並列非同期アルゴリズムを用いた熱流体解析の高速化*
(点SOR法による解析の高速化評価)大島 丈治*¹, 永田 真也*²
石川 博章*¹, 古藤 悟*¹Acceleration of Computational Fluid Dynamics
by Parallel Asynchronized Iterative Algorithms
(An Evaluation of Speed-up for Point SOR Method)Takeharu OSHIMA, Shinya NAGATA,
Hiroaki ISHIKAWA and Satoru KOTOH

This paper describes a methodology for the implementation of asynchronous parallel iterative algorithms, which is the domain decomposition method. This approach reduces the cost of synchronization that affects synchronous algorithms. As an application of the proposed approach, the algorithms based on the well-known point-SOR methods are presented in both synchronous and asynchronous forms. Results obtained using the MIMD (Multiple Instruction stream Multiple data stream) superparallel computer are given, for the model problem of the solution of a Poisson equation.

Key Words: Computational Fluid Dynamics, Numerical Analysis, Finite Difference Method, Domain Decomposition Method, Parallel Algorithms, Multiprocessor Systems, Asynchronous Iterations, Convergence of Iterations

1. 緒 言

熱流体数値解析の並列化手法⁽¹⁾は、計算領域をいくつかの領域に分割し、各領域内の計算をそれぞれ一つのプロセッサに割り振り、境界条件をプロセッサ間通信で受渡しする領域分割法(Domain Decomposition Method)^{(1)~(9)}が主流である。例えば、領域分割法を有限要素法^{(1)(3)~(5)}あるいは有限差分法^{(1)(6)~(9)}に適用して高い並列化効率(1個のプロセッサで計算した際の計算時間と n 個のプロセッサを並列に動作させた際の計算時間を n で割った値との比)を達成したという報告がなされている。領域分割法の問題点は、境界条件を受け渡すために行うプロセッサ間通信によって、通信の同期待ちが生じる点や、さらに計算空間を複数の領域に分割することによって新たに分割境界が発生し、分割面での収束性が劣化する点等が挙げられる。これら二点に関してさまざまな研究⁽⁸⁾⁽⁹⁾がなされており、既存のプログラム資産を並列計算機上にインプリメントするうえでの重要な指針を示している。

一方、このようなソフト面からの並列化へのアプローチのほかに、ハード面からのアプローチが考えられる。従来行われてきた研究は、与えられた計算機のアーキテクチャに適した既存プログラムあるいはアルゴリズム改良の議論に終始せざる得なかった。汎用並列計算機のアーキテクチャは汎用性を最大限に抽出する構造となっており、熱流体数値解析の基本である、偏微分方程式の解法に対して最適なハード構成とはいえない。もしも、熱流体数値解析のための専用計算機が実現されれば、汎用並列計算機のパフォーマンスを飛躍的に凌駕することが可能となる。

そこで、本報においては、並列計算機の最大の特徴であるプロセッサ間通信方式に着目し、熱流体数値解析を高速化するのに必要なプロセッサ間通信方式について考察し、偏微分方程式の数値解析に最適な通信方式を模索する。

2. 偏微分方程式の新高速解法の提案

現在のほとんどの並列計算機が、プロセッサ間通信に関しては、プロセッサ間のデータ授受の信頼性を保証するために、プロセッサ間で通信待ちを行う同期方式のアーキテクチャを採用している。流体解析のように隣接プロセッサ間の通信が多発する場合には、この

* 原稿受付 平成6年9月21日。

¹ 正員, 三菱電機中央研究所 (〒661 尼崎市塚口本町 8-1-1)。² 正員, 三菱電機本社設計システム技術センター (〒661 尼崎市塚口本町 8-1-1)。

同期通信方式を使用している限り、プログラム上の工夫によってある程度は通信同期待ちの時間を短縮することはできても、根本的に排除することは不可能である。ところが、プロセッサ間のメモリアクセス動作をハンドシェイクを必要としない非同期通信方式とした場合、通信同期待ちを完全に排除することができる。

ここで、この非同期通信機構を有したマルチプロセッサシステムの例を取り上げてみる。まず、並列化効率の極端な悪化から従来では行われていなかった超細粒度(プロセッサが行う演算量に対し、通信量が極端に多い場合)の計算手法が可能となる。図1に示した計算手法の概念は、1個のプロセッサに一つの流体要素を割り当てて、ハードウェア構成に熱流体物理現象を直接反映させたものである。物理現象においては、隣接する流体要素間の相互作用に伴い、全空間で同時並列に物理量が変化していく。そこで、一つの流体要素に1個のプロセッサを対応させ、隣接流体要素間の相互作用をプロセッサ間通信で置き換え、同時並列な情報伝搬を実現するために、プロセッサ間のメモリアクセスを非同期とすることで、流体物理現象に忠実なマルチプロセッサシステムが実現される。

