高水準数値シミュレーション言語 DEQSOL の並列計算機向けトランスレータ

大河内 俊夫† 金野 千里† 猪貝 光祥††

数値シミュレーション分野での並列計算機の利用が，高い性能と価格性能比の良さから注目されている．しかし，現在の分散メモリ型並列計算機のプログラミング環境のもとでは，ユーザがデータの分割，処理の分割，通信の記述等を行う必要があり，プログラム開発の工数が膨大になる．これが並列計算機の幅広い普及の障害になっている．一方，並列計算機のプログラミングの困難さを克服するための手段として，高水準の記述からの並列化がある．DEQSOL (Differential Equation Solver Language) は，熱拡散，流体等の物理現象の数値シミュレーションを対象とした専用高水準言語システムである．DEQSOL プログラムから分散メモリ型並列計算機用の FORTRAN プログラムを生成する並列化トランスレータを試作し，並列計算機 nCUBE 2 上で評価を行った．131,000 格子点の 3 次元熱拡散問題に適用した場合，16 プロセッサを用いて，14.3 倍の高速化を実現した．

High Level Numerical Simulation Language DEQSOL for Parallel Computers

Toshio Okochi,† Chisato Konno† and Mituyoshi Igai††

Massively parallel computers have been shown to be efficient for numerical simulation of PDE (Partial Differential Equation) problems. However, it is very time consuming to develop simulation programs for them. This is because current programming languages require the programmer to specify a problem to be solved at a low level of abstraction. An alternative approach is to specify the problem to be solved at a high-level. DEQSOL (Differential Equation Solver Language) is a high-level programming language specially designed to improve programming productivity for PDE problems including heat diffusion and fluid dynamics. This paper describes the transformation technique to generate FORTRAN programs that can invoke high performance of parallel computers by using the implicit parallelism in the DEQSOL description for the finite difference discretization method (FDM) facility. Preliminary results show that for grids consisting of 131,000 points, speed-up in excess of 14.3 can be achieved on an nCUBE 2 system by using 16 processors.

1. はじめに

熱拡散や流体などの数値シミュレーション分野での並列計算機の利用が，高い性能と価格性能化の良さから注目されている．しかし，現在の分散メモリ型並列計算機，特に MIMD (Multiple Instruction Multiple Data) 型の並列計算機では，プログラミングは FORTRAN，C などの言語から並列制御ライブラリ群を呼び出す方式が主流になっている．このような環境のもとでは，ユーザがデータの分割，処理の分割，通信の記述等の作業を行う必要があり，プログラム開発の工数が膨大になる．さらに並列計算機の能力を十分に発揮するプログラムを開発するためには，計算機の性能に関する諸パラメータやプログラムの計算負荷等について理解したうえで適切な負荷分散や通信のスケジューリングを行う必要がある．数値シミュレーションのプログラムは通常比較的単純な制御構造を有しているが，それでも一般の数値計算ユーザにとっては，こうした作業を必要とする状況は，並列計算機を利用する上で重大な障害になっていると考えられる．この並列計算機のプログラミングネックの問題に対しては，FORTRAN，C 等の言語からの自動並列化を行う方法，あるいは FORTRAN，C 等の言語に並列制御のための指示文を入れた言語から自動並列化を行う方法の研究が進められている[8)~10),17)~20)]．しかしこれらは，現時点では，データ参照関係がコンパイル時に静的に

† (株)日立製作所中央研究所
Central Research Laboratory, Hitachi Ltd.
†† (株)日立超 LSI エンジニアリング
Hitachi ULSI Engineering Ltd.

解析可能である等の制約があり[18]，プログラマがそれを理解してコーディングしなければならない，といった課題がある．一方，数値シミュレーション等の分野では，対象とするモデルを直接的に表現するような言語によってこの問題を解決することに期待が持たれている[1]．数値シミュレーションに現れる計算は，多くの場合物理現象に元来内在する並列性を有しており，物理現象を直接的に表現できる言語であれば，プログラミングの負担を軽くすると同時に，処理系は物理現象自体がもっている並列性を利用して高性能な並列化を実現できると考えられる．

本研究の目的は，数値シミュレーション分野を対象にした高水準言語 DEQSOL を並列計算機に実装することにより，この分野で並列計算機を容易に利用できる環境を提供することにある．DEQSOL の並列計算機向けトランスレータの実現方式，およびその評価実験結果について報告する．

2. DEQSOL の概要

DEQSOL (Differential Equation Solver Language) は，熱伝導解析，流体解析などの物理現象の数値シミュレーションを対象とした専用高水準言語システムである[2)~4)]．当初，プログラミングの工数削減とベクトル型スーパーコンピュータの利用を容易にすることを目的として開発された．DEQSOL では，ユーザは，形状・境界条件・初期条件等の物理モデル，空間メッシュ・時間刻み，計算アルゴリズム等の数値計算モデルを簡潔に表現したプログラムを作成する．計算アルゴリズムは，微分方程式レベルの数式記述によって記述される．このプログラムを DEQSOL トランスレータが FORTRAN の数値計算プログラムに変換する．DEQSOL トランスレータはまず微分方程式レベルの数式記述を離散化処理により有限個の解析格子点上に定義された変数の計算に変換し，並列化処理により並列実行のためのデータ分割，処理の分割を決定し，コード生成処理により FORTRAN プログラムを生成する（**図 1**）．

