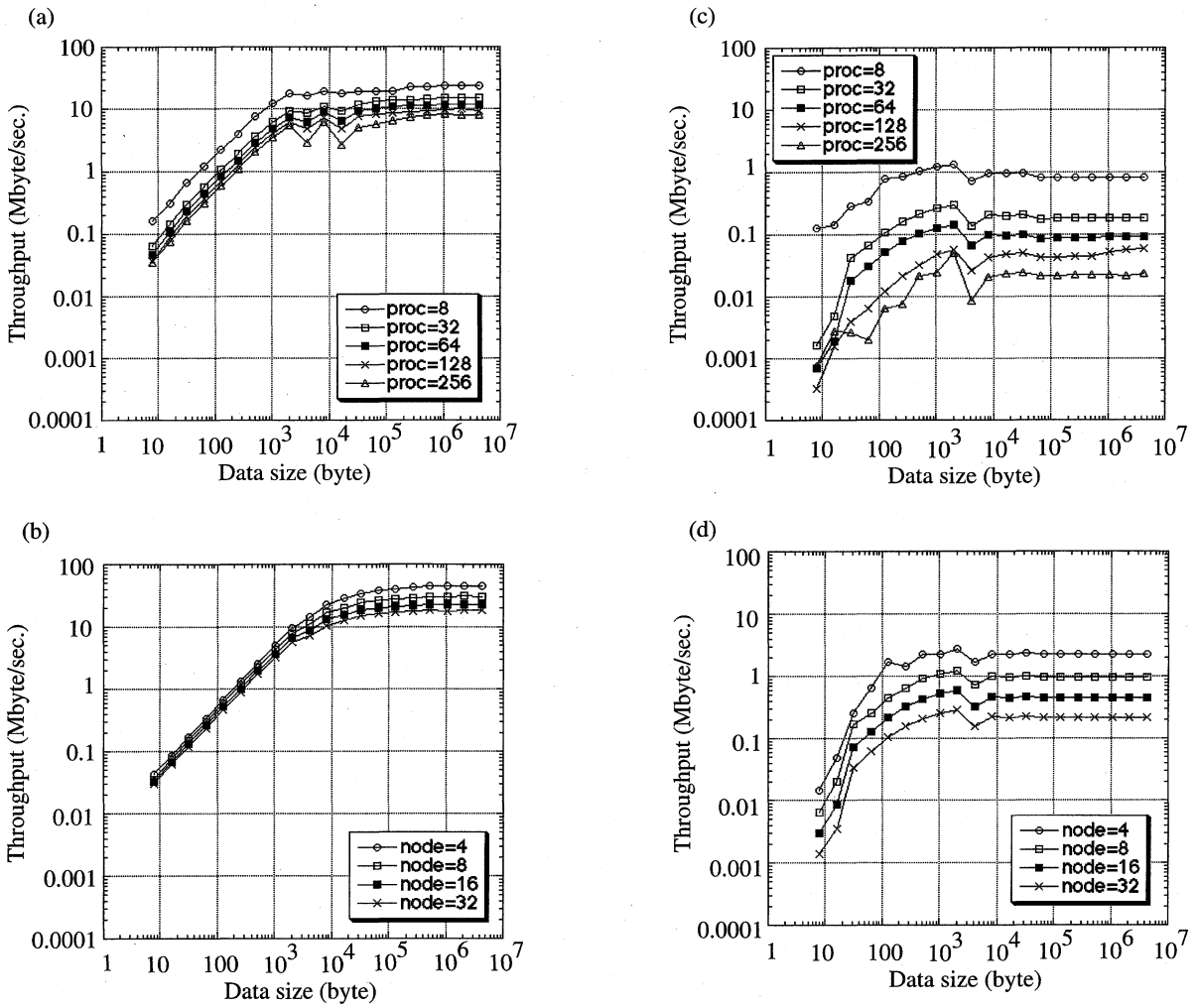

Fig. 5 Throughput of both a group and point-to-point communication with two different hardware architecture types. (a) group communication among processors based on distributed memory type, (b) group communication among nodes based on distributed memory type, (c) point-to-point communication between processors based on distributed memory type, (d) point-to-point communication between nodes based on distributed memory type.

方法は，それぞれのデータ通信を行うMPI関数の前後に時間測定関数を設置し，そのMPI関数に要した時間を100回測定し，その平均を算出した。また，一対一通信の場合においては，送信側と受信側で計測された時間すべて加算しその平均を算出する方法によった。

全体的な傾向として，データサイズが大きくなるにつれて通信性能は向上し，データサイズがある規模以上になるとスループットは一定に近づく。これは，通信を行うごとに一定の立ち上がり時間がかかるため，通信するデータ規模が小さい場合はこの立ち上がり時間の影響を受けることによる。

またプロセッサ数あるいはノード数が増えると通信性能が低下し，その低下の度合いはグループ通信より一対一通信の方が大きいことがわかる。また，グループ通信は一対一通信に比べてスループットが良いことがわかる。特に一対一通信の場合には，1つのプロセッサにデータを集中させているので，通信の競合が多く起こることにより通信性能が低下すると考えられる。ただし，上記はあくまでSR8Kのネットワークにより得られた性能であり，ネットワークトポロジ等の違いに通信性能は依存することに注意されたい。

5. 数値実験データを用いた並列性能評価

ここでは，前節までに示してきた並列手法に関して，その並列性能評価を数値実験データを用いて行う。また，最後に現存の標準的なパーソナルコンピュータとの処理時間の比較も行う。

5.1 数値実験データ

数値実験に用いるモデルならびに発震・受振点展開をFig. 6に示す。均質速度（3000 m/s）の媒体において深度500(m) の地点に散乱点が存在するモデルである。Fig. 6に示すように展開長600(m) において発震・受振点を展開する。なお，発震・受振点は同一地点に設定

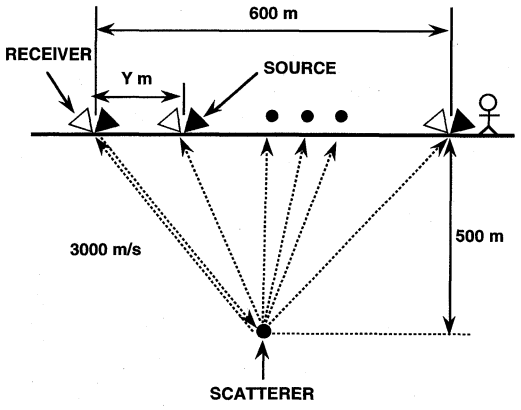

Fig. 6　Numerical model for seismic survey and the specifications of data acquisition. Point scatterer is placed at a depth of 500 m in a medium with constant velocity of 3000 m/s. The spacing and the number of sources and receivers are variable. Scattered waves were produced by the convolution method.

