ポ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*稲垣 詠一^{*1}, 河村 洋^{*1}Efficiency of Parallel Computation of Poisson Equation
using a Transputer Array

Eiichi INAGAKI and Hiroshi KAWAMURA

A high-speed calculation of the Poisson equation is of an essential importance in the numerical analysis of thermohydraulic problems. The present study aims to examine the efficiency of parallel computation with a transputer array. Geometric parallelism of the two-dimensional Poisson equation is studied using the domain decomposition method divided into 4×4 transputers at most. The efficiency of parallel computation is measured with Parallel FORTRAN by the Jacobi or SOR method in terms of the number of transputers, mesh size and domain boundary communication. An equation to estimate the efficiency of the parallel computation is proposed and compared with the measured values. The results show that the proposed equation is in good agreement with the measured values.

Key Words: Numerical Analysis, Transputer Array, Poisson Equation, Parallel Computation, Efficiency

1. 緒 言

最近, 数値解析の処理能力を高める一つの手段として並列処理が精力的に研究されるようになってきた。いくつかのプロセスを並列に実行するためには, お互いのデータを通信しながら行う必要がある。このような処理を効率良く実行するために, CPS (Communicating Sequential Processes) の概念で設計された演算素子がトランスピュータである。トランスピュータは4つの通信チャンネルを持ち, その組合せによって並列処理システムが容易に構成でき, また拡張も容易である。しかし, 拡張にともない演算素子数が増えるにしたがって, 通信時の同期待ちの影響が並列演算の計算効率に大きく影響することとなる。そのため, トランスピュータを用いて, 有限要素法⁽¹⁾, 熱伝導方程式⁽²⁾についての研究が行われている。また一般に流れ解析の分野においては, 時間に対して陽的に計算可能な圧縮性流体について, Gropp-Smith⁽³⁾がAlliantなど, さらに大型の並列計算機における計算効率の検討を行っている。

ポアソン方程式は, 数多くの物理現象を記述する基本的な方程式型の一つである。特に熱・流体工学の分野においては, ポアソン方程式は非圧縮性流体のナビエ・ストークス方程式の解法や定常熱伝導方程式に現れ, 数値計算上も多くの計算時間を費やす部分となっている。したがって, ポアソン方程式の計算の効率化を図ることは, 熱・流体工学をはじめとする広い工学分野の数値解析の高速化に対する寄与が大きい。ポアソン方程式の解法には, いくつかの方法があるが, なかでもその簡便さからSOR系の解法が用いられることが多い。しかもSOR系の解法は前進(後退)逐次代入のような処理を含まないために, 並列演算に適した解法であると考えられる。そこで本研究は, SOR法によるポアソン方程式の計算効率について検討することを目的とする。

著者らはすでに4台のトランスピュータを用いた並列システムで熱流体の数値解析を行い, 計算効率や速度向上率について報告してきた⁽⁴⁾。本研究ではトランスピュータを16台に拡張し, ポアソン方程式の並列演算の計算効率を領域分割法によって求め, 通信順序やプロセッサ数の影響について検討し, 並列演算の計算効率を予測する近似式を提案する。

* 平成3年11月14日 第4回計算力学講演会において講演, 原稿受付 平成4年2月21日。

^{*1} 正員, 東京理科大学理工学部(〒278 野田市山崎東亀山)。

記 号

- a : 一格子点の計算時間 s/点/ステップ
 b : 一格子点の分割境界の通信時間 s/点/ステップ
 c : 収束判定フラグの通信時間 s/ステップ
 E_p : 計算効率 $= \frac{T(1)}{PT(P)}$
 m : 分割境界通信回数(最大値)
 $P=2^2$ のとき $m=2$
 $P \geq 3^2$ のとき $m=4$
 N : 一辺の格子点数
 N_m : $P=2^2$ のとき $N_m=N-2$
 $P \geq 3^2$ のとき $N_m=N$
 n : 反復回数(ステップ数)
 P : プロセッサ数
 $T(1)$: 1台のプロセッサによる計算時間 s
 $T(P)$: P 台のプロセッサによる計算時間 s