図 2 にシミュレーション対象のモデルの例を，**図 3** にその DEQSOL による記述をそれぞれ示す．DEQSOL による問題記述は以下の部分から構成される．

（a）解析対象の物理モデルの記述（図 3，ステップ 1～2，5～10）

DOMAIN 文（ステップ 1）は解析対象領域を座標値で記述し，TIME 文（ステップ 2）は解析する時間範囲の始点，終点を指定する．REGION 文（ステップ 5）は部分領域を定義する．ここで定義した部分領域名称は，境界条件等の記述に用いる．CONST 文，VAR 文，SVAR 文（ステップ 6, 7, 8）は変数，定数

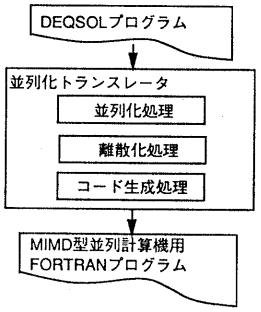


図 1 DEQSOL の処理フロー
Fig. 1 Processing flow of DEQSOL.

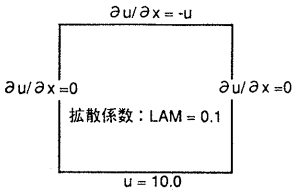


図 2 シミュレーションの対象
Fig. 2 An example of numerical simulation model.

```
DOMAIN X=[0:1],Y=[0:1];                              (1)
TIME T=[0:100];                                      (2)
TSTEP DLT=[0(0.0005)100];                            (3)
MESH X=[0:1:100],Y=[0:1:100];                        (4)
REGION TOP=((0:1),1),BOT=((0:1),0),                  (5)
 LEFT =(0.0,*) ,RIGHT=(1.0,*);
CONST LAM=0.1;                                       (6)
VAR U,UOLD,DLTU;                                     (7)
SVAR EPS1;                                           (8)
BCOND UOLD=0.8  AT LEFT,,                            (9)
 UOLD=0.5  AT RIGHT
  DY(UOLD)=0.0  AT TOP+BOT,
  U:=UOLD;
ICOND  UOLD=1.0,U:=UOLD;                            (10)
SCHEME;
  ITER NT UNTIL [EPS1 LE 1.0D-5];                   (11)
    U=UOLD+DLT*(-DIV(-LAM*GRAD(UOLD)));             (12)
    DLTU=(U-UOLD)/UOLD;                             (13)
    CALL NORM(EPS1,DLTU);
    UOLD=U;                                         (14)
  END ITER;                                         (15)
END SCHEME;
```

図 3 DEQSOL による記述例
Fig. 3 An example of DEQSOL description.

を宣言する．DEQSOL ではこれらの宣言文によって，温度や圧力のような空間領域上で定義される物理変数を宣言することができる．このような変数は離散化処理により，FORTRAN プログラム中では有限個の解析格子点上に定義された値の配列に変換される．BCOND文(ステップ9)は境界条件を記述し，ICOND文（ステップ10）は初期条件を与える．

（b）数値解析モデルの記述（図3，ステップ3，4）

TIME 文（ステップ3）は差分法の解析格子を，各座標軸方向の格子点数で指定する．TSTEP 文（ステップ4）はシミュレーションの時間刻みを指定する．

（c）微分方程式レベルの解法アルゴリズムの記述（図3，ステップ 11〜15）

解法アルゴリズムの記述は，物理変数に対する代入文や繰り返し制御，条件文等を用いて行う．ITER 文，END ITER 文（ステップ 11，15）は繰り返しループを記述する制御文である．ステップ 12 からステップ14までの代入文の列は計算手順の記述であり，このうちステップ 12 は拡散方程式の陽解法表現である．アルゴリズムの記述には一階，二階の微分演算子を使うことができる．プログラム中の DX，DY は，それぞれ通常の表記における $\partial/\partial x$，$\partial/\partial y$ を表す．

DEQSOL はベクトル型スーパーコンピュータに対して，平均して 95% 程度の高いベクトル化率を実現している[2)~4)]．本研究では DEQSOL を，数値シミュレーション分野で重要性を増しつつある分散メモリ型並列計算機のプログラミング言語として提供することを目的として，DEQSOL による問題記述プログラムから並列計算機用の FORTRAN プログラムを自動生成する並列計算機向け DEQSOL システムの試作，評価を行った．DEQSOL の問題記述プログラムは，シミュレーションを行う物理モデルが直接的に表現されており，これをもとに解析領域を並列計算機の各プロセッサに割当てることにより，問題の持つ本質的な並列性を生かした並列実行が可能となる．なお，DEQSOL は離散化手法として差分法，有限要素法を有しているが，本報告では離散化手法を差分法に限定している．解析領域の次元は1〜3次元までを適用範囲としている．