し，その距離間隔を変数Y(m) とする。ここでは距離間隔変数Yを12, 6, 3, 1.5(m) の4種類を設定し，それぞれをMODEL1～MODEL4と呼ぶ。この場合，発震・受振点数はそれぞれ51発震・51受振，101発震・101受振，201発震・201受振，401発震・401受振となることによりデータ規模を変化させる。

数値記録は以下に述べるように散乱波走時にリッカー波形をコンボリューションする方法により作成した。発震点，散乱点，受振点の幾何的配置で決定される波線の距離を媒質速度で除算することにより散乱波走時を求める。その走時に相当する箇所にインパルスを立て，最後にリッカー波形(中心周波数25 Hz)をコンボリューションする。時間サンプリング間隔は0.6 msec，サンプリング数は1000個である。1サンプルのデータサイズは4バイトである（ただし擬似ベクトル機能が有効に働くために，プログラムの中では8バイトに変換する)。以上の仕様におけるデータ規模としては，51発震・51受振の場合で約11メガバイト，101発震・101受振の場合で約44メガバイト，201発震・201受振の場合で約171メガバイト，401発震・401受振の場合で約681メガバイトであり，それぞれData 1～Data 4と呼ぶ。このような数値記録に対して，媒質速度（3000 m/s）を既知として，松島ほか（2001）の方法に基づいたPSTM処理を実施する。イメージングを行うトレース間隔は6(m) とし，PSTMトレース作成位置は展開長600(m) において合計101地点となる。なお，以上の仕様でPSTM処理を行う場合，プロセッサ間通信量は一定で，プロセッサ内でのデータ処理部分が大小変化する状況であることに注意されたい。

5.2　並列性能評価結果

前述した2種類のハードウェアアーキテクチャと2種類の通信関数を組み合わせることにより，以下の4種類の並列処理に関して性能評価を行った。

ケース1：ノード内プロセッサを単位とした分散メモリ並列化＋演算付グループ通信

この場合は1つのノードに搭載されている8個のプロセッサを独立させて処理を行い，演算付グループ通信により処理結果を求める。プロセッサを8, 32, 64, 128, 256個使用して上述の4種類のデータ（Data 1～Data 4）に対してPSTMを実施したときの処理時間をFig. 7(a)に示す。なお，ここでの処理時間はユーザCPU時間とシステムCPU時間との合計とする。Fig. 7(a)のそれぞれのグラフにおいて横軸はプロセッサ数で縦軸は処理時間を示す。Fig. 7(a)においてプロセッサ数増加に応じた処理時間の速度向上率を求めた結果をFig. 7(b)に示す。なおFig. 7(b)において，8個のプロセッサを使用した場合を1とした正規化を行った。Fig. 7(b)より，データ規模が小さい場合においては，プロセッサ数を増やしても処理時間はそれほど向上しないが，データ規模が大きくなるにしたがってプロセッサ数の増加率に対する処理時間の速度向上率が良くなっていることがわかる。データ規模が小さい場合において，処理速度が向上しない理由は，後述するように，処理全体における通信時間の占める割合が大きくなるためである。

上記の処理時間には計算部分に費やされる時間と通信部分に費やされる時間に分けられ，処理時間全体に占める通信時間の割合を求めた結果をFig. 7(c)に示す。規模の小さいデータを多くのプロセッサで処理すると，通信時間が占める割合が大きくなっていることがわかる。なお，通信時間の算出においてはプロファイラを使用することにより，MPI通信ライブラリ関数が費やした時間をすべて合計することにより求めている。

また，Fig. 7(a)の結果を得る際における浮動小数点演算性能を評価した結果をFig. 7(d)に示す。Fig. 7(d)において，横軸はプロセッサ数，縦軸は1ノード当たりの浮動小数点演算性能を示す。なお1ノード当たりの浮動小数点演算性能の算出には，全浮動小数点演算数を実行時間で割り算し，さらに使用するノード数（8個のプロセッサを1ノードに換算する）で割り算することにより求めた。

ケース2：ノード内自動並列・ノード間分散メモリ並列による階層的な並列化＋演算付グループ通信

この場合では以下のような階層的な並列処理を行う。1つのノードに搭載されている8個のプロセッサを共有メモリ計算機として，コンパイラによるノード内並列処理を行い，それぞれのノード間で得られた結果を演算付グループ通信を利用してノード間通信することにより処理結果を求める。ノードを1, 4, 8, 16, 32個使用して上述の4種類のデータ（Data 1～Data 4）に対して