2. 使用機器とテスト問題

ポアソン方程式としては前報⁽⁴⁾と同じ二次元の熱伝導問題を対象とし、一辺の長さ1の正方形の領域に熱源 Q を加えて温度分布を求める問題を取り上げる。

$$\lambda \left(\frac{\partial^2 T}{\partial x^2} + \frac{\partial^2 T}{\partial y^2} \right) + Q = 0 \quad \dots\dots\dots (1)$$

$$Q = \sin(\pi x) \sin(\pi y), \quad (0 \leq x, y \leq 1)$$

境界条件: $T=0$ (全境界上)

式(1)を差分化し、パラレルフォートラン⁽⁵⁾⁽⁶⁾を用いてヤコビ法とSOR法による計算コードを作成した。収束条件は各計算点において各ステップ毎の変分の最大値が 10^{-5} 以下とした。格子点数は 60×60 と 120×120 である。

トランスピュータボード B 003(T 800 $\times$ 4, 各トランスピュータは 256 K の DRAM を内蔵, 通信速度 1

図1 トランスピュータの格子接続

Mbps)を4枚使用して、図1に示すようにT 800を最大 4×4 の正方配列に接続した。並列演算の方法は領域分割とし、領域をトランスピュータと同数に正方格子に分割し、均等な格子点数をおのおののトランスピュータに割り当てた。計算はまずヤコビ法によって行った。これはヤコビ法では各格子点の計算に1ステップ前の値のみを使う逐次緩和法であり、領域の分割法に依存しなくなるためである。この場合、計算の流れは、①分割境界値の通信、②各格子点の計算、③収束判定のためのフラグ通信であり、各ステップごとに上記の①、②、③を繰り返す。SOR法については後の考察において検討する。

3. 測定結果と計算効率

3.1 測定結果 正方形領域を均等に分割するので、各プロセッサの格子点計算の負荷はほとんど差がなく、計算効率に影響するのは、分割境界の通信と各ステップごとの収束判定のためのフラグ通信である。まず、領域を16分割した場合について、分割境界通信の回数を最小4回にした場合と、一例として12回にした場合、さらに毎回收束判定をする場合(ケース1)としない場合(ケース2)の計算時間を測定し、計算効率 E_p を比較した。結果を図2に示す。

3.2 分割境界通信 ケース1において、分割境界通信が12回の場合に4回の場合に比して計算効率が低下しているのは、フラグ通信に要する時間のため収束判定の同期待ちが大きくなるためと考えられる。一方、ケース1の分割境界通信が4回の場合とケース2の場合には、計算効率にほとんど差が現れない。この理由についてつぎに検討する。まず、分割境界通信

図2 計算効率(領域16分割)

が12回で、かつ毎回収束判定を行わない場合について、16個のプロセッサ(A~P)の並列演算のタイムチャートを図3に示す。プロセッサA~Pが分割境界通信と格子点計算を繰返しつつ並列に走っていることを表している。図中○印つき点線は各プロセッサがどのようなタイミングで分割境界の通信を行っているかを示し、矢印線は格子点計算を表す。正方領域を均等に分割しているので、各プロセッサの矢印線の長さは同一と仮定している。計算開始時点から、まず分割境界通信を完了したプロセッサは、順次、次のステップの格子点計算を実行する。

一例として分割領域の四隅に配置されるプロセッサPとAに着目してみる。Pは計算開始時点でLとOとの2回分の分割境界通信を終えるとすぐに、ほかのプロセッサと独立に格子点計算を実行する。計算を終えると、再びLと分割境界の通信をしなければならないが、この時点ではまだLプロセッサが格子点の計算中であるから、終了時点まで待たされる(同期待ち)。次のPとOの分割境界通信でも同じことが起こる。結局Pプロセッサは2回の同期待ちと2回の分割境界通信の後に、第2ステップ以降の格子点計算を実行できることになる。

また、Aプロセッサは、計算開始時点でPと同様に2回の分割境界通信後、格子点計算を実行し、次のA-B、A-E間通信のときに通信相手が他のプロセッサと通信中であるために待たされる点が異なる。分割領域の側面に配置されるプロセッサ(B, C, E, I, H, L, N, O)でも、3回の分割境界通信と、通信先のプロセッサが計算中かまたは通信中のために1回の待ち時