3. 関連する研究

並列計算機を対象とした，DEQSOL と類似の目的を持った高水準言語システムとしては，慶応大学の DISTRAN[5)]，米国 Purdue 大学の ELLPACK[6)] がある．DISTRAN は数値シミュレーションに関する特別な知識を持たないユーザを主な対象とし，標準的な計算アルゴリズムを組み込んでいるのに対し，DEQSOL はユーザによる計算アルゴリズムの柔軟な記述を可能にしている点に特徴がある．また，ELLPACK は DEQSOL と類似した言語仕様であるが，処理方式として，標準的な解法ライブラリ群を並列化したものが用意されていて，それを呼び出す方式を採っており，DEQSOL のような自動並列化機能は有していない．

4. DEQSOL トランスレータの並列化方式

4.1 基本方針と並列化のための課題

並列実行方式としては，解析対象領域を分割して各要素プロセッサに割り付け，各要素プロセッサが自分に割り付けられた領域内の解析格子点上の変数の定義処理を担当する方式を採る．並列実行方式としてはこれ以外にもさまざまなものが可能と考えられるが，問題の持つ並列性を生かした方式として最も自然であり，数値シミュレーション分野の並列処理方式として実績のある方式でもあるので，これを踏襲した[11)~14)]．

並列実行方式をこのように決めたとき，DEQSOL トランスレータにおいては，領域分割の決定方法，およびプロセッサデータ通信の自動生成方法が高い実行性能を実現するための鍵になる．DEQSOL は，ユーザによる柔軟な計算アルゴリズムの記述を可能としている点に特徴があり，したがって記述されたアルゴリズムに応じて，それに適した領域分割やプロセッサ間通信の実行方式を決定する必要がある．この点が，アルゴリズムがあらかじめ組み込まれているシミュレーションパッケージや，数値計算ライブラリなどの場合と本質的に異なっている．以下 4.2 節で領域分割決定方法について，4.3 節でプロセッサ通信の自動生成方法について説明する．

4.2 領域分割決定方法

上述したように空間領域の分割に基づいてデータおよび処理の分割を行う場合，使用可能なプロセッサ台数，シミュレーションモデルの形状，計算アルゴリズムに応じて，最も効率良く計算が実行できるように領域分割方式を決定することが課題となる．また，大規模な並列計算機を使用する場合，必要以上に多くのプロセッサを用いると，通信のオーバーヘッド等のために実行性能の低下を招くこともある．このため使用するプロセッサの台数を適切に決定することも必要になる．

本システムでは，DEQSOL プログラムの実行過程を仮想的にトレースして実行時間を推定し，この評価に基づいて実行時間が最小になると推定される領域分割方式を選択する方法をとった．すなわち，領域分割方式の複数の候補について，浮動小数点演算性能，通信の性能などの並列計算機の主要な性能パラメータをもとに実行時間を概算し，それが最小になる分割方式を選び出す．実行時間を推定するための方法，および領域分割方式の候補を選び出す方法について以下で説明する．

4.2.1 実行時間推定

DEQSOL プログラムの並列実行において現われる演算は，(1)格子点ごとに並列に実行されるデータ並列演算，(2)総和計算，(3)各プロセッサの担当領域の境界データの通信，からなる．まずこれらの基本演算の実行時間を推定するためのモデルを構築する．プログラム全体の実行時間は，これら基本演算に要する時間を累計することで求める．そのために，計算機の性能に関して以下の仮定を設ける．

(仮定1) 浮動小数点演算性能

浮動小数点演算の所要時間を一定値 e と仮定する．またループ制御のオーバーヘッドとして，一つのループ当たり一定値 C を仮定する．例えば次のようなFORTRAN プログラムの浮動小数点演算を考える．

```
      DOUBLE PRECISION A(L), B(L)
      DO 10 I=1, L
10    A(I)=A(I)+B(I)**2
```

この計算の所要時間は次式で計算される．

$$2Le+C.$$

浮動小数点演算の性能は一般にはメモリアクセスの性能などにも大きく依存するため，このような仮定は一般には成り立たない．しかし，ここで対象としているのは DEQSOL トランスレータが出力する FORTRAN プログラムのみであり，浮動小数点演算とメモリアクセス回数の比率，アクセスするメモリアドレスに共通した特徴を有している．また nCUBE 2 の要素プロセッサはキャッシュメモリをもたない構成のため，メモリアクセス速度はほぼ一定している．このため，このような仮定でも十分な精度で実行時間推定を行うことができる．

(仮定2) 通信性能

データ長 L のデータを送信，受信するのに要する時間を，T_s および β をシステムに固有の定数として，

$$T(L)=T_s+\beta L.$$

と仮定する．この仮定は nCUBE 2 における実測データ[7]に基づいている．

この仮定の下で，格子系と領域分割が与えられたときの上記3種類の基本演算の実行時間の算出方法を以下簡単な例で示す．総プロセッサ数を n_p とし，各プロセッサが2次元領域上の $n_x n_y$ 個の格子点からなる部分格子の計算を行う場合を考える．各基本演算の実行時間は，以下に示すような式で推定できる．