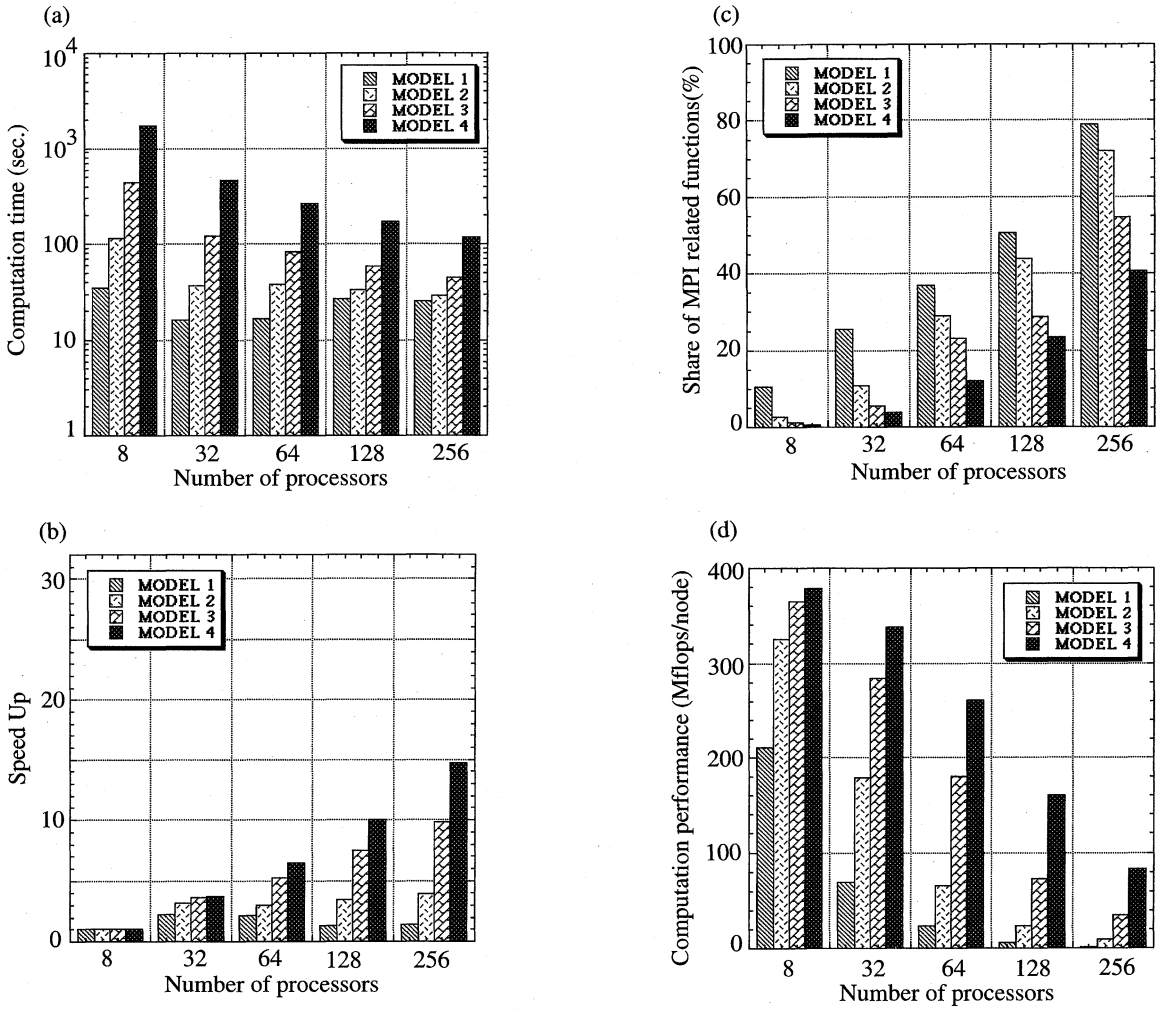

Fig. 7 (a) Measured computation time of the PSTM parallel processing as a function of the number of processors and data size in the case of distributed memory type with group communication, (b) Speed-up of the PSTM parallel processing as a function of the number of processors and data size in the case of (a), (c) Share rate of MPI related functions as a function of the number of processors and data size in the case of (a), (d) Computation performance as a function of the number of processors and data size in the case of (a).

PSTMを実施したときの処理時間をFig. 8(a)に示す。Fig. 8(a)のそれぞれのグラフにおいて横軸はノード数で縦軸は処理時間を示す。Fig. 8(a)においてノード数に応じた処理時間の速度向上率を求めた結果をFig. 8(b)に示す。Fig. 8(b)より，データ規模が小さい場合においては，ノード数を増やしても処理時間はそれほど向上しないが，データ規模が大きくなるにしたがってプロセッサ数の増加率に対して処理時間の向上率が良くなっていることがわかる。ケース1の場合に従って，処理時間全体に占める通信時間の割合を求めた結果をFig. 8(c)に示す。Fig. 8(c)より，規模の小さいデータを多くのプロセッサで処理すると，通信時間の占める割合が大きくなっていることがわかる。ケース1の場合と比べると，全体的に通信時間の占める割合が小さくなっていることがわかる。また，浮動小数点演算性能を評価した結果をFig. 8(d)に示す。

ケース3：ノード内プロセッサを単位とした分散メモリ並列化＋一対一通信

この場合はケース1の場合と同じハードウェアアーキテクチャを使用するが，通信方法として一対一通信ライブラリを使用する。前述したようにそれぞれのプロセッサで処理された結果は，1つのプロセッサに集中させ，そのプロセッサにおいて送信されてくる結果の総和をとり，ファイルに書きだす。この作業のために，他のプロセッサからデータを受信するこのプロセッサには処理量に関して，他のプロセッサのα%の負荷量を課す。負荷率αの最適化については後述する。

プロセッサを8, 32, 64, 128, 256個使用して上述の4種類のデータ（Data 1～Data 4）に対してPSTMを実施したときの処理時間をFig. 9(a)に示す。Fig. 9(a)のそれぞれのグラフにおいて横軸はプロセッサ数で縦軸は処理時間を示す。処理時間の速度向上率を求めた結果を

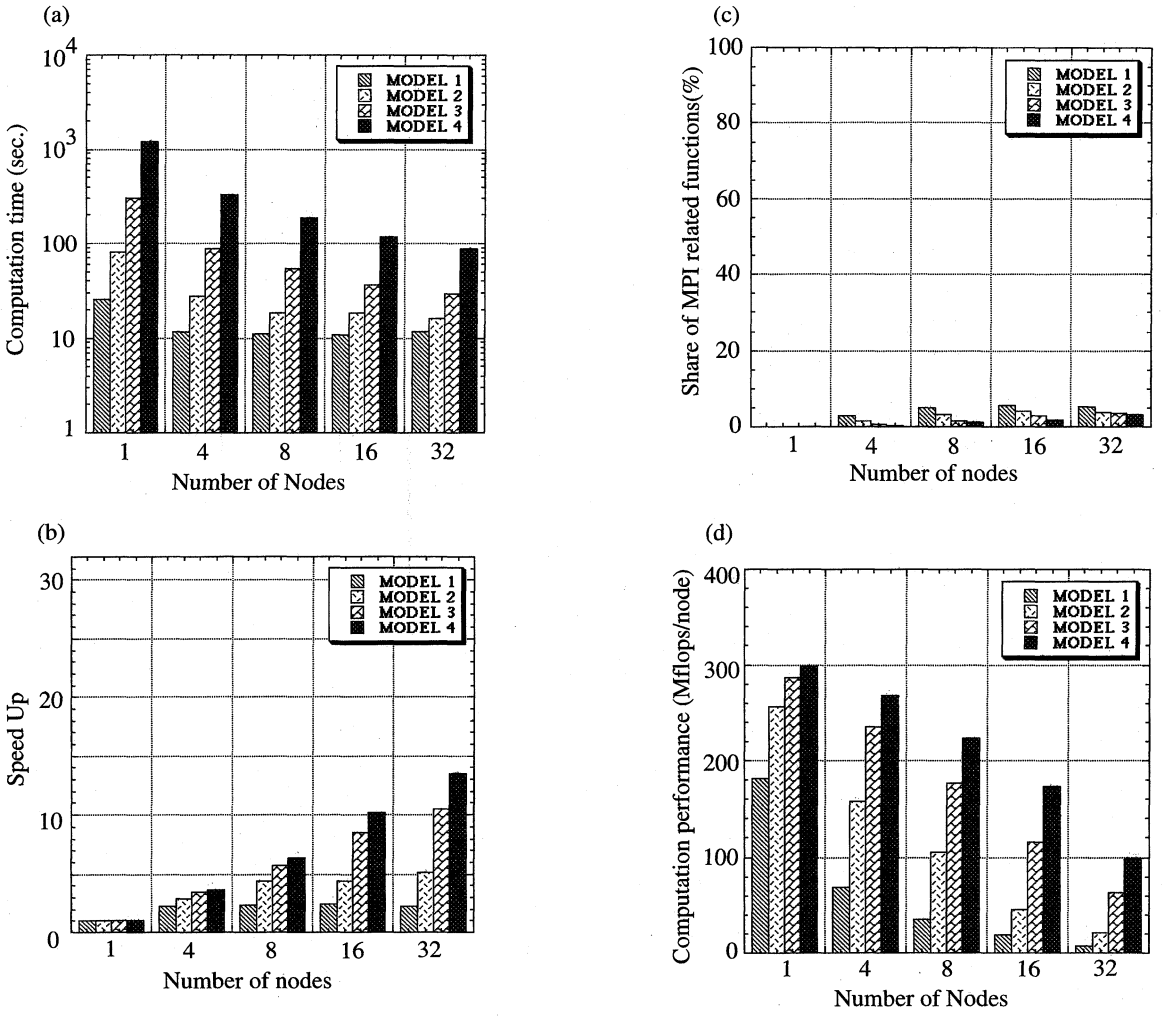

Fig. 8 (a) Measured computation time of the PSTM parallel processing as a function of the number of nodes and data size in the case of shared/distributed memory type with group communication, (b) Speed-up of the PSTM parallel processing as a function of the number of nodes and data size in the case of (a), (c) Share rate of MPI related functions as a function of the number of nodes and data size in the case of (a), (d) Computation performance as a function of the number of nodes and data size in the case of (a).

