ハイパフォーマンス
コンピューティング 52－6
（1994．7．22）

陽的解法を用いた3次元流体コードの並列化

坂上仁志

姫路工業大学工学部情報工学科

現在のスーパーコンピュータを大きく越える超高速演算能力を持つことが期待されている複数命令分散メモリー・タイプの並列計算機を主眼におき，陽的解法を用いた3次元流体コードIMPACT-3Dの並列化について述べる．並列計算機では，3次元シミュレーション空間を小さな部分空間に分割して各プロセッサーに割り振り負荷の分散をはかるが，この空間の分割方法の違いと達成できるスピードアップ率および並列実行効率について検討する．更に，プロセッサーを結合するネットワーク構造とプロセッサー間のデータ通信速度の関係について触れ，それが並列実行効率に与える影響を述べる．

A PARALLEL IMPLEMENTATION OF THREE-DIMENSIONAL EXPLICIT FLUID CODE

Hitoshi Sakagami

Department of Computer Engineering, Faculty of Engineering,
Himeji Institute of Technology
2167 Shosha, Himeji-shi, Hyogo, 671-22 Japan

This paper describes a implementation of three-dimensional explicit fluid code, IMPACT-3D onto multiple-instruction multiple-data distributed memory type parallel systems. The three-dimensional simulation space were divided into sub-spaces and each sub-space was assigned to each processor in order to keep load balances between processors. The effective parallel efficiency is discussed with relation to the different dividing methods. The interconnecting network topology and the communication rate are also evaluated to determine the effective parallel efficiency.

1．はじめに

核融合，流体力学，気象，計算化学等の分野ではスーパーコンピュータを用いたシミュレーションが重要な研究手段になっており，それがより詳細に，より大規模になるのに伴って，現存のものよりも更に高性能なスーパーコンピュータを必要としている．しかし，既存技術の延長で製造されるスーパーコンピュータでは大幅な性能向上を望めないため，超高速演算を達成する別の方法として複数のプロセッサーを持つ，いわゆる並列計算機が期待されている．

そこで，本論文では複数命令分散メモリー・タイプの並列計算機を主眼におき，陽的解法を用いた3次元流体コードIMPACT-3Dの並列化について述べる．並列計算機では，3次元シミュレーション空間を小さな部分空間に分割して各プロセッサーに割り振り，負荷の分散をはかるが，2章ではこの空間の分割方法の違いによる並列実行効率の差について述べる．3章ではプロセッサーを結合するネットワーク構造とプロセッサー間のデータ通信速度について触れ，4章で結論を述べる．

2．シミュレーション空間の分割

IMPACT-3Dは，3次元シミュレーション空間にカーテシアン座標系（直交座標系）を用いて立方格子状の計算グリッドを導入し，離散化された流体方程式を陽的解法で解いている．このため，計算には近距離の絡合いしかなく，シミュレーション空間を部分空間に分割して，各々の部分空間を別々のプロセッサーに割り振って並列計算を行なうことが原則として容易である．

多次元の時間発展は分ステップ法を用いて1次元問題の重ね合わせで解いており，空間の微分には5点差分スキームを使用している．1次元方向のみを考えると部分空間の境界は図1のようになる．

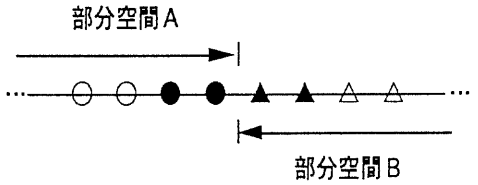

図1：部分空間の境界

部分空間内のデータだけでは境界上の点は計算できないので，図1の部分空間A（B）において●（▲）の点を計算するためには，部分空間B（A）の▲（●）の点のデータを得なければならない．このため，隣接した部分空間を受け持つプロセッサー間で時間ステップごとに境界のデータを交換する必要が生じる．

