44

NWT向け並列Fortranプログラミングについて

吉田正廣* 中村孝* 福田正大*

岡田信** 中村修一**

Parallel Fortran Programming for NWT

by

Masahiro Yoshida, Takashi Nakamura and Masahiro Fukuda
National Aerospace Laboratory
Shin Okada and Shuichi Nakamura
Fujitsu Limited

ABSTRACT

The Numerical Wind Tunnel(NWT) is under research and development by NAL-Fujitsu joint activity. It is a parallel computer system of distributed memory architecture composed of vector processors.

In this paper, we show the programming method of Parallel Fortran by gradually catching the descriptions for high-speed execution, in the case of parallelizing and tuning usual Fortran programs for NWT.

We present the examination results using a concrete procedure.

1．はじめに

数値風洞（ＮＷＴ）は，ベクトル計算機を要素とする分散メモリ型並列計算機システムである。ＮＷＴ上のソフトウェアについて，これまで，プログラムを高速実行するための基本機能と，プログラム言語の仕様及び実行方式等を検討してきた[1,2]。

本報告では，通常のFortranプログラムを基にしてＮＷＴ向けに並列化・高速化を行う場合を考え，段階を追って高速実行のための記述を採り入れていくことのできる，並列Fortranプログラムの記述方法について，検討結果を具体的手順を示しながら報告する。

2．システム概要

プログラム言語から見た計算機システムは，図１のイメージである。

ここで，プロセサNo.1，No.2，No.3 ...は，互いに対称形である。また，メモリの読み書きにおけるデータアクセス速度は，各プロセサに対してデータがどの空間にあるかによって差があり，速い順に次のとおり（図１の太い矢印）である。

・ローカル空間（自プロセサ）

・自プロセサに配置されているグローバル空間

・他プロセサに配置されているグローバル空間

なお，他プロセサのローカル空間へは，直接に読み書きすることはできない。

3．高速化手順

通常の逐次実行形式のFortranプログラムを基に，並列性，高速性を引き出して，高速実行を可

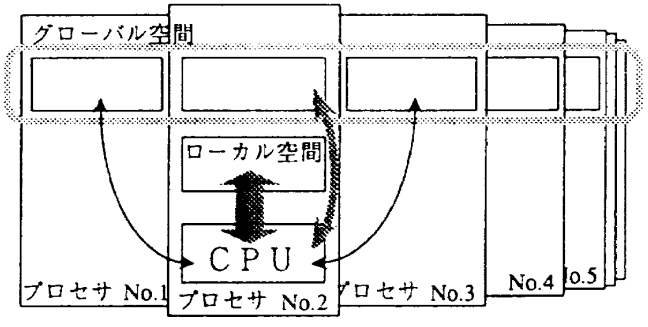

図１　ＮＷＴ計算機システムのメモリ構成

*航空宇宙技術研究所

**富士通株式会社

能とするための並列Fortranプログラミングの手順は，次の各段階に分解できる。

(1) 手続きの分割
(2) データの分割配置
(3) ローカル空間へのコピー
(4) グローバル空間の直接アクセス

段階が進むにつれ，並列化のプログラミングの手間が増え，また，必ずしもすべてのプログラムに適用可能とは限らなくなるが，高速実行性は確実に向上する。

3.1 例題プログラム

図2のプログラムを例として，各段階ごとの並列化手順を示す。

このプログラムでは，配列Aの値を，配列Bを作業用として用いながら，繰り返し更新するものである。ここで，関数の形で書いてあるfとg，特にgについてはかなりの演算量がある計算であり，各プロセサのベクトル命令を有効に利用するものと想定している。

3.2 準備段階：複数プロセサの起動

並列化の準備段階として，使用するプロセサの台数の指定と，複数プロセサ上で動作させるプログラム部分の指定とを行う必要がある。前者の指示を行う記述をPROCESSOR宣言文，後者をPARALLEL REGION構造とする。

```
      PROGRAM EXAMPLE
      REAL A(1000),B(1000)

      READ(1) A

      DO 100 ITIME=1,100

        DO 10 I=1,1000
          B(I)=f(A(I))
 10     CONTINUE

        DO 20 I=2,999
          A(I)=g(B(I-1),B(I),B(I+1))
 20     CONTINUE

100   CONTINUE

      WRITE(2) A
      END
```