Fig. 9(b)に示す。Fig. 9(b)において，ケース1の場合と同様に，8個のプロセッサを使用した場合を基準とする（すなわちこの場合を1個のプロセッサに相当すると考える）。前述と同様に，処理時間全体に占める通信時間の割合を求めた結果をFig. 9(c)に示す。Fig. 9(c)とFig. 7(c)とを比較すると，通信占有率はどちらの通信方法でも，大差がないことがわかる。しかし，この場合に相当する通信性能の比較（Fig. 5(a)とFig. 5(c)）においては，1桁以上グループ通信の方が通信性能の点で優位であった点と合致しない。すなわち，Fig. 5(a)のグループ通信性能は過大評価であったことを示唆する。演算部分を含まず，グループ通信性能のみを評価する場合に過大評価になってしまう原因の究明は今後の課題としたい。また，浮動小数点演算性能を評価した結果をFig. 9(d)に示す。

ケース4：ノード内自動並列・ノード間分散メモリ並列による階層的な並列化＋一対一通信

この場合はケース2の場合と同じハードウェアアーキテクチャを使用し，コンパイラによるノード内並列処理を行うが，ノード間の通信方法としてケース3と同様に一対一通信ライブラリを使用する。ノードを1，4，8，16，32個使用して上述の4種類のデータ（Data 1～Data 4）に対してPSTMを実施したときの処理時間をFig. 10(a)に示す。Fig. 10(a)のそれぞれのグラフにおいて横軸はノード数で縦軸は処理時間を示す。ノード数に応じた処理時間の速度向上率を求めた結果をFig. 10(b)に示す。前述と同様に，処理時間全体に占める通信時間の割合を求めた結果をFig. 10(c)に示す。Fig. 10(c)とFig. 8(c)とを比較すると，通信占有率はどちらの通信方法でも，大差がないことがわかり，前述と同様にFig. 5(b)に示したグループ通信性能は過大評価であったことを示唆する。また，浮動小数点演算性能を評価し

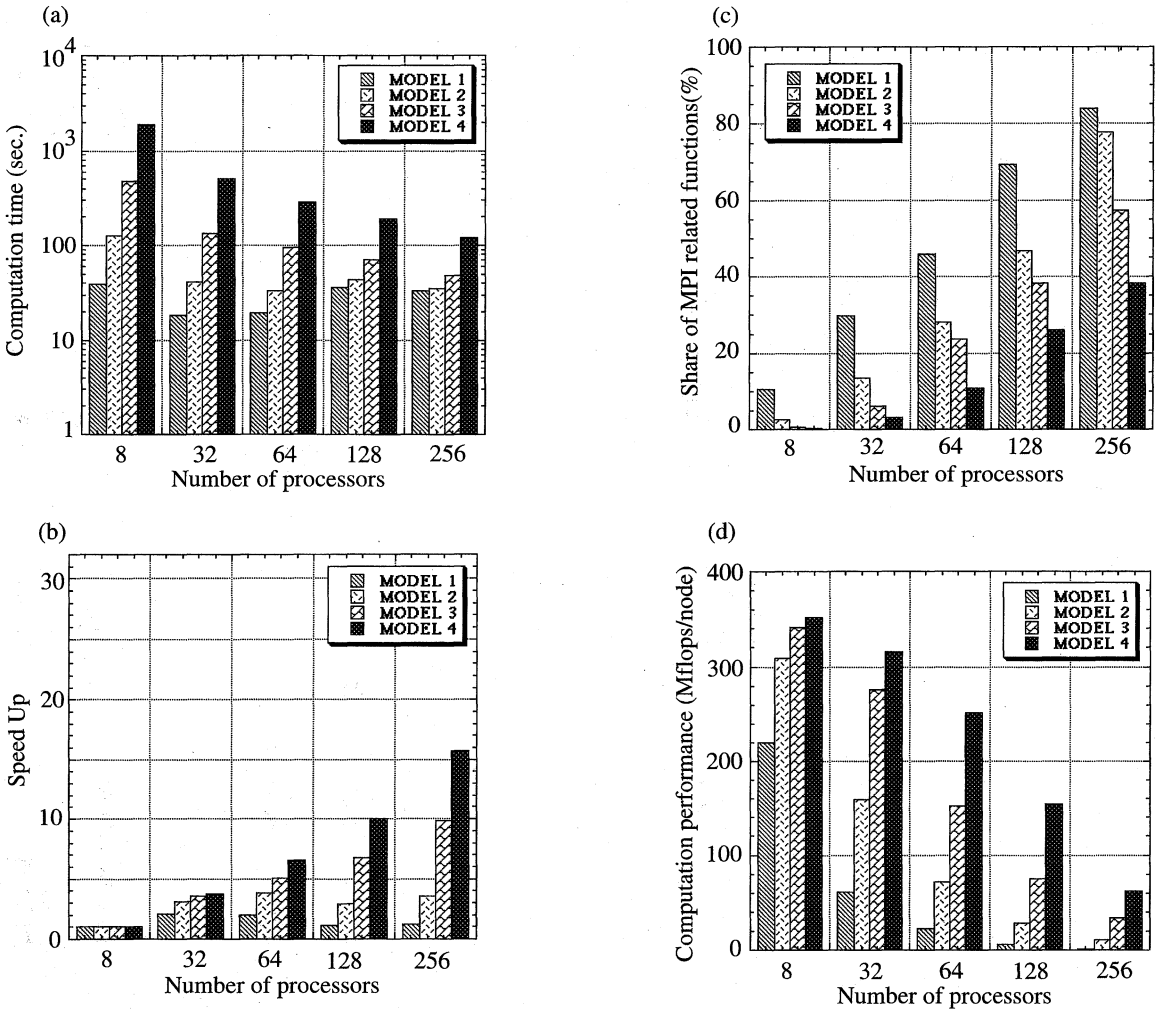

Fig. 9 (a) Measured computation time of the PSTM parallel processing as a function of the number of processors and data size in the case of distributed memory type with point-to-point communication, (b) Speed-up of the PSTM parallel processing as a function of the number of processors and data size in the case of (a), (c) Share rate of MPI related functions as a function of the number of processors and data size in the case of (a), (d) Computation performance as a function of the number of processors and data size in the case of (a).