(1) データ並列演算

DEQSOL ではデータ並列演算を自然な2重ループのプログラムで実行する．1格子点当たり f 回の浮動小数点演算を要する計算を実行とするときの実行時間は次の式で計算される．

$$M=n_y(n_x fe+C).$$

(2) 総和計算

総和計算は，各プロセッサが部分和を計算し，二進木型の通信で総和計算および結果の分配を行う．実行時間は次の式で計算される．

$$M=T(1)2\log_2 n_p+(n_x n_y).$$

(3) 各プロセッサの担当領域の境界データの通信

隣接する部分領域の数が最も多いプロセッサが，実行時間が最長になる．簡単のため格子状の領域分割を仮定し，上下左右のプロセッサと通信を行うとすると，通信実行時間は送受信を合わせて次の式で計算される．

$$M=2(2T(n_x)+2T(n_y)).$$

並列実行過程のトレースを行う場合，一般には上記の仮定以外に通信のレイテンシ (latency：データが通信ネットワークを伝わって転送元から転送先に届くまでに要する時間)を考慮する必要がある．しかし，レイテンシは仮定2に示した通信のためのプロセッサの処理に要する時間に比べ相対的に小さいこと[7]，および4.3節で述べる通信のスケジューリングによって実際にはレイテンシをほとんど隠蔽できることから，本システムでは実行時間推定においてはこれを無視した．

4.2.2 推定精度の検証

上記の実行時間推定方式の精度を検証するために，2次元拡散問題 (21×21 格子点) を例に，いくつかの異なる領域分割で実行した場合の実際の実行時間と，上記の方法による推定値との比較を行った．nCUBE2 を用い，2，4，8台のプロセッサを用い，x 軸方向，y 軸方向の分割数の組み合わせを変えた場合について，推定実行時間と実測時間を**図4**に示す．性能に関するパラメータとしては，典型的な例題の実測値に基

づいて，以下の値を用いた．

$T_s=80\ \mu$sec. $\beta=0.063\ \mu$sec/byte.

$e=2\ \mu$sec. $C=50\ \mu$sec.

領域分割は，**図5**に例示するような均等分割を用いている．図4中の n_{px}, n_{py} はそれぞれ，x 軸方向，y 軸方向の分割数を示している．

上記の例では，推定値と実測値が 15% 程度の変動幅の範囲で一致している．相対的に実行効率の良い領域分割を選び出すというここでの目的のためには十分な精度であると考えられる．

4.2.3 領域分割方式の決定

領域分割方式の候補として，各座標軸方向を演算負荷が均等になるように分けた格子状の分割を基本とする．これは，領域分割を複雑にすると通信相手となるプロセッサ数が増加し通信処理に係わる負荷が増大するために，性能向上が期待しにくいからである．その上で，計算対象領域の形状が複雑なために格子状の分割では各プロセッサの負荷の差が非常に大きくなってしまう場合に限り，部分的補正を加えた分割方式を候補に加える．そしてこれらの候補について実行時間を推定し，推定実行時間が最小となるものを選択する．以下に2次元の場合について使用するプロセッサ数と領域分割を決定するアルゴリズムを示す．ここで N_p は利用可能な最大のプロセッサ数，P_{ij} は格子状に分割された領域のうち x 軸方向 i 番目，y 軸方向 j 番目の部分領域を担当するプロセッサ，L_{ij} は P_{ij} の浮動小数点演算の負荷を示す．また T_s は 4.2.1 項で通信性能の仮定に用いた定数である．一番外側のループにおいて，$n_{px}n_{py}\leqq N_p$ となるすべての整数の組 (n_{px}, n_{py}) を評価対象とすることによって，実際に使用するプロセッサ数も選択している．

```
E=+∞;
for (npxnpy≦Np となる整数の組 (npx, npy) について)
{
    y 軸方向を演算負荷均等に npy 個に分割;
    x 軸方向を演算負荷均等に npx 個に分割;
    実行時間 E(npx, npy) を推定;
  if(E(npx, npy)<E)
    {領域分割方式を記録; E=E(npx, npy);}
  for(j=1;j<npy;j++){
    if(Lij(i=1～npx) の最大値と最小値の差
      >通信回数 *Ts)
    Pij(i=1～npx) の担当領域を演算負荷均等に分
      割;
    実行時間 E(npx, npy) を推定;
    if(E(npx, npy)<E)
      {領域分割方式を記録; E=E(npx, npy);}
  }
}
```

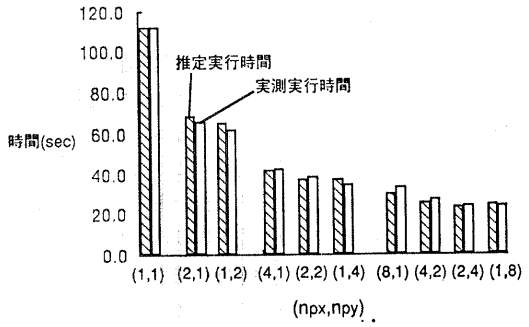


図 4 2次元熱拡散問題での推定実行時間と実測実行時間
Fig. 4 Actual and estimated time for 2-dimensional heat diffusion problem.