間を生ずる。結局、分割境界通信が12回の場合であっても、反復が2回目以降では、どのプロセッサについても4回の分割境界通信時間と格子点計算時間の和を1サイクルとして逐次計算のステップを n 回繰返すこととなる。

さて、図4(a), (b)に分割境界通信回数を整理した場合の通信順序を示す。通信順序1, 2, 3, 4は互いに交換可能である。隣り合うプロセッサ以外は互いに独立に通信ができるため、プロセッサ数が9以上であれば分割境界通信は最小4回となる。したがって、分割境界通信が4回で、収束判定を行わない場合には、図3の図中の◇時点までの12回の分割境界通信部分と図4(a)の分割境界通信タイムチャート部分とを置き換えれば、上述の考察がそのまま成立する。すなわち、四隅のプロセッサと側面のプロセッサは、それぞれ2回と1回の通信待ち(同期待ち)が起こるため、どのプロセッサも計4回の分割境界通信と格子点計算のサイクルを繰返すことになる。

3.3 測定結果の考察 以上のことから、図1のケース1とケース2の場合のタイムチャートの概略図は図5のように表される。

収束判定のためのフラグ通信は各領域に分割された全プロセッサのANDゲートに対応し、フラグの整数値の通信である。トランスピュータの通信速度は非常に大きく、このような整数値のフラグ通信時間は分割境界値のデータ通信あるいは格子点計算時間に比して極めて小さいので、図5ではパルスとしてフラグ通信を表している。この時点で全プロセッサは、強制的な同期待ちを受ける。繰返しステップごとに収束判定を

図3 並列演算のタイムチャート
(分割境界通信12回)

(a) 通信順序

(b) 境界通信順序(最小4回)
(通信回数を整理して4回とした場合)

図4 分割境界通信の順序

受けるケース 1 の場合は、ケース 2 と異なり分割境界通信時には同期待ちを受けず、格子点計算後の収束判定時に同期待ちを受け、1 サイクル(分割境界通信と格子点計算)が最大の計算時間に全体の並列演算時間が支配されるのが特徴である。したがって、ケース 1 の分割境界通信回数が 12 回の場合が最も計算効率が低下する。一方、ケース 1 の分割境界通信を 4 回に整理した場合の 1 サイクルは、図 5 から明らかなように、ケース 2 の分割境界通信が 12 回と 4 回の場合と同じになり、計算効率にほとんど差が現れない。わずかの差は計算開始時点の分割境界通信回数と終了時点における計算時間に起因しており、格子点数が大、すなわち繰返しステップ数が大きくなるほど 4 回の分割境界通信と格子点計算のサイクルが支配的となり、この差は無視できるオーダとなることを示している。結局、分割境界通信を 4 回に整理し、かつ収束判定を行わない場合が、最も計算効率が良くなっている。

4. 計算効率の予測式

4.1 予測式の導出 計算効率と領域分割数、格子点数、分割境界通信の関係について格討する。まず、

(a) 繰返し計算のフローチャート

(b) 並列演算のタイムチャート

図 5 並列演算のタイムチャート概略図

反復に依存する演算時間について考える。今回は境界条件としてディリクレ条件を与えているため、実境界の演算を除けるから、格子点数は $(N-2)^2$ となり、並列演算を行わない場合の計算時間 $T(1)$ は、