た結果を Fig. 10(d)に示す。

5.3 標準的なパーソナルコンピュータとの演算性能比較

ここでは，前述の並列計算機を利用した処理時間結果に対して，現存の標準的なパーソナルコンピュータ（以下 PC と呼ぶ）による処理時間結果を比較する。用いた PC は Pentium III の 1 GHz（公称演算性能は 1 ギガフロップス）のプロセッサ，1 ギガバイトのメモリが搭載されており，OS には Linux（Kernel 2.2.16），コンパイラには gcc（version 2.96）を用いている。前述の Fig. 8(a)の結果（前述の 4 種類の並列処理の中で比較的に処理速度の速いもの）と比較したものを Fig. 11に示す。Fig. 11のグラフにおいて横軸はプロセッサ数（PC と表示しているデータは PC の場合の処理時間を示す）で縦軸は処理時間を示す。Fig. 11において，処理するデータ規模が小さいと並列計算機の本来の性能が出ていないことがわかる。例えば，ノード数32の処理時間とPCでの処理時間とを比較すると，Data 4 の場合には約70倍ほど並列計算機の方が速いが，Data 1 の場合には約 8 倍程度並列計算機の方が速いだけである。

6. 演算性能と通信性能評価による並列処理時間推定

ここでは，演算速度と通信速度の 2 つのパラメータに着目し，それらの基本性能に基づいて並列処理時間を推定する関係式を導入することを試みる。このような推定式により，任意の入力・出力データサイズに対して任意のプロセッサ数を利用して，どれくらいの時間で処理を実行できるかを予測できる。なお，通信性能については既に述べているので，ここでは推定式導入に先立ち，演算性能を評価する。

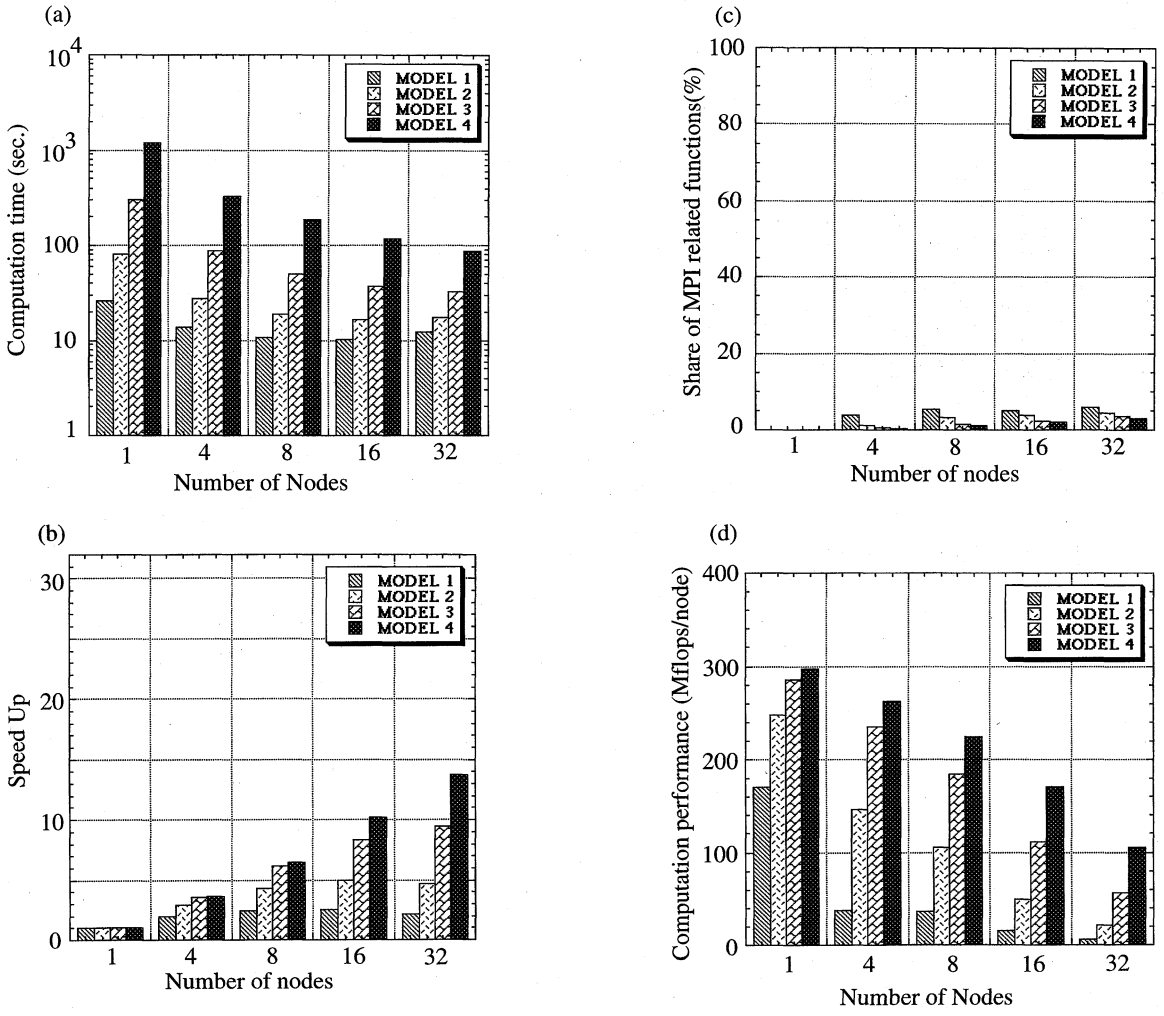

Fig. 10　(a) Measured computation time of the PSTM parallel processing as a function of the number of nodes and data size in the case of shared/distributed memory type with point-to-point communication, (b) Speed-up of the PSTM parallel processing as a function of the number of nodes and data size in the case of (a), (c) Share rate of MPI related functions as a function of the number of nodes and data size in the case of (a), (d) Computation performance as a function of the number of nodes and data size in the case of (a).

6.1　演算性能評価

まず，演算速度 *Vcalc* を以下の手順で求めた。1トレースは Y 個のサンプル（1サンプルは D メガバイトである）で構成され，X 本のトレースで構成される入力データを使用して任意の1点において K 個のプロセッサを用いてPSTM処理を実施することを考える。PSTM処理後の断面の仕様は，トレース数を *Tr_o*，各トレースのサンプル数を *Ns_o* とする。

1個のプロセッサ（あるいはノード。以下ではノードという表記を省略）が担当するデータサイズは，(入力データサイズ)/(使用するプロセッサ数）で計算されるので，$D \cdot X \cdot Y/K$（メガバイト）となる。PSTM処理全体から通信部分を除いた部分において時間測定関数を用いて処理に要する時間（1個のプロセッサあたり）を測定し，これを *Tcalc*（秒）とする。

以上より1個のプロセッサを用いて任意の1点において PSTM 処理するための演算速度は下記により計算される。

$$Vcalc=\frac{D \cdot X \cdot Y}{K \cdot Tcalc} \qquad (2)$$

1個のプロセッサを用いた場合と1個のノードを用いた場合において，(2)式に基づいて測定された演算速度を Fig. 12に示す。なお，$D=0.000008$，$X=80$，160, 320, 640, 1280, 2560, 5120, 10240, 20480, 40960, 81920 の11種類，$Y=1000$, $K=8$ とした。Fig. 12において，横軸はデータサイズ（単位：メガバイト），縦軸は演算速度（単位：メガバイト/秒）である。処理するデータサイズには大きく依存せず，演算速度 *Vcalc* は1プロセッサの場合で平均的に9530（メガバイト/秒），1ノードの場合で96700（メガバイト/秒）である。

また，PSTM を実施するポイント数（PSTM 処理後

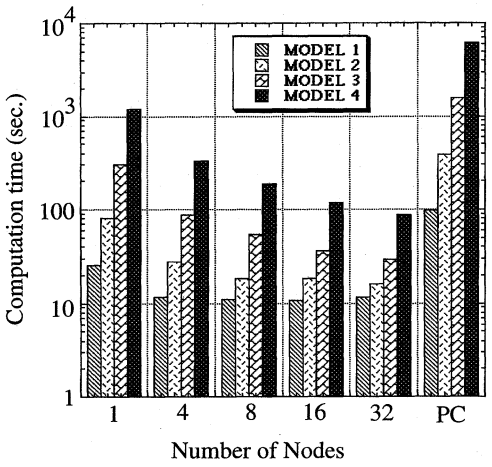

Fig. 11 Comparison of measured computation time between SR8K (shared/distributed memory system with group communication) and a standard personal computer with Intel Pentium III processor 1 GHz.