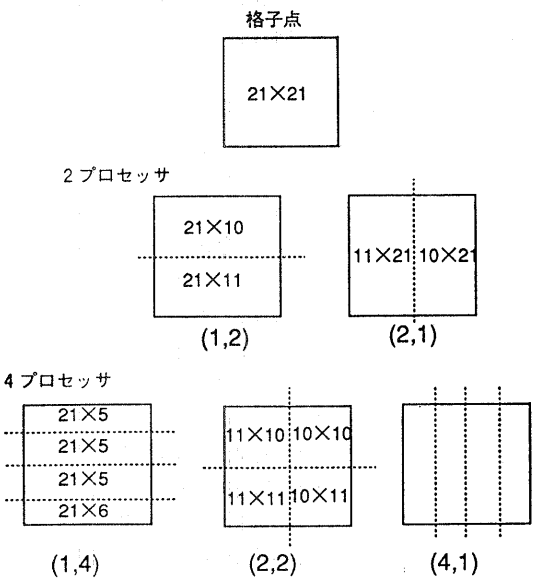


図 5 領域分割の例
Fig. 5 Example of domain decomposition.

4.3 プロセッサ間通信

4.3.1 プロセッサ間データ参照の解析

解析対象領域を分割してプロセッサを割り付け，並列にシミュレーションを実行する方式において，プロセッサ間で通信が必要になるのは，他のプロセッサの担当領域上の変数値を参照する場合である．プロセッサ間で通信を行う必要のあるデータは，DEQSOL プログラムの実行文に現れる微分演算子を差分法で離

散化するときの評価規則によって決定することができる．**図 6** に，DEQSOL の代入文に対するプロセッサ間データ参照解析の手順を示す．まず微分演算子の評価規則に従って，ある格子点上でその微分演算子を評価するときに参照する変数の定義点を示す参照パターンを作成する (Step 1)．次にこれを，参照する変数ごとに重ね合わせることによって，文としての参照パターンを作成する (Step 2)．これと領域分割の図を合わせて，各プロセッサの参照するデータの定義点を得る (Step 3)．

4.3.2 通信コードの生成

DEQSOL プログラムの実行において，プロセッサ間にまたがるデータ参照は，上で述べたように，実行文ごとにその参照データが静的に決定される．分散メモリ型並列計算機の場合，プロセッサ間のデータ転送は多くの場合システムコールによって実現されており，通信データ量に関係なく 1 回の通信に要する時間が大きいのが一般的である[7),15)]．このため，プロセッサ間のデータ転送の回数を削減することが実行性能向上のために不可欠である．本システムでは，計算アルゴリズム全域にわたるデータ依存解析を行うことによって本質的に必要な通信のみを抽出し，さらに通信の要否が実行時にしか決まらないものに対しては動的に通信の実行を制御する方式を採用した．これにより，ユーザが記述したどのような計算アルゴリズムに対しても，無駄がなく，かつデータ待ちによるプロセッサのアイドル時間の少ない効率的な通信を可能にした．以下に通信を決定するルールを示す．ここで，基本ブロックとは DEQSOL の解法アルゴリズム記述の中の，条件文や繰り返し制御等の制御文を含まないひとまとまりの部分を指す．

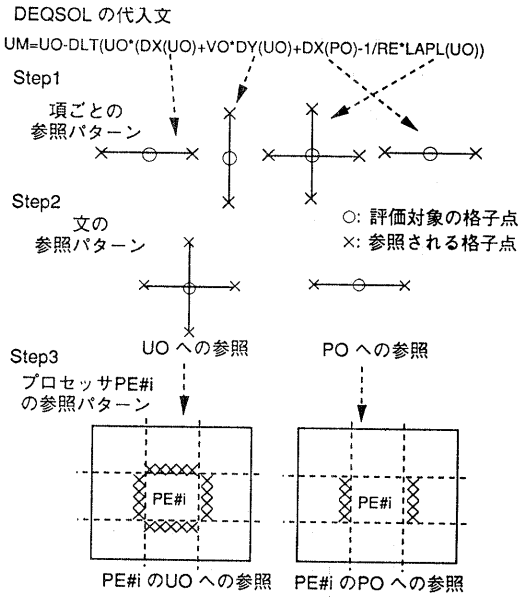


図 6 プロセッサ間データ参照の解析
Fig. 6 Inter-processor data reference analysis.