IMPACT-3Dの各モジュールは次のような性質を持つ．また，括弧内は時間を1サイクル発展させるために各モジュールが実行される回数を示す．

CMPRAM: 時間ステップ△tの計算をする．全シミュレーション空間の値が必要である．（1回）

ADV3DX: x方向の時間発展を解く．部分空間で独立に計算できるが，隣接空間同士でデータの交換が必要である．（3回）

ADV3DY: y方向の時間発展を解く．部分空間で独立に計算できるが，隣接空間同士でデータの交換が必要である．（2回）

ADV3DZ: z方向の時間発展を解く．部分空間で独立に計算できるが，隣接空間同士でデータの交換が必要である．（2回）

CMP3DP: 圧力の計算をする．部分空間で独立に計算できる．（7回）

2.1　z方向のみの分割

まずは，z方向にのみシミュレーション空間を分割する方法について考える．ADV3DX，ADV3DYの計算には，データ交換は必要ないが，ADV3DZの計算には必要になる．x，yおよびz方向のシミュレーション空間の長さをそれぞれL_x，L_yおよびL_zとすると，ADV3DZ実行時に部分空間の境界で隣接プロセッサーが交換するデータ量は，片側2平面，1平面当たり6変数，1変数は4バイトなので$L_x \times L_y \times 4 \times 6 \times 4 = 96 L_x L_y$バイトとなる．データ交換はプロセッサー間の通信機能（データの送信と受信）で行なうとすると，データ交換のためには同期を2回取らなければならないので，結局1時間サイクル当たりのデータ交換に要する時間は，次のようになる．

$$t_{comm} = 192\, L_z L_x (1/S_{send} + 1/S_{recv}) + (1+4)\, t_{sync}$$

ここで，t_{sync}は同期オーバーヘッドであり，S_{send}は複数のプロセッサーが同時に他のプロセッサーに送信できる速度であり，S_{recv}は他のプロセッサーから受信できる速度である．

ADV3DX，ADV3DYの計算は無駄なく並列化できるが，ADV3DZの計算には，部分空間ごとに4メッシュ分を重複して計算しなければならない．また，CMPRAMおよびCMP3DPの演算時間は短いので，これらの寄与は無視できる．z方向の分割数をn_zとすると，1時間サイクル当たりの計算時間は，演算時間はシミュレーション空間の大きさに比例すると考えられるので次のようになる．

$$t_{calc} = (3\, t_{advx} + 2\, t_{advy} + 2\,(4\,(n_z - 1) + L_z)/L_z\, t_{advz})/n_z$$

ここで，t_{advx}，t_{advy}，t_{advz}は，それぞれADV3DX，ADV3DY，ADV3DZの分割前の実行時間である．一般には，分割数が増えるとプロセッサー当たりの演算量が減りベクトル処理等の実行効率が下がるため分割数分だけは速くならないが，ここではそれを無視している．

2.2　yz方向の分割

次に，yz方向にシミュレーション空間を分割する方法について考える．ADV3DYの実行時にはy方向に隣接するプロセッサーの境界でのみzx平面のデータを交換する必要があるが，そのデータ量は，z方向の分割数をn_zとすると$1/n_z$に少なくなる．ADV3DZでも同様なので，全体で1時間サイクル当たりのデータ交換に要する時間は次のようになる．

$$t_{comm} = 96\,(2\, L_z L_x / n_z + 2\, L_x L_y / n_y)(1/S_{send} + 1/S_{recv}) + (1+8)\, t_{sync}$$

ADV3DYでは，y方向の分割に対してはメッシュを拡張して計算する必要があるが，z方向の分割に対しては必要ないので，重複計算分を同様に考慮すると1時間サイクル当たりの計算時間は，

$$t_{calc} = (3\, t_{advx} + 2\,(4(n_y - 1) + L_y)/L_y\, t_{advy} + 2\,(4(n_z - 1) + L_z)/L_z\, t_{advz})/n_y n_z$$

となる．

2.3　x y z方向の分割

最後に，x y zすべての方向にシミュレーション空間を分割する場合を考える．隣接するプロセッサーの境界で交換するデータ量は，2方向で分割されるため更に少なくなるので，同様に考えて全体で1時間サイクル当たりのデータ交換に要する時間は，

$$t_{comm} = 96\,(3\,L_y L_z / n_y n_z + 2\,L_z L_x / n_z n_x + 2\,L_x L_y / n_x n_y)(1/s_{send} + 1/s_{recv}) + (1+14)t_{sync}$$

となり，重複計算分を考慮すると1時間サイクル当たりの計算時間は，

$$t_{calc} = (3\,(4(n_x-1) + L_x)/L_x\ t_{advx} + 2\,(4(n_y-1) + L_y)/L_y\ t_{advy} + 2\,(4(n_z-1) + L_z)/L_z\ t_{advz})/n_x n_y n_z$$