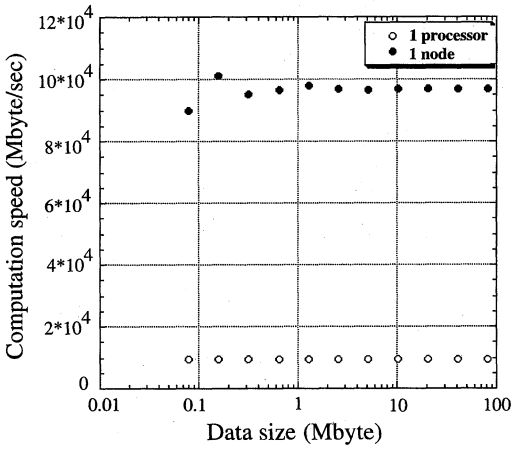

Fig. 12 Measured computation speed for PSTM as a function of data size. Black circles indicate the case of 1 node, while blank circles indicate the case of 1 processor.

の総サンプル数）は Tr_o*Ns_o であるので，処理開始から終了までの（1個のプロセッサあたり）総演算時間 *Tcalc*（秒）は以下の(3)式で計算される。

$$Tcalc\_all=\frac{D\cdot X\cdot Y\cdot Tr\_o\cdot Ns\_o}{K\cdot Vcalc} \tag{3}$$

なお，以下では，1個のプロセッサあたりの総演算時間，総通信時間，全処理時間を考えるが，これらが並列プログラムの総演算時間，総通信時間，全処理時間とそれぞれ一致すると仮定する。

6.2 通信性能評価

Fig. 5(a)～(d)により得られた通信速度を *Vcomm*（メガバイト/秒）とすると，*Vcomm* は通信するデータの大きさと使用するプロセッサ数の大小に依存することがわかるので，通信するデータの大きさを *Ds*（メガバイト）とし，プロセッサ数を *K* として *Vcomm* (*Ds*, *K*) と表現する。通信する回数を *Nc* とすれば，総通信時間 *Tcomm_all*（秒）は以下の(4)式で得られる。

$$Tcomm\_all=\frac{Ds\cdot Nc}{Vcomm(Ds,K)} \tag{4}$$

6.3 並列処理時間推定式の導入

ここでは，上記で得られた演算性能と通信性能の評価結果を基に並列処理時間の推定式を導出する。

6.3.1 グループ通信の場合

いま，入力データとして，トレース数 *Tr_i*，1本のトレースあたりのサンプル数を *Ns_i* とする。PSTM の出力データとして，トレース数 *Tr_o*，1本のトレースあたりのサンプル数を *Ns_o* とする。また使用するプロセッサ数を *Np* とする。また1つのサンプルのデータサイズは8バイトとする。

以上の条件により，(3)式において $D=0.000008$, $X=Tr\_i$, $Y=Ns\_i$, $K=Np$ となる。総演算時間 *Tcalc_all*（秒）は以下の(5)式になる。

$$Tcalc\_all=\frac{0.000008\cdot Tr\_i\cdot Ns\_i\cdot Tr\_o\cdot Ns\_o}{Np\cdot Vcalc} \tag{5}$$

一方，通信速度に関する(4)式において，$Ds=0.000012*Ns\_o$（PSTM の並列処理の説明の際に述べたように，通信には重合振幅データと重合数データの2種類送信する。重合振幅データは8バイト，重合数データは4バイトなので合計12バイト，これが *Ns_o* 個），$Nc=Tr\_o$ である。また *Vcomm* (*Ds*, *K*) に関しては，通信性能を実測したグラフ（Fig. 5）から回帰曲線を求める（これを *Vcomm_reg* とする）。以上により，総通信時間 *Tcomm_all*（秒）は以下の(6)式になる。

$$Tcomm\_all=\frac{0.000012\cdot Ns\_o\cdot Tr\_o}{Vcmm\_reg} \tag{6}$$

(5)式と(6)式により，PSTM 処理に要する総処理時間 *Tall*（秒）は以下の(7)式により得られる。

$$Tall=Tcalc\_all+Tcomm\_all \tag{7}$$

6.3.2 一対一通信の場合

グループ通信の場合と同様，入力データとして，トレース数 *Tr_i*，1本のトレースあたりのサンプル数を *Ns_i* とする。PSTM の出力データとして，トレース数 *Tr_o*，1本のトレースあたりのサンプル数を *Ns_o* とし，1つのサンプルのデータサイズは8バイトとする。また使用するプロセッサ数を *Np* とする。

前述したように，一対一通信の場合は，一つのプロ

セッサに結果を集計してその結果をファイルに書き出すために，集計を実施するプロセッサ（以下では集計プロセッサと呼ぶ）には，解析部分の負荷を他のプロセッサに比べて軽減する必要がある。その負荷率を α% とする（すなわち解析を担当するデータ規模が他のプロセッサの α% である）。以下では集計プロセッサとそれ以外のプロセッサを別々に考える。

(1) 集計プロセッサ

この場合は，(3)式において $D=0.000008$, $X=\alpha\cdot Tr\_i/(100\cdot(Np-1+\alpha/100))$, $Y=Ns\_i$, $K=1$ となる。総演算時間 *Tcalc_all*（秒）は以下の(8)式になる。

$$Tcalc\_all=\frac{0.000008\cdot\alpha\cdot Tr\_i\cdot Ns\_i\cdot Tr\_o\cdot Ns\_o}{Vcalc\cdot 100\cdot(Np\_1+\alpha/100)}+Tsum\cdot Tr\_o \quad (8)$$

(8)式において，*Tsum* は集計プロセッサが一本分の出力トレース結果のデータを集計してファイルに書き出すのに要する時間（秒）である。*Tsum* は集計されるデータの規模に依存する（Fig. 13）ので，*Tsum* に関しても Fig. 13における回帰曲線を求めておく。