図2 例題プログラム

図3のプログラムは，図2のプログラムに，PROCESSOR宣言文とPARALLEL REGION構造を加えた例である。

PROCESSOR宣言文は，プログラムを実行させるのに使用するプロセサの台数を指定する。この台数は，プログラムの並列性，演算時間とのバランス，そしてNWT運用におけるジョブクラス等を勘案して決める必要がある。なお，現在は，後述のデータ分割時の命令効率等から，ここで指定する台数は定数として考えている。

PARALLEL REGION構造は，プログラムのある部分をPARALLEL REGION文とEND PARALLEL文で囲むもので，複数台のプロセサ上で実行させることを指示する。図3の例では，10台のプロセサ上で実行される。PARALLEL REGION構造の外側は，従来どおりのプロセサ1台による実行である。

注意すべき点は，PARALLEL REGION構造の内側の文は，複数台のプロセサで実行されるとはいっても，同じことを何台もで実行しているに過ぎない点である。PARALLEL REGION構造だけでは実行時間短縮効果は全く無く，それがこの段階を“準備段階”と称している理由である。

3.3 第1段階：DOループの並列実行

多くのFortranプログラムにおいては，DOル

```
      PROGRAM EXAMPLE
      REAL A(1000),B(1000)
      PROCESSOR P(10)

      READ(1) A

      PARALLEL REGION
      DO 100 ITIME=1,100

        DO 10 I=1,1000
          B(I)=f(A(I))
 10     CONTINUE

        DO 20 I=2,999
          A(I)=g(B(I-1),B(I),B(I+1))
 20     CONTINUE

100   CONTINUE
      END PARALLEL

      WRITE(2) A
      END
```

図3 準備段階のプログラム

ープによる繰り返しを各回転ごとに並列に実行することが可能であり，高速化効果がある[1]。そこで，PARALELL REGION構造の内側において，ＤＯループの各回転を複数プロセサで分担して実行させる指示を行う。この指示を行う記述をSPREAD DO構造とする。

各回転の分担実行において，各回転で計算された値をＤＯループ終了後に参照するためには，そのデータをグローバル空間に置く必要がある。この指示を行う記述は，GLOBAL宣言文とする。

図４のプログラムは，図３のプログラムの内側二つのＤＯループをSPREAD DO構造とし，必要なGLOBAL宣言文を加えた例である。

SPREAD DO構造は，DO構造をSPREAD DO文とEND SPREAD文で囲むもので，そのＤＯ変数の上下限間を分割し，複数のプロセサでＤＯ変数の別々の値について実行を行うことを指示する。図４の例では，DO 10 のＤＯループについて，図５のように，インデクスの値 100個分ごとが一つのプロセサで実行され，10台のプロセサによって，合わせて1000回分の計算が行われることになる。

ここでの注意点として，SPREAD DO構造の対象ＤＯループが，並列実行可能かどうかがある。各

```
      PROGRAM EXAMPLE
      REAL A(1000),B(1000)
      PROCESSOR P(10)
      GLOBAL A,B

      READ(1) A
      PARALLEL REGION
      DO 100 ITIME=1,100

        SPREAD DO
        DO 10 I=1,1000
          B(I)=f(A(I))
   10   CONTINUE
        END SPREAD

        SPREAD DO
        DO 20 I=2,999
          A(I)=g(B(I-1),B(I),B(I+1))
   20   CONTINUE
        END SPREAD

  100 CONTINUE
      END PARALLEL
      WRITE(2) A
      END
```

図４　第１段階のプログラム

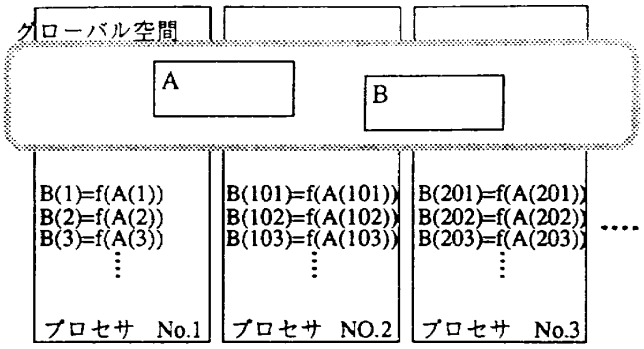

図５　ＤＯループの並列実行

回転で同じ変数への代入を行っていたり，図６のプログラム例のようなデータ依存関係がある場合は，並列実行できず，SPREAD DO構造にしてはならない。プログラムの高速実行のためのオーバヘッド軽減には，できるだけ大きな範囲での並列実行可能なＤＯループを選択することが，重要である。