$$T(1) = na(N-2)^2 \quad \dots\dots\dots (2)$$

で表される。

次に、 P 台のプロセッサによる正方分割のときの並列演算時間について考える。計算領域内の格子点を P 個のプロセッサが分担するから、各領域の一辺の格子点数は $N/\sqrt{P}$ となる。このとき図 6 に示すように、格子点計算では実境界値を計算する必要がないから、実境界に接するプロセッサと接しないプロセッサでは格子点数が異なる。 $P=2^2$ のときは、各プロセッサはそれぞれ 2 辺を実境界に接しているから、格子点数は $[(N-2)/\sqrt{P}]^2$ であり、 $P \geq 3^2$ では実境界に接しない中央部のプロセッサの格子点数が最大になり、 $(N/\sqrt{P})^2$ となる。ここで、一辺の格子点数が最大値となるプロセッサに着目する理由は、受け持ち格子点数の少ないプロセッサも同期待ちのため格子点数最大のプロセッサに格子点の計算時間が支配されるためである。以上を統一的にまとめて領域分割数によらず分割境界の一辺の格子点数を $N_m/\sqrt{P}$ と表せば、($P=2^2$ のとき $N_m=N-2$, $P \geq 3^2$ のとき $N_m=N$)、格子点の計算時間は $a(N_m/\sqrt{P})^2$ となる。

以上より、並列演算時間は格子点の計算時間と分割

図 6 計算効率の予測式

境界の通信時間とフラグ通信時間の和を n 回反復するから

$$T(P) = n[a(N_m/\sqrt{P})^2 + mbN_m/\sqrt{P} + c] \quad \dots\dots\dots(3)$$

で表わされる。式(2)、(3)を計算効率 E_p の定義式に代入して、

$$\begin{aligned} E_p &= T(1)/PT(P) \\ &= \frac{na(N-2)^2}{Pn[a(N_m/\sqrt{P})^2 + mbN_m/\sqrt{P} + c]} \\ &= \frac{1}{\frac{N_m^2}{(N-2)^2} + \frac{bmN_m\sqrt{P}}{a(N-2)^2} + \frac{cP}{a(N-2)^2}} \end{aligned} \quad \dots\dots\dots(4)$$

ここで、 c/a (フラグ通信時間と一格子点の計算時間の比)は十分小さいので、分母の第3項は無視すると、結局、計算効率は

$$E_p = \frac{1}{\left(1 + \frac{b}{a} \frac{m\sqrt{P}}{N_m}\right) \left(\frac{N_m}{N-2}\right)^2} \quad \dots\dots\dots(5)$$

と表される。ここで、分母の $N_m/(N-2)$ は各分割領域の格子点数の不均一の度合いを表し、本研究のように正方領域を等分割する場合には、 $P=2^2$ のときは1、 $P \geq 3^2$ のときでも、比較的少ないプロセッサに等分割する場合には、ほとんど1に近い。このように、計算効率は格子点数やプロセッサ以外に一格子点の分割境界通信時間と演算時間の比である b/a に大きく依存することとなる。

実際の計算時間は領域分割数と格子点数を変えて測定した。測定結果と式(5)(点線)の比較を図7、8に

示す。測定結果より、 b/a は、図7の格子点数が 60×60 、 $P=4^2$ のときを基準にして、 $b/a=0.7657$ を得る。

図7でA、B線は分割境界通信の回数の差と計算効率の関係を示している。図7中点線が式(5)を示す。特にB線が $P=2^2$ のとき相対的に計算効率が良くなっているように見えるのは、図6に示すとおり、 $P=2^2$ ではどのプロセッサも2辺を境界に接するという特殊条件にあるため通信回数が最小2回となることによる。そこで、プロセッサ数 $P \geq 3^2$ の時と同様に分割境界通信をあえて4回行った場合がA線である。一格子点あたりの通信時間 b を大きくした場合に対応し、計算効率は大きく低下する。図1で分割境界の通信回数を12回にしたとき著しく計算効率が低下する場合も式(3)で説明できる。逆に、C線は各分割領域での格子点演算を4回繰返し $a \rightarrow 4a$ した場合である。一格子点あたりの計算量が大きくなる場合に対応し、計算効率は良くなる。これは図8に示した格子点数をパラメータ(格子点数 60×60 と 120×120)にしたときの計算効率と同じ関係になっている。各プロセッサが独立に計算できる格子点数が増加するほど計算効率が上がることを示している。図7、8とも式(5)は並列演算の計算効率を良く表現している。

以上により正方形領域を等分割するこの種の並列演算では、いったん少数のプロセッサ(たとえば $P=2^2$)で b/a を知れば、より大きな P 台のプロセッサに対しても、実際にそのようなプロセッサを組むことなく並列演算時間を予測できる。とくにプロセッサ数 P を一定にして一辺の格子点数 N を増加させると、計算効率は