一方，通信速度に関する(4)式において，$Ds=0.000012*Ns\_o$, $Nc=Tr\_o$ である。また *Vcomm*（*Ds*, *K*）に関しては，通信性能評価で実測されたグラフ（Fig. 5）から回帰曲線を求める（これを *Vcomm_reg* とする）。以上により，総通信時間 *Tcomm_all*（秒）は以下の(9)式になる。

$$Tcomm\_all=\frac{0.000012\cdot Ns\_o\cdot Tr\_o}{Vcmm\_reg} \quad (9)$$

(8)式と(9)式により，PSTM 処理に要する総処理時間 *Tall* は前述の(7)式により得られる。

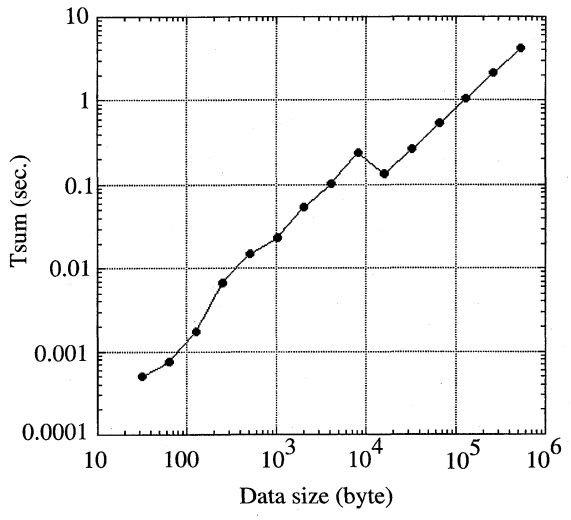

Fig. 13　Measured computation time for Tsum as a function of data size.

(2) 集計プロセッサ以外のプロセッサ

この場合は，(3)式において $D=0.000008$, $X=(Np-1)\cdot Tr\_i/(Np-1+\alpha/100)$, $Y=Ns\_i$, $K=Np-1$ となる。総演算時間 *Tcalc_all*（秒）は以下の(10)式になる。

$$Tcalc\_all=\frac{0.000008\cdot(Np-1)\cdot Tr\_i\cdot Ns\_i\cdot Tr\_o\cdot Ns\_o}{Vcalc\cdot(Np-1)\cdot(Np-1+\alpha/100)} \quad (10)$$

一方，通信速度に関する(4)式において，$Ds=0.000012*Ns\_o$, $Nc=Tr\_o$ である。また *Vcomm*（*Ds*, *K*）に関しては，通信性能評価で実測されたグラフ（Fig. 5）から回帰曲線を求める（これを *Vcomm_reg* とする）。以上により，総通信時間 *Tcomm_all*（秒）は以下の(11)式になる。

$$Tcomm\_all=\frac{0.000012\cdot Ns\_o\cdot Tr\_o}{Vcomm\_reg} \quad (11)$$

(10)式と(11)式により，PSTM 処理に要する総処理時間 *Tall*（秒）は前述の(7)式により得られる。なお，集計プロセッサの通信時間を表す(9)式と，それ以外のプロセッサの通信時間を表す(11)式とが同等であるが，これは全通信が同時に実施されると仮定しているためである。

上記により，集計プロセッサとそれ以外のプロセッサでの総処理時間が求められた。効率的に並列処理が実施されるのは，両者の総処理時間が等しくなる場合である。すなわち，「(8)＋(9)＝(10)＋(11)」の関係式より α の値を設定することである。入力データならびに出力データ仕様，使用するプロセッサ数が決まれば，簡単な算術により導くことができる。

6.4　数値実験データへの適用

ここでは，前述の数値実験の仕様に対して，前述の4種類の並列方法による処理時間の見積もり評価を行う。

数値実験の仕様により，演算時間に関する(5)式において，*Tr_i* に関しては（2601, 10201, 40401, 160801）の4種類，$Ns\_i=1000$, *Np* に関しては，分散メモリ型の場合は（8, 32, 64, 128, 256）の5種類，ノード内共有メモリ型の場合は（1, 4, 8, 16, 32）の5種類，$Tr\_o=101$, $Ns\_o=1000$ である。また演算速度 *Vcalc* に関しては，分散メモリ型の場合は9530（メガバイト/秒），ノード内共有メモリ型の場合は96700（メガバイト/秒）とする。

一方，通信時間に関する(6)式において，$Tr\_o=101$, $Ns\_o=1000$, *Vcomm_reg* に関しては，Fig. 5(a)～(d)で得られた4種類の通信性能のプロット結果に対して回帰曲線を求める。