また、近年研究が盛んになってきている、ニューラルネットワークを用いた有限要素法解析⁽¹⁰⁾においても、1プロセッサに1ニューロンを割り当て、プロセ

ッサ間通信を非同期方式にすることで、ニューロンを刺激するタイミングをランダムかつ同時並列に行い、ニューラルネットワークの命令レベルの並列性を十分に抽出することが可能となる。

これらの偏微分方程式の高速解法を可能とするシステムを実現するには、非同期通信による反復収束計算の可能性を確認する必要がある。非同期収束演算に関する研究は、汎用並列計算機の通信方式を確立していく過程で行われ、数学的なアプローチから収束性を議論したもの⁽¹¹⁾⁽¹²⁾や、4プロセッサ程度の小規模のマルチプロセッサシステムを使用した実験⁽¹¹⁾⁽¹³⁾⁽¹⁴⁾等が散見される。しかし、実際に大規模並列計算機を使用しての数値実験はなされていなかった。

本研究は、並列計算方式の重要課題である通信同期待ち問題を完全に解決する手段として、非同期通信方式を取り上げ、PVM(Parallel Virtual Machine)⁽¹⁵⁾あるいはバリエーションによる同期通信方式との比較を行い、ポアソン方程式の並列反復演算を例に取り、非同期計算特有の問題点やその可能性について考察した。

3. マルチプロセッサ間の通信形式

本研究に使用した並列計算機は、表1に示すような諸元をもつMIMD型、分散メモリー型の高並列計算機で、各プロセッサは高速データ通信ネットワークを介して三次元トラス状に接続されている。この計算機は、PVM、バリエーション、非同期通信の三つの通信方式を備えており、計算機の演算処理能力を変えず、通信方式のみを入れ替えることが可能である。

ここで、PVMは通信をするプロセッサ間でデータを送る側と受ける側の双方が、それぞれ送信と受信の関数コールをすることでデータの通信を行う、メッセージパッシング方法である。また、非同期通信は通信をするプロセッサ間の一方のみが指定した相手プロセッサの動作とは独立に相手メモリーからデータを読み出すshmem_getと呼ばれる関数をコールすることで、非同期のメモリアクセスを行う通信方法である。さらに、バリエーションはshmem_getによってデー

図1 熱流体物理現象と超細粒度の対比

表1 計算に使用した高並列計算機の諸元

processor	clock time	150 MHz
	peak performance	150 Mflops
	DRAM	2 Mwords
	data cash	1 Kwords
system	data transfer rate	300 MB/sec
	peak performance	4.8 Gflops (32PE)

タ通信を行い、その通信の前後においてバリヤーと呼ばれる関数をコールして、全プロセッサの処理がそこに到達するまで各プロセッサをアイドル状態にしておく方法で、PVM に比べ高速の同期通信が可能となる。

4. 検 証

4.1 領域分割法の問題点 まず、並列処理における各種処理時間を、以下のように定義する。

計算時間＝演算時間＋通信時間

通信時間＝同期待ち＋転送時間＋通信オーバーヘッド
ただし、通信のオーバーヘッドとは、通信のための関数コールを行うのに要する時間とする。

領域分割法による並列化効率を支配しているファクタには、(1)転送時間の増加、(2)分割面での減速緩和による収束性劣化、(3)通信同期待ち、が考えられる。(1)に関しては並列計算機の内部データ転送速度と転送ワード数から転送時間を算出することは容易であり、それによれば計算時間に転送時間が占める割合は微々たるものであることがわかっている。

問題となるのは、(2)と(3)である。(2)の収束性の劣化は点 SOR 法において顕著に現れ、島野・荒川⁽⁸⁾あるいは坂本・河村⁽⁹⁾が指摘しているように、分割境界近傍の格子点で局所的に減速緩和が施されるのが原因で、収束性が劣化する。加速係数の最適値もプロセッサ数が増加するにつれ小さな値を取らざるを得なくなり、並列化効率が急激に劣化する主要因となっている。(3)の通信同期待ちに関しては、従来より指摘されることが多かったが、通信時間を構成する各要素を確実に分離評価することが困難であったため、定量的な議論はなされていなかった。これらの同期計算における諸問題は、非同期計算においても重要課題と考えられる。