ルール 1) 一つの基本ブロック内に同一変数の定義と参照がある場合

定義した直後に送信 (send) し，参照する直前に受信 (receive) する．

ルール 2) 複数の基本ブロックにまたがって，同一変数の定義と参照がある場合

基本ブロックの入り口で，ブロック内で参照する全データの送信 (send) を行い，受信 (receive) はそれぞれの変数を参照する直前に行う．

以上の方式をとったとき，条件分岐の結果によって実行されない可能性のある基本ブロックで定義する変数を，実行フローの下流にある別の基本ブロックで参照する場合，通信の要否が実行時に決まるため，実際には必要のない通信が生ずる場合がある．これを回避するために，さらに次の方法を取り入れた．

ルール 3) 通信の要否が実行時に決まるものについては，変数の更新の有無を実行時に判断して通信の実行を制御するコードを生成する．

4.3.3 DEQSOL による記述の利点

ここでは，上記の DEQSOL プログラムの並列化方法を，FORTRAN 等の汎用言語の並列化と比較して考察する．

(1) FORTRAN レベルのプログラムからの並列化における自動データ分割で，単純なループ部分だけを対象としたものでなく，ある程度の規模のプログラムへの適用事例があるものとして，M. Gupta らの研究[18)]がある．これは，多次元配列の各次元の分割の種類（ブロック分割，サイクリック分割等）および分割数を，通信負荷も評価に入れて全体として実行時間を最小化するように決定する方法を提案している．しかし，分割方式をブロック等分割，サイクリック分割といった，データサイズが均一になるようなものの中から選択しているため，領域形状が複雑で配列要素に対する演算量が不均一な場合に適切な分割にならない場合がある．これに対し本方法では，DEQSOL プログラムの領域形状や格子点の記述を利用して演算負荷のばらつきも評価し，データ量が不均一になるような分割も候補に含めて実行時間を最小化する分割を選択し

ている．

（2） FORTRAN 等の並列化においても，通信対象データがコンパイル時の静的解析で決定できない場合に，実行時に通信データを決定する方法についての研究[17]が報告されている．しかし，従来の FORTRAN の並列化の研究では，通信コードの生成および最適化処理の対象はほぼ単一のループ（多重ループも含めて）に限られている．一方，DEQSOL プログラムは領域全体での一つの物理変数の更新処理を1ステップで表現している．これは，FORTRAN で記述した場合に，計算対象領域の各境界面，内部等のそれぞれの部分領域の計算を行う数個から数十個の DO ループに相当する．このため上記ルール3を DEQSOL プログラムに適用することで，FORTRAN の並列化において多くの DO ループにまたがる通信最適化を行うことに相当する処理を実現できる．

5. 性能評価

試作したシステムをいくつかの例題に適用して，nCUBE 2 上で性能評価を行った．例題としては，流体解析（例題 1, 2），および熱拡散解析（例題 3, 4）を取り上げた．各例題での並列化の効果を**図 7** から**図 10** に示す．

計算アルゴリズムとしては，流体解析では SMAC 法[16]を，熱拡散解析では陽解法を用いている．16プロセッサを用いた場合に，1,681 格子点の流体解析では約 8.7 倍（並列化効率 54%*），131,000 格子点の熱拡散解析で，約 14.2 倍（並列化効率 90%）を達成している．

例題3について，領域形状および格子点数を**図 11** に，領域分割手法による実行時間の違いの評価を**表 1** に示す．ここでは，領域分割手法として，本システムで用いた方法と，本システムの方法において各格子点での演算量が均一と仮定した方法，すなわち領域形状の窪みの部分を考慮しない方法，および高さ方向の単純な格子点数均等分割を行う方法，の3通りを比較した．上記2番目の方法が，M. Gupta ら[18]の方法を適用した場合に相当する．表中の「分割数」は，各座標軸方向ごとの分割数を（x 軸方向分割数，y 軸方向分割数，z 軸方向分割数）の形式で示す．本システムで用いた方法は，高さ方向の単純な格子点数均等分割に対して最大28%，領域形状の窪みの部分を考慮しない方法に対して最大 12% の高速化を実現している．**表**

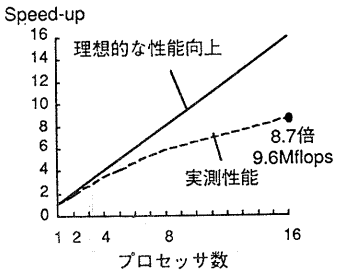


図 7 流体解析（2次元，41×41 格子点）
Fig. 7 Fluid flow analysis (2-dim, 41×41 grid points).

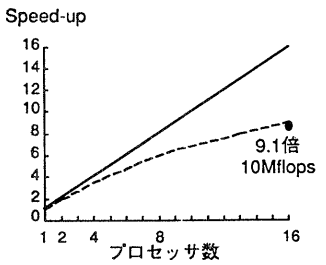


図 8 流体解析（2次元，111×71 格子点）
Fig. 8 Fluid flow analysis (2-dim, 111×71 grid points).

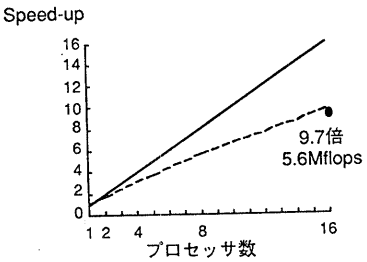


図 9 熱拡散解析（3次元，15×13×21 格子点）
Fig. 9 Head diffusion analysis (3-dim, 15×13×21 grid points).