GLOBAL宣言文は，変数又は配列をグローバル空間に割り付けることを指示する。図４の例では，ＤＯループの中で値が代入される配列ＡとＢが，グローバル空間に割り付けられる。

この段階の並列化は，共有メモリ型の並列処理と同等である。

３．４　第２段階：データの分割配置

ＮＷＴにおけるデータアクセス速度は，“２．システム概要”にあるとおり，他プロセサに配置されているグローバル空間よりも，自プロセサに配置されているグローバル空間にデータがある方が速い。そこで，GLOBAL宣言文による配列の割り付けを，積極的に自プロセサに来るように指定する。このための配列及びＤＯループの分割方法を指示する記述は，“/記法”とする。

“/記法”で指定する分割方法は，分割対象の上下限と，割り当て先のプロセサとの組みで表せる。その指定をする記述は，INDEX PARTITION宣言文

```
      DO 10 I=1,999
        B(I)=B(I+1)-B(I)
10 CONTINUE
```

図６　並列化できないＤＯループ

とする。

図7のプログラムは，図4のプログラムにINDEX PARTITION宣言文を加え，GLOBAL宣言文とSPREAD DO文に“/記法”を追加した例である。

ここでは，ＤＯループの繰り返しの分割方法と配列の分割方法とを，一致させることが重要である。プログラム中の各所から参照される配列は，すべてのＤＯループの分割とは一致させられない場合もあるが，参照頻度等を考え，一致性を高くとることが重要である。図7においては，二つの配列と二つのＤＯループに，“/Q”という同一の分割方法を指定している（図8参照）。

ところで，図7のDO 20のループのように，境界条件により上下限を狭めて使うことがしばしばあるが，そのまま分割を行うと，図9ａにあるように分割箇所がそろわない場合がありうる。これでは効果が半減するので，INDEX PARTITION宣言文でインデクス範囲を1～1000と指定し，それに基づいて2～999の上下限を分割することにより，図9ｂのように，そろった分割とすることができる。

なお，このような指定を行っても，DO 20ループ内のB(I-1)やB(I+1)におけるように，特定の

```
      PROGRAM EXAMPLE
      REAL A(1000),B(1000)
      PROCESSOR P(10)
      INDEX PARTITION Q=(P,1:1000)
      GLOBAL A(/Q),B(/Q)

      READ(1) A
      PARALLEL REGION
      DO 100 ITIME=1,100

        SPREAD DO /Q
        DO 10 I=1,1000
          B(I)=f(A(I))
   10   CONTINUE
        END SPREAD

        SPREAD DO /Q
        DO 20 I=2,999
          A(I)=g(B(I-1),B(I),B(I+1))
   20   CONTINUE
        END SPREAD

  100 CONTINUE
      END PARALLEL
      WRITE(2) A
      END
```

図7　第2段階のプログラム

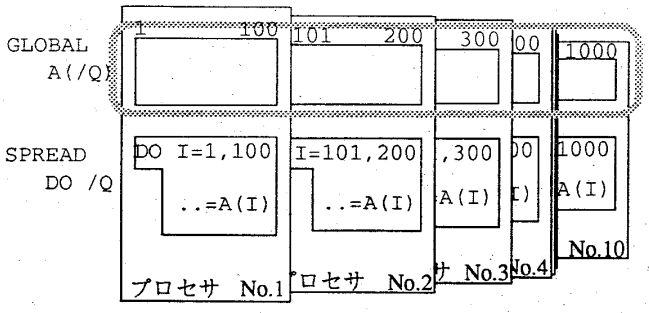

図8　配列とＤＯループの分割配置

ＤＯ変数の値のときに，参照配列要素が自プロセサ内に無く，隣接プロセサへのアクセスとなることがある。ただし，多くの場合において参照データには局所性があり，大部分のアクセスは自プロセサに収まるように記述可能と考えられる。

3.5　第3段階：ローカル空間へのコピー

ＮＷＴにおけるデータアクセス速度は，ローカル空間の方が，自プロセサに配置されているグローバル空間よりも，さらに高速である。そこで，グローバル空間に置かれたデータをいったんローカル空間へコピーし，演算ののち，結果をグローバル空間へ書き戻すことにする。