そこで、非同期通信方式を使った計算と同期通信方式によるものと比較することで、点 SOR 法における通信同期待ちが全計算時間に占める割合や、非同期計算方式における収束性の問題を明らかにする。

4.2 検定問題 非圧縮性流体の熱流体数値解析をポアソン方程式で記述される問題に帰着することができる。解析の高速化は収束反復計算の並列化効率を上げることに置き換えられる。そこで、本研究では、ポアソン方程式に焦点を絞り、下記のディリクレ問題に対して、 L_2 残差で 10^{-6} の収束判定条件を与え、検定問題とした。

方程式：

$$\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = -\sin x - \cos y \quad (0 \leq x, y \leq 2\pi) \quad \dots\dots\dots (1)$$

境界条件：

$$u = \sin x + \cos y \quad \dots\dots\dots (2)$$

式(1)を二次精度中心差分により離散化すると、

$$\frac{u_{i+1,j} - 2u_{i,j} + u_{i-1,j}}{\Delta x^2} + \frac{u_{i,j+1} - 2u_{i,j} + u_{i,j-1}}{\Delta y^2} = -\sin x - \cos y \quad \dots\dots\dots (3)$$

である。現在の計算繰返し回数を N とすると、次の繰返し回数 $N+1$ における u の値はできる限り新しい反復段階の値 ($N+1$ の値) を用いれば、

$$u_{i,j}^{N+1} = u_{i,j}^N + \omega \alpha \left[\frac{u_{i+1,j}^N - 2u_{i,j}^N + u_{i-1,j}^N}{\Delta x^2} + \frac{u_{i,j+1}^N - 2u_{i,j}^N + u_{i,j-1}^N}{\Delta y^2} + S_{i,j}^N \right] \quad \dots\dots\dots (4)$$

と表現できる。ただし、

$$\alpha = \frac{\Delta x^2 \Delta y^2}{2(\Delta x^2 + \Delta y^2)} \quad \dots\dots\dots (5)$$

$$S_{i,j}^N = -\sin x - \cos y \quad \dots\dots\dots (6)$$

であり、 ω は加速係数である。計算手順は、以下に示す点 SOR 法の計算手順に従った。計算格子を順にスウィープしながら、各格子点上の繰返し回数 $N+1$ における u の値を求め、全計算空間の格子点計算が終了した時点で収束判定を行い、収束した場合には、その時点での u の値を計算結果とし、未収束の場合には、再び反復計算を繰返すものとする。さらに、計算領域を小領域に分割して各プロセッサに分担させる領域分割法により並列処理を行う。領域分割法では、分割によってできた分割境界値をプロセッサ間で通信し、境界条件を全計算領域に浸透させる。

また、通信方式の違いによる収束性の相違と同期待ちの時間を正確に測定するために、プログラム中では、点 SOR のソルバ部分は全く同一コードとする。分割境界値の通信、および各プロセッサにおける収束判定結果の通信については、収束演算の 1 反復が終わる度に行うものとする。ここで、収束判定については、繰返し演算の 1 反復が終わる度に各プロセッサごとに分割領域の収束判定を行い、収束の場合は true を、未収束の場合は false を収束判定ビットとしてメモリーに格納し、プロセッサ間通信によって、全プロセッサの収束判定ビットの論理積をとることにより実施する。ちなみに、通信回数を最小にすると、 2^n 個のプロセッサ間で論理積をとる場合には、 n 回の通信で全プロセッサの収束判定ビットの論理積を取ることが可能である。

4.3 プロセッサ数と加速係数の関係 図 2 に、

図2 64×64 格子における加速係数 ω と収束反復回数の関係

図3 プロセッサ数と収束反復回数の関係
(図の縦軸は、収束反復回数 N を、64×64 の格子点を1個プロセッサで計算した場合の収束反復回数 N_1 で無次元化している)

非同期通信において、64×64 の格子数を最大 32 個のプロセッサにまで均等に割り振り、加速係数 ω をパラメトリックに変化させた場合の収束反復回数を示している。図2より、プロセッサ数が増加すると、収束反復回数を最小にする ω の最適値は急激に小さくなり、個々のプロセッサが受け持つ格子点数が減少するにもかかわらず、収束反復回数が増加することがわかる。

図4 同期通信と非同期通信での分割境界面のデータ授受の相違(上：同期通信，下：非同期通信)