となる．

2.4　スピードアップ率と並列実行効率

データ交換時間の占める割合r_{comm}および重複計算を考慮した演算のみの並列効率$r_{calc.eff}$を次のように定義する．

$$r_{comm}(\%) = t_{comm}/(t_{comm} + t_{calc}) * 100$$
$$r_{calc.eff} = ((3\,t_{advx} + 2\,t_{advy} + 2\,t_{advz})/n)/t_{calc}$$

ここで，nは並列処理を行なうプロセッサー数であり，z方向にのみ分割した場合には$n = n_z$，y z方向に分割した場合には$n = n_y n_z$，x y zすべての方向に分割した場合には$n = n_x n_y n_z$となる．

また，並列化による実行のスピードアップ率$r_{speedup}$および並列実行効率r_{eff}は，次のように定義できる．

$$r_{speedup} = (3\,t_{advx} + 2\,t_{advy} + 2\,t_{advz})/(t_{comm} + t_{calc})$$
$$r_{eff} = r_{speedup}/n$$

並列計算機システムを構成するプロセッサーのピーク性能を1GFLOPS程度と想定するとNEC SX-2の実測値を考慮して，演算時間は，

$$t_{advx} = t_{advy} = t_{advz} = 1 \cdot L_x L_y L_z / 80^3\ (\mathrm{sec})$$

と評価できる．通信速度は種々の条件により影響を受けるが，ここでは簡単のため一定と考える．実際に稼働している分散メモリー・タイプの並列計算機システムを参考にして，実効的な通信速度を次

のように評価する．

$$S_{send} = S_{recv} = 20\ (MB/sec)$$

また，同期オーバーヘッドはマイクロ秒程度と考えられるので，データ交換に要する時間や演算時間に比べて無視できるため$t_{sync}=0$とする．

以上で評価された値を用い，３つの分割方法について，データ交換時間の占める割合および演算のみの並列効率を図２に，スピードアップ率および並列実行効率を図３に示す．ただし，パラメータは，$L_x=L_y=L_z=L=256$および$n_x=n_y=n_z$とした．

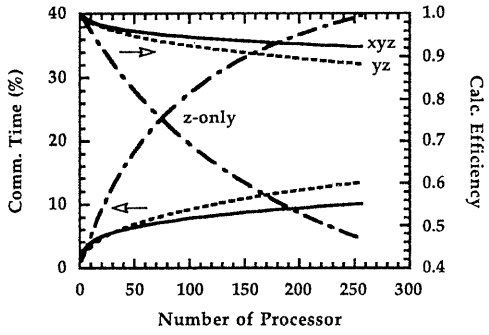

図２：データ交換時間と演算の並列効率

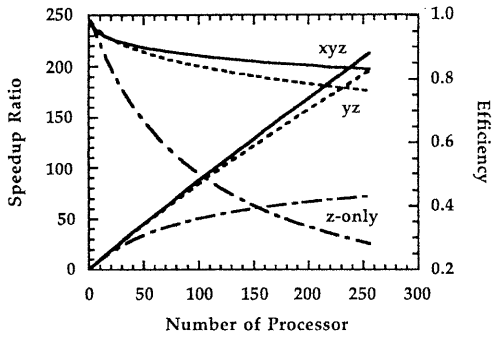

図３：スピードアップ率と並列実行効率

ｚ方向のみの分割の場合は，プロセッサー数が増えると演算時間はほぼ～１／ｎになるが，データ交換時間はプロセッサー数に関係なく一定なのでデータ交換時間の占める割合が急速に大きくなる．また，プロセッサー数が増えて分割数が多くなるとプロセッサー当たりのｚ方向の長さが短くなるため重複計算のオーバーヘッドが非常に大きくなり演算の並列効率が悪くなる．このため，２５６プロセッサーで７２倍のスピードアップ，並列実行効率０.２８しか得られない．ｙｚ方向，ｘｙｚ方向に分割する場合はプロセッサー数が増えるとデータ交換時間はそれぞれ～$n^{-1/2}$，～$n^{-2/3}$と減少するので，データ交換時間の占める割合はそれぞれ～$n^{1/2}$，～$n^{1/3}$とｚ方向にのみ分割する場合に比較して，ゆっくりとしか増加しない．また，１方向当たりの長さが極端には短くならないので重複計算のオーバーヘッドもあまり大きくならず，演算の並列効率はほぼ０.９以上である．このため，２５６プロセッサーでは，ｙｚ方向に分割する場合は１９５倍のスピードアップおよび並列実行効率０.７６が得られ，ｘｙｚ方向に分割する場合には２１２倍のスピードアップおよび並列実行効率０.８３と十分高い並列性能が得られる．