図7 分割境界通信と格子点計算の影響

図8 実測値と予測式の比較

表 1 SOR 法による収束回数

Domain	Mesh 60 x 60		Mesh 120 x 120	
	Iteration Number		Iteration Number	
	Q ≠ 0 Dirichlet	Q = 0 Dirichlet	Q ≠ 0 Dirichlet	Q = 0 Dirichlet
1 x 1	1747	2474	5103	8096
2 x 2	1773	2510	5140	8144
3 x 3	1781	2528	5149	8173
4 x 4	1793	2553	5169	8218

$$E_p \cong \frac{1}{1 + \frac{bm\sqrt{P}}{aN}} \cong 1 - \frac{bm\sqrt{P}}{aN} \dots\dots\dots (6)$$

にしたがって増大する。逆に N を一定にしてプロセッサ数 P を増加させる場合には、計算効率 E_p は減少する。

4.2 SOR 法の格計 次に、SOR 法について検討する。SOR 法は格子点の計算にそれまでに更新された値を使うので、収束回数は領域の分割の仕方に依存すると考えられる。そこで、格子点数が 60×60 と 120×120 の場合でかつ、内部発熱のある式(1)の場合と内部発熱 Q はゼロで境界値が非ゼロである場合についてそれぞれ繰返し回数を測定した。結果を表1に示す。今回は過緩和による計算の加速を行うことは主眼ではないため、SOR 法の過緩和係数は格子点によらず1とした。したがって、今回の解法はガウス-ザイデル法となっている。収束条件はいずれも 10^{-5} である。表1から分かるように、たしかに SOR 法では分割数が増加するにしたがって繰返し回数が増大している。いま、初期値をゼロとして計算を開始する場合を考えると、内部熱源の場合には繰返しステップごとに全領域で値は増加するため、隣接値の値が更新された値であるか、1ステップ前の値であるかの差はそれほど大きくない可能性も考えられた。そこで内部発熱をゼロとし、境界値に非ゼロの値を指定した場合の計算を行った。しかし、測定の結果、反復回数の増加は、内部発熱、境界値指定にはほとんど依存せず、格子点数

が 60×60 の場合で、たかだか3%以内であり、格子点数が 120×120 の場合には1.5%以下であった。ただし、並列の度合いが非常に大きくなり、一つのプロセッサが分担する格子点数が極端に小さくなれば、この影響は無視できなくなるであろう。

5. ま と め

最大16台のトランスペュータを用いて二次元ポアソン方程式の領域分割法による並列化を行った。まず、正方形領域を16分割した場合の並列演算の計算効率を求め、分割境界通信の順序をタイムチャートを用いて検討した。その結果、プロセッサ数が9以上のときは分割境界の通信を最小4回に整理することができ、また、タイムチャートは分割境界通信や格子点計算の並列演算をよく説明できる一つの手段であることを示した。

次に、領域分割数と格子点数を変えて計算効率を求め、計算効率の近似式を提案した。この近似式は正方形領域を等分割する領域分割法の並列演算の計算効率を良く予測しており、分割境界通信時間と格子点計算時間の比が重要なパラメータであることを示した。さらに逐次緩和法の解法と計算効率の関係について考察した。

本研究は文部省科学研究費の一部を用いて行ったことを付記する。計算の実行については、本学学生の島田祐滋、三上 隆、武田芳和君の協力を、また、並列演算アレーの構築にあたってはSGS トムソン(株)のご協力を得た。

文 献

- (1) 三好・高野, 機講論, 第3回計算力学講演会, No. 900-69, (1990), 189.
- (2) 彦坂・井門, 文献(1)の209ページ.
- (3) Gropp, G. W. and E. D. Smith, *Computers & Fluids*, 18-3(1990), 289-304.
- (4) 河村・稲垣, 機論, 57-540, B(1991), 222.
- (5) *Parallel Fortran User Guide*, (1988), 3L Ltd.
- (6) IMS Technical Note 6, *INMOS Limited*, (1988).