つまり、複数のプロセッサで非同期のメモリアクセスにより収束演算を行う場合、 ω の値を適正に選定しないと、並列化効率が上がらないばかりでなく、収束解さえ得られない事態が考えられ、計算に際し注意が必要となる。

4.4 非同期計算の収束性 次に、図3は同期計算と非同期計算の収束性の違いを示すもので、PVM、バリエーションとともに反復回数が同じであるため、一括して同期計算として示し、非同期計算と比較したものである。図2から明らかなように、加速係数の違いによって収束反復回数は大きく異なるので、まず ω をパラメトリックに変化させ、 ω の最適値を求めたうえで、その場合の反復回数を各プロセッサ数に対する収束反復回数として定義した。図3中の縦軸は、収束反復回数 N を、64×64 の格子点を1プロセッサで計算した場合の収束反復回数 N_1 で無次元化したものである。図3より、同期計算の収束性が非同期計算の収束性を上回っていることがわかる。ここで、プロセッサ数と収束性の劣化具合の関係をを見ると、プロセッサ数の増加にかかわらず、非同期計算の反復回数は、同期計算の1.2～1.3倍程度であることがわかる。

この収束性の違いは、隣接プロセッサ間でやりとりされる分割境界データの新鮮さに起因する。図4は分割境界面でのデータ授受の様子を、同期計算と非同期計算で比較したものである。点SOR法は、収束反復回数が $N+1$ 回目の計算を行っている場合、隣接点では

きる限り新しい既知の値を用いる。例えば、図4中の斜線の格子点の計算を行うものとする、左および下の格子点の値は $N+1$ 、右と上の格子点の値は N の値で評価する。この計算を、1個のプロセッサで行う場合、ディリクレ条件の下では、境界上の値は恒等的に最新の値 $N+1$ となる。一方、領域分割法の場合、分割することによって生じた分割境界上の値は、プロセッサ間通信で隣接プロセッサより与えられる。同期計算の場合、通信時における各プロセッサの収束反復回数は、プロセッサ間の通信同期によってすべて等しいことが保証されているため、転送されてくる分割境界値は全プロセッサで N の値となる。

一方、非同期計算の場合、アクセスした隣接プロセッサのメモリー上に存在する u の値の繰返し回数は、各プロセッサの計算速度が1クロックでもずれば、 N あるいは N よりも古い値、さらには、 N より新しい値となる可能性がある。したがって、分割境界値は同期計算よりも古い値あるいは新しい値となり、これが原因で、分割境界面での減速緩和または加速緩和が助長され、分割境界面での不安定性から全計算領域の収束性が劣化する。

また、非同期計算を行ううえで興味深い点に、全プロセッサが収束を完了するまでにかかる反復回数は、計算を実行するごとに異なり、いわゆるカオス的な収束挙動を示すことが挙げられる。そのため、図3および図4の結果は、各条件につき5回の計算を行い、その平均値で示している。

4.5 非同期計算の計算速度 最後に、計算時間の比較を行う。図5は、PVM、バリエーション同期、非同期通信による計算時間の違いを、 256×256 の格子点数で比較したものである。PVMは通信のオーバーヘッドが大きいので、計算時間がかなりかかっており、プロセッサ数を増やすと、計算時間に占める通信時間の割合が極端に大きくなって、計算時間がかえって増えてしまう傾向にある。一方、バリエーション同期と非同期通信は、PVMよりも高速の計算速度が得られている。バリエーション同期と非同期通信を比較した場合、ほぼ互角の計算パフォーマンスを示している。図3より、非同期計算のほうが2~3割収束反復回数が多いので、当然計算時間も同程度多いはずであるが、実際には計算時間にほとんど差はない。これは、非同期にメモリーアクセスを行うことで、プロセッサ間の同期待ちが排除されたことによるもので、収束反復回数の増加による遅延要因と、同期待ち排除による加速効果がほぼ同一オーダーであることを意味する。したがって、点SOR法を領域分割法で並列処理し、加速係数 ω を

図5 256×256 格子の計算におけるプロセッサ数と計算時間の関係

最適値に設定して運転した場合、非同期でも同期計算に匹敵する高速演算が可能であることが示された。

4.6 非同期計算における収束判定 ここで、非同期計算における収束判定法についてふれておきたい。従来の同期型の収束判定法の場合、各プロセッサ内での収束判定結果を集計して全計算空間の収束完了を確認する際に、必ずプロセッサ間に同期をかけて通信を行う必要があった。一般に、収束性のよい計算領域を受け持つプロセッサは早期に定常値に近づくのに対し、不安定点を含む領域を担当するプロセッサでは定常値に至るまでに余分な繰返し回数を必要とし、負荷のアンバランスが生じるため、プロセッサ間の同期を取る従来の方法によると、安定領域を受け持つプロセッサの計算パワーが犠牲になっていた。