３．ネットワーク構造とデータ通信速度

２章ではデータ通信速度を一定と考えたが，実際には実装上の問題からデータ通信速度はネットワーク構造に大きく依存する．同一コストの並列計算機システムでは，ｘｙｚ方向の分割に適しているハイパーキューブ構造や多段クロスバーよりｙｚ方向の分割に適しているトーラス構造が，更に，ｚ方向のみの分割に適しているリング構造のほうが，ネットワーク構造の特性を考えるとデータ通信速度が速いと一般的に考えられる．そこで，データ通信速度をｙｚ方向に分割する場合は１０倍に，

z方向のみに分割する場合は１００倍にしたときのスピードアップ率および並列実行効率を図４に示す．図３と比較すると両方の場合とも結果が改善されていることが分かるが，z方向のみに分割する場合は重複計算のオーバーヘッドが大きいため，いくらデータ通信速度を速くしてもこれ以上の改善は望めない．しかし，ｙｚ方向に分割する場合には，データ通信速度を速くすることで非常に高い並列実行効率が得られ，むしろ元のデータ通信速度でｘｙｚ方向に分割する場合よりも良くなっている．

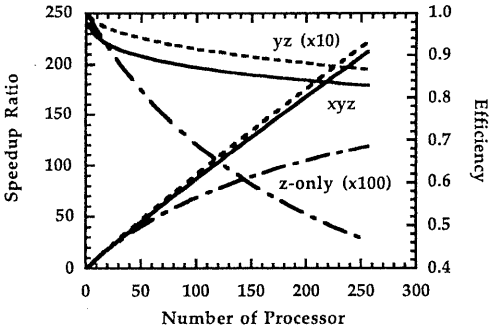

図４：通信速度の違いによる並列実効効率の変化

このため，本論文で考察しているプロセッサーの能力およびデータ通信速度では，ハイパーキューブ構造や多段クロスバーで並列計算機システムを構成してｘｙｚ方向に分割する方法より，トーラス構造を用いたデータ通信速度が１０倍のシステムでｙｚ方向に分割する方法の方が，陽的解法を用いたIMPACT-3Dを並列化する場合には高いスピードアップ率が得られることになる．

ただし，CMPRAMの計算にはすべてのプロセッサーのデータが必要なため隣接プロセッサー間だけでなくブロードキャスト的な通信を行なわなければならないが，一般にハイパーキューブ構造や多段クロスバー型のネットワークの方がトーラス構造より高速にこれを実行できる．本論文では，CMPRAMの計算で通信されるデータ量が極めて小さいため，このブロードキャスト的通信の時間を無視したが，この通信方法が非効率的な方法でトーラス構造の並列計算機システムにインプリメントされているなら，それらを考慮して議論しなければならないことを付記しておく．

５．結論

陽的解法を用いた３次元流体コードIMPACT-3Dをシミュレーション空間の分割によって並列化することを考え，その分割方法としてz方向のみの分割，ｙｚ方向の分割およびｘｙｚ方向の分割の３種類の場合を考慮した．

z方向のみに分割する場合，演算，データ通信のどちらもオーバーヘッドが大きいのでいくらデータ通信速度を速くしても低い並列実行効率しか得られない．

ｘｙｚ方向に分割する場合は非常に高い並列実行効率が得られるが，この方法に最適な並列計算機システムであるハイパーキューブ構造や多段クロスバー型のネットワークでは，実装の難しさやメモリーコンフリクト等によるデータ通信速度の低下が問題となる．更に，ベクトル長が短くなるために発生するパフォーマンスの低下も問題となるため，実際にはこの方法が必ずしも最良とはならない．

本論文で用いているパラメータの範囲では，トーラス構造の並列計算機システムが同じコストでハイパーキューブ構造や多段クロスバー型のシステムより１０倍のデータ通信速度を実現できるのなら，トーラス・システムを用いてｙｚ方向に分割して並列化する方法が最良となる．ベクトル長によるパフォーマンスを考慮すると，このｙｚ方向に分割する方法が，ｘｙｚ方向に分割する方法に比べて更に有利になる．