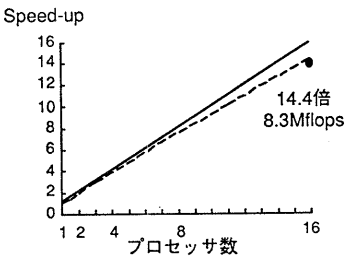


図 10 熱拡散解析（3次元，64×32×64 格子点）
Fig. 10 Head diffusion analysis (3-dim, 64×32×64 grid points).

* 並列化効率＝（1プロセッサで実行したときの所要時間）/（並列に実行したときの所要時間×プロセッサ数）．

2に，例としてプロセッサ数8の場合の本システムの方法による領域分割を示す．

また，通信の要否を実行時に判定することによる性能向上効果を，**図 12** に示す例題5を用いて評価を行った．例題5は，発熱量を変化させて装置内の温度を一定に保つ制御のシミュレーションである．発熱量 F を装置内の温度が1.2℃を越えた場合には減少させ（ステップ3），1.0℃ を下回った場合に熱量を初期値に戻す（ステップ4）．格子点規模は 41×41，時間反復は3,000回である．8プロセッサで実行した場合，実行時判定を行わない場合の実行時間は 23.2 sec，本システムの実行時判定方式を用いた場合には 20.7 sec となり，約 11% の性能向上効果が得られている．この例題では発熱量はステップ3または4のどちらかの判定条件が成立した場合にのみ変更され，発熱量 F が変更された場合はステップ5の直前で通信が必要になる．通信要否の実行時判定を行わない場合には，この通信を発熱量 F の変更の有無にかかわらず毎回行うことになる．実際の計算では発熱量 F の変更は時間反復3,000回のうち260回生じている．

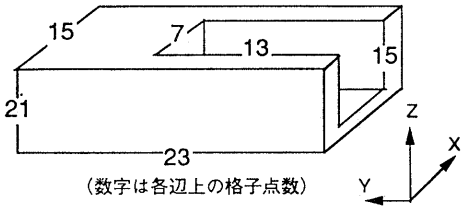


図 11 例題3の領域形状と格子点数
Fig. 11 Figure and grid points of the example 3.

表 1 領域分割手法と並列実行時間（単位：秒）
Table 1 Domain decomposition method and execution time (sec).

プロセッサ数	領域分割手法					
	一方向格子点均等分割		領域形状の窪みを考慮しない方法		本システムの方法	
	分割数	実行時間	分割数	実行時間	分割数	実行時間
4	(1,1, 4)	751	(1, 2, 2)	712	(1, 2, 2)	657
8	(1,1, 8)	442	(1, 2, 4)	410	(1, 2, 4)	362
16	(1,1,16)	330	(1, 2, 8)	265	(1, 2, 8)	242

```
ITER NT UNTIL [NT EQ 3000];                              (1)
   CALL NORMM(UMAX,U);                                   (2)
   IF UMAX GT 1.2  THEN F=F*0.9 AT S ;ENDIF;             (3)
   IF UMAX LT 1.0  THEN F=1.0   AT S ;ENDIF;             (4)
  U=UOLD+DLT*(-DIV(-LAM*GRAD(UOLD))+F);                  (5)
  DLTU=(U-UOLD)/UOLD;                                    (6)
  CALL NORM2(EPS1,DLTU);                                 (7)
  UOLD=U;                                                (8)
END ITER;                                                (9)
```

図 12 例題5のアルゴリズム記述
Fig. 12 The algorithm description of example 5.

次にプロセッサ数選択の効果が現れる例を，前述の例題2（2次元流体解析，111×71 格子点）で示す．利用可能なプロセッサ数を 64 として本システムの領域分割方式を適用した結果は，プロセッサ数36，分割数 (1, 36) が選択され，実行時間は 52.2 秒である．これに対し，プロセッサ数選択を行わない場合は，分割数 (2, 32) が選択され，実行時間は 56.2 秒となり，約 8% の性能向上効果が現れている．

表 2 プロセッサ数8のときの領域分割
Table 2 Domain decomposition for 8 processors.

プロセッサ番号	担当領域					
	x 軸方向		y 軸方向		z 軸方向	
	下限	上限	下限	上限	下限	上限
0	0	14	0	10	0	5
1	0	14	11	22	0	5
2	0	14	0	12	6	10
3	0	14	13	22	6	10
4	0	14	0	12	11	15
5	0	14	13	22	11	15
6	0	14	0	12	16	20
7	0	14	13	22	16	20

6. おわりに

並列計算機向き DEQSOL トランスレータの自動並列化の基本方式について述べてきた．DEQSOL プログラムには，シミュレーションの対象が直接的に表され，計算アルゴリズムが抽象度の高い表現で記述されている．このことが，演算負荷の解析やデータ依存解析を容易にし，高性能な自動並列化を可能にしている．今後の技術的な課題として，次の点が挙げられる．

（1）差分法や有限要素法による数値シミュレーションの場合でも，演算負荷の空間的分布が実行時にしか決まらないようなものもある．こういった問題に対しては，本システムのような並列実行時間の静的推定に基づいて領域分割を決定する方法では対応できない．問題の性質から実行負荷を予測する手法や動的に