そこで、非同期の通信方式に着目する。各プロセッサの収束判定終了後の収束判定結果の通信を行うための命令文に、“もしも収束したなら通信を行う”という条件文を付加する。この通信命令へのIF文等の条件文付加は、同期型の通信形式の場合、プロセッサごとの通信コール回数が異なり、デッドロックと呼ばれる通信不能状態に陥る可能性があったため、分割領域内の収束判定ビットがfalseであっても、必ず全プロセッサとの通信を行い、同期回数を同一にしなければならなかった。しかし、非同期型の通信形式の場合、各プロセッサが隣接プロセッサの収束判定結果をハンドシェイクなしに読み出すことができるため、条件付きのプロセッサ間通信が可能となる。つまり、非同期のメモリーアクセスを使用することによって、各プロセッサにおける収束判定を独立して行い、収束判定ビット

が false の場合, 他のプロセッサとの収束判定ビット通信は行わず, すぐに次の反復計算に移り, 通信時間を削除することで, 不安定領域の収束計算を他よりも先行させるプログラミングが可能である。実際, 256×256 の計算格子を 32 個のプロセッサで計算した場合, 最適加速係数 $\omega = 1.88$ において 5.7% の高速化が達成された。

5. 結 論

熱流体数値解析の基本である楕円形偏微分方程式の点 SOR 法による解析を, 領域分割法で並列処理する際, プロセッサ間通信方式を PVM, バリヤーシクロ, 非同期方式の 3 種類で行い, 非同期のメモリアクセス方式によっても同期方式であるバリヤーシクロと同程度の速度で安定に収束解を得ることが可能であることが示された。さらに, 本報告で示したマルチプロセッサ間の非同期通信方式による偏微分方程式解析手法は, (1) プロセッサごとの処理負荷に偏りがある場合, (2) 超細粒度のアルゴリズムを使用したマルチプロセッサシステム, に適用することで画期的な高速演算が可能になると考えられる。

文 献

- (1) Simon, H. D., *Parallel Computational Fluid Dynamics*, (1992), The MTT Press.
- (2) Gropp, G. W. and Smith, E. B., Computational Fluid Dynamics on Parallel Processors, *Comput. Fluids*, **18**-3 (1990), 289.
- (3) Yagawa, G., Yoshioka, A., Yoshimura, S. and Soneda, N., *Int. J. Comp. Struct.*, **38** (1991), 615.
- (4) 吉岡・矢川・吉村・曾根田, 大規模・超高速計算力学のためのネットワーク・コンピューティング手法の開発, 機論, **57**-541, A (1991), 1964.
- (5) 矢川・塩谷, データ分散管理型並列有限要素法, 機講資, VI (1994), 37.
- (6) 稲垣・河村・三上・武田, ボアソン方程式のトランスピュータを用いた並列演算における計算効率, 第 4 回計算力学講演会論文集, (1991).
- (7) 河村・稲垣, トランスピュータを用いた熱・流体数値解析の並列演算, 機論, **57**-540, B (1991), 2702.
- (8) 島野・荒川, 領域分割法を用いた並列計算手法による非圧縮性流れの数値解析, 機論, **59**-567, B (1993), 3340.
- (9) 坂本・河村, MAC 系解法による二次元流れの並列計算, 機論, **60**-569, B (1994), 99.
- (10) 奥田・矢川・永井, 非圧縮性流れ解析へのニューラルネットワークの適用, 機講論, II (1994), 421.
- (11) Baudet, G. M., Asynchronous Iterative Methods for Multiprocessors, *J. ACM*, **25**-2 (1978), 226.
- (12) Wei, J., Parallel Asynchronous Iterations of Least Fixed Points, *Parallel Comput.*, **19** (1993), 887.
- (13) Barlow, R. H. and Evans, D. J., Parallel Algorithms for the Iterative Solution to Linear Systems, *Comput. J.*, **25**-1 (1982), 56.
- (14) Conforti, D., ほか 5 名, A Model of Efficient Asynchronous Parallel Algorithms on Multicomputer Systems, *Parallel Comput.*, **18** (1992), 31.
- (15) Geist, A., ほか 5 名, *PVM 3 User's Guide Reference Manual*, (1993).