以上により，前述の数値実験の場合と同様にケース1

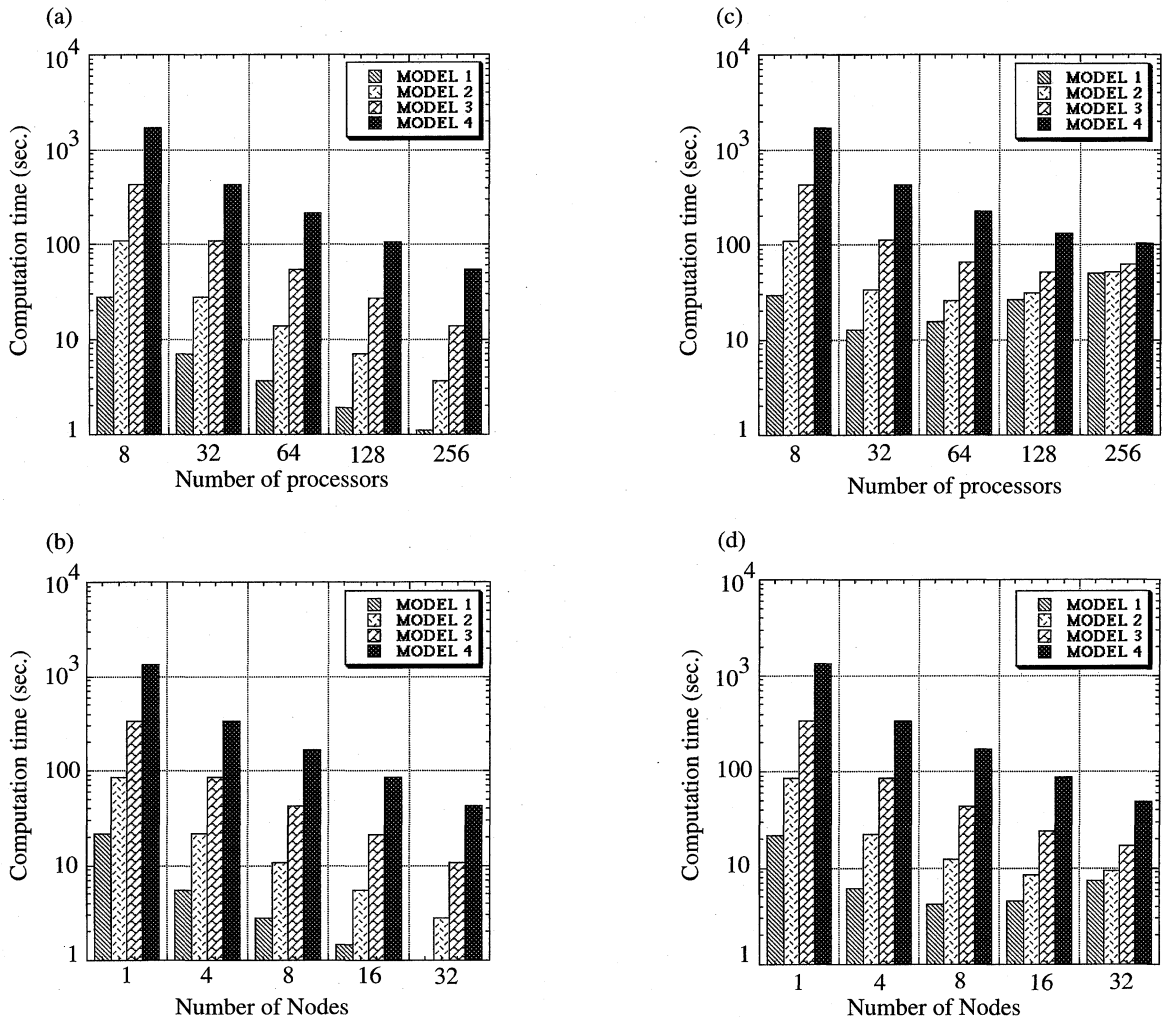

Fig. 14 (a) Theoretical computation time of the PSTM parallel processing in the case of Fig. 7(a), (b) the case of Fig. 8(a), (c) the case of Fig. 9(a), (d) the case of Fig. 10(a).

～ケース4までの場合について処理時間を推定した結果をそれぞれFig. 14(a)～(d)に示す。ケース1～ケース4の場合の実測結果（Fig. 7(a)，Fig. 8(a)，Fig. 9(a)，Fig. 10(a)）と推定結果（Fig. 14(a)～(d)）を比較すると以下のことがわかる。一対一通信の場合には，実測と推定結果が概ね一致しているのに対してグループ通信の場合には，特にプロセッサあるいはノード数が多くなるほど（通信占有率が大きくなるほど），実測と推定結果との差が大きくなっている。このようにグループ通信の場合に，推定と実測に差が出てしまうのは，前述のようにグループ通信性能評価（Fig. 5(a)とFig. 5(b)）が過大評価であったことによる。過大評価の原因については，ハードウェアの特性による可能性があるが，詳細な原因は不明である。

7. 結　論

本研究では，松島ほか（2001）が提案した重合前時間マイグレーション手法をMPIと呼ばれる通信ライブラリを用いたメッセージパッシング方式を採用して並列計算機に実装した。並列処理する際に，2種類のハードウェアアーキテクチャ（分散メモリ型と階層メモリ型）と2種類のプロセッサ間通信方法（グループ通信と1対1通信）を組み合わせて4種類の並列処理を比較した。また，使用する並列計算機の演算性能ならびに通信性能を評価結果を基にして，PSTMの並列処理時間を推定する試みも行った。その結果以下のことがわかった。

- ハードウェアアーキテクチャに関しては，ノード内プロセッサを単位とした分散メモリアーキテクチャに比べて，ノード内はコンパイラによる自動並列かつノード間は分散メモリ並列のアーキテクチャの方が優位性があった。これは前者に比べて後者の通信占有率が相対的に小さいことによることと，ノード内自動並列が効率的に行われたことによる。
- 通信方法に関しては，通信性能のみの評価では1対1通信に比べてグループ通信に優位性がみられたが，演算も含めた処理時間実測では，両者に大きな差は見られな

かった。これは，演算部分を含まず通信性能のみを評価したときにグループ通信性能を過大評価してしまったことが原因であると考えられるが，この原因の究明は今後の課題としたい。また，通信性能の評価においては，プロセッサ同士を結合するネットワークトポロジの影響を受けるので，どのような通信をどのようなネットワークトポロジ上で行うのが最適かを考慮する必要がある。これに関する検討も今後の課題としたい。

• 入力データの規模，得られる出力データの規模，使用するプロセッサ数により，処理時間は変化する。使用する並列計算機の演算性能ならびに通信性能を評価結果を基にして，PSTMの並列処理時間を推定することを試みた。これにより，何個のプロセッサを使用すればどれくらいの処理時間で実行できるかを予測することができる。このことはデータ処理上有効な情報になるばかりでなく，計算機資源の有効利用にもつながると思われる。

また，さらなるプログラムチューニングとして，ハードウェアの特性を考慮したさらなるチューニングも今後の課題としたい。特にSR8Kは，リモートDMA（Direct Memory Access）転送機能と呼ばれるハードウェア機構を有し，通信のためのバッファを介することなく直接通信を行うため，ハードウェアのピーク性能に迫る転送バンド幅を達成することができるので，今後はこの機能に関する検討も進めたい。

謝　辞

本研究の一部は産業技術総合研究所先端情報計算センターの高性能計算機利用促進課題のもとで実施され，本研究で作成した並列プログラムのチューニングにあたり，株式会社日立製作所公共システム事業部科学技術システムセンタ・ソフトウェア事業部の御支援を受けました。高性能計算機利用促進課題の関係者各位に謝意を表します。また，匿名の査読者の方々には，詳細な査読をしていただき，論文改善のための有益なご指摘をいただきました。感謝の意を表します。

参　考　文　献

Amdahl, G. M. (1967) : Validity of single-processor approach to archiving large scale computing capabilities, *Proceeding of Spring Joint Computer Conference*, **30**, 483–485.

Chang, H., VanDyke, J., Solano, M., McMechan, G. and Epili, D. (1998) : 3–D prestack Kirchhoff depth migration: From proto type to production in a massively parallel processor environment, *Geophysics*, **63**, 546–556.

Epili, D. and McMechan, G. (1996) : Implementation of 3–D prestack Kirchhoff migration, with application to data from the Ouachita frontal thrust zone, *Geophysics*, **61**, 1400–1411.

French, W. S. (1992) : Implications of parallel computation in seismic data processing, *The Leading Edge*, **11**, 22–25.

Gropp, W., Lusk, E. and Skjellum, A. (2000) : Using MPI 2nd Edition, MIT Press, 346p.

Zhang, L. (1997) : Evaluating the performance and accuracy of 3–D prestack depth migration, *67th Annual Internat. Mtg., Soc. Expl. Geophs., Expanded Abstracts*, 1414–1417.

佐藤岳彦・松岡俊文・鶴　哲郎（1995）：深度マイグレーションアルゴリズム，物理探査学会第92回学術講演会論文集，135–139.

妹尾義樹・左近彰一（2000）：並列化の基礎とプログラミングインタフェース，計算工学，**5**, 51–55.

中島義成・松岡俊文・鶴　哲郎（1997）：キルヒホッフ法による3次元重合前時間マイグレーション処理，物理探査学会第96回学術講演会論文集，135–139.

古村孝志・纐纈一起・竹中博士（2000）：大規模3次元地震波動場（音響場）モデリングのためのPSM/FDMハイブリッド型並列計算，物理探査，**53**, 294–308.

松島　潤・六川修一・横田俊之（2001）：重合速度解析をともなう散乱重合法による反射法地震探査データ処理，物理探査，**54**, 73–89.