ここで作業用に使用するローカル空間上の領域は，各々のプロセサごとに個別に確保するのではなく，論理的に連続した一つの配列を分割して，その一部分ずつを各プロセサのローカル空間に割り当てる形とする。これは，3.4項で，グローバル配列を自プロセサ内での参照が多くなるように分割したのと，同様の考え方である。

ただし，グローバル空間ではプロセサが異なっても参照することはできたのに対し，ローカル空

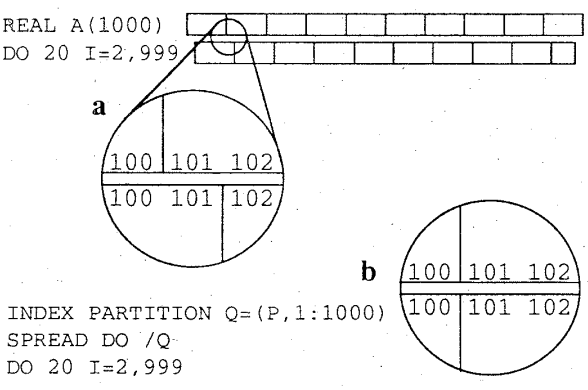

図9　分割の一致

間ではこれができないので，ＤＯループの分割における配列参照と，ローカル配列の分割とは，完全に一致させなければならない。

このための，ローカル配列の分割を指示する記述は“/記法”付きのLOCAL宣言文，グローバル空間との間の転送を指示する記述はSPREAD MOVE構造とする。

図１０のプログラムは，図７のプログラムにローカル配列Ｃを加え，作業用として使われている配列Ｂをグローバルからローカルに変え，そして，転送のためのSPREAD MOVE構造を追加した例である。グローバル配列Ａの値をローカル配列Ｃへコピーし，Ａの代わりにＣを使ってローカル空間上で演算し，そののちにＣの値をＡに書き戻すようにしている。

ここで特徴的なのは，DO 20のループの内側に現れているB(I-1)，B(I+1)に対する扱いである。Ｉの値がＤＯループ分割後の下限値又は上限値のときに，通常の配列の分割方法においてはこれらの配列要素は自プロセサに無く，参照ができない。

こういった問題は，自プロセサ内に，正規の分割結果よりも左右に数要素分多く領域を持ち，その要素については隣接プロセサと重複して，グローバルからのコピー又は計算結果の代入を行うことによって解決できる。この指示を行う記述を，“/記法”のWING項とし，その要素を“袖”と呼ぶ。

図１１は，図１０のプログラムにおけるデータの流れであり，グローバル配列Ａからローカル配列Ｃへ左右１要素分ずつ重複して複写し，ローカル配列Ｂへは左右１要素分を余計に計算して代入し，これを用いてＣの値を計算し，最後にＡに書き戻す様子を示している。

ローカル空間へのコピーにおいては，できるだけ大きな単位で行うことができるようにすることが重要である。

３．６　第４段階：自グローバルの直接アクセス

ＮＷＴにおいては，図１にあるとおり，グローバル空間は，各プロセサのメモリを使用して形成されている。そこで，自プロセサに配置されているグローバル空間については，これをローカル空間とみなしてアクセスすれば，さらに高速化が可能である。このための記述は，グローバル配列とローカル配列とのEQUIVALENCE宣言文とする。

```
      PROGRAM EXAMPLE
      REAL A(1000),B(1000),C(1000)
      PROCESSOR P(10)
      INDEX PARTITION Q=(P,1:1000)
      GLOBAL A(/Q)
      LOCAL B(/Q(WING=1:1)),C(/Q(WING=1:1))

      READ(1) A
      PARALLEL REGION
      DO 100 ITIME=1,100

        SPREAD MOVE /Q(WING=1:1)
        DO 9 I=1,1000
   9      C(I)=A(I)
        END SPREAD
        SPREAD DO /Q(WING=1:1)
        DO 10 I=1,1000
          B(I)=f(C(I))
  10    CONTINUE
        END SPREAD
        SPREAD DO /Q
        DO 20 I=2,999
          C(I)=g(B(I-1),B(I),B(I+1))
  20    CONTINUE
        END SPREAD
        SPREAD MOVE /Q
        DO 21 I=1,1000
  21      A(I)=C(I)
        END SPREAD

 100  CONTINUE
      END PARALLEL
      WRITE(2) A
      END
```

図１０　第３段階のプログラム