負荷分散を調整する手法を開発する必要がある．

（2） 並列計算機の要素プロセッサとして，RISC プロセッサを用いたものが主流になっている．RISC プロセッサの場合，キャッシュメモリのヒット率が性能に大きく影響するため，性能の予測ははるかに困難になる．メモリアクセスの解析等を取り入れた負荷推定の技術が必要になる．

参 考 文 献

1) 小柳：なぜユーザはベクトル計算機から離れられないのか，JSPP '92, pp. 31-32 (1992).
2) 梅谷，ほか：数値シミュレーション用プログラミング言語 DEQSOL，情報処理学会論文誌，Vol. 26, No. 1, pp. 168-180 (1985).
3) 佐川，ほか：数値シミュレーション言語，DEQSOL，情報処理学会論文誌，Vol. 30, No. 1, pp. 36-45 (1989).
4) Umetani, Y. et al.: DEQSOL: A Numerical Simulation Language for Vector/Parallel Processors, *Proc. IFIP*, TC 2/WG 22.5, pp. 147-164, North-Holland (1985).
5) 鈴木，ほか：DISTRAN システムの並列計算機上への実装，JSPP '91, pp. 301-308 (1991).
6) Houstis, E. N. and Rice, J. R.: Parallel ELLPACK: A Development and Problem Solving Environment for High Performance Computing Machines, *Programming Environments for High-Level Scientific Problem Solving*, Gaffney, P.W. and Houstis, E.N. (Eds.), Elsevier Science Publishers B. V., North-Holland (1992).
7) Strauss, H.: Performance of Message Passing on the nCUBE 2, *Parallel Computing '91*, Poster Session (1991).
8) Hiranandani, S., Kennedy, K. and Teng, C.: Compiler Optimizations for FORTRAN D on MIMD Distributed-Memory Machines, *Proc. Supercomputing '91*, pp. 86-100 (1991).
9) Zima, H. P., Bast, H. and Gerndt, M.: SUPERB: A Tool for Semi-automatic MIMD/SIMD Parallelization, *Parallel Computing*, Vol. 6, pp. 1-18 (1986).
10) Karp, A. H. et al.: A Comparison of 12 Parallel FORTRAN Dialects, *IEEE Software*, Vol. 5, No. 5, pp. 52-67 (1988).
11) Galea, E. R. et al.: A Parallel Implementation of a General Purpose Fluid Flow Code and its Application to Fire Field Modeling, *Parallel Computing '91*, pp. 601-608 (1991).
12) Solchenbach, K. et al.: Grid Applications on Distributed Memory Architectures: Implementation and Evaluation, *Parallel Comput.*, 7, pp. 341-356 (1988).
13) Solchenbach, K.: Application Software for Suprenum, *Supercomputer 30*, Vol. 6, No. 2, pp. 44-50 (1989).
14) 星野 力編著：PAX コンピュータ，オーム社 (1985).
15) Hockney, R. W. et al.: Comparison of Communications on the Intel iPSC/860 and Touchstone Delta, *Parallel Computing*, Vol. 18, pp. 1067-1072 (1992).
16) Patankar, S. V.: *Numerical Heat Transfer and Fluid Flow*, Hemisphere Publishing Corporation (1980).
17) Li, J. and Chen, M.: Compiling Communication—Efficient Programs for Massively Parallel Machines, *IEEE Trans. on Parallel and Distributed Systems*, Vol. 2, No. 3, pp. 361-376 (1991).
18) Gupta, M. and Banerjee, P.: Demonstration of Automatic Data Partitioning Techniques for Parallelizing Compilers on Multicomputers, *IEEE Trans. on Parallel and Distributed Systems*, Vol. 3, No. 2, pp. 179-193 (1992).
19) Saltz, J. H. et al.: Run-Time Parallelization and Scheduling of Loops, *IEEE Trans. on Computers*, Vol. 40, No. 5, pp. 603-611 (1992).
20) Koelbel, C. and Mehrotra, P.: Compiling Global Name-Space Parallel Loops for Distributed Execution, *IEEE Trans. on Parallel and Distributed Systems*, Vol. 2, No. 4, pp. 440-451 (1992).

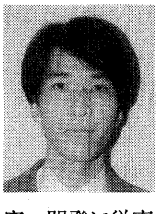
（平成 5 年 9 月 16 日受付）
（平成 6 年 2 月 17 日採録）

大河内俊夫

1960 年生．1985 年東京大学大学院理学系研究科修了．同年(株)日立製作所入社．並列処理ソフトウェアの研究開発に従事．

金野 千里（正会員）

昭和 27 年生．昭和 50 年東京工業大学理学部情報科学科卒業．昭和 52 年同大学院修士課程修了．同年(株)日立製作所入社．以来，中央研究所にて，図形処理システム，数値計算技術の研究・開発に従事．応用数理学会会員．



猪貝 光祥（正会員）

1963 年生．1987 年横浜市立大学文理学部物理学課程修了．同年(株)日立超 LSI エンジニアリング入社．以来，科学技術計算用ソフトウェアおよびその並列化手法に関する研究・開発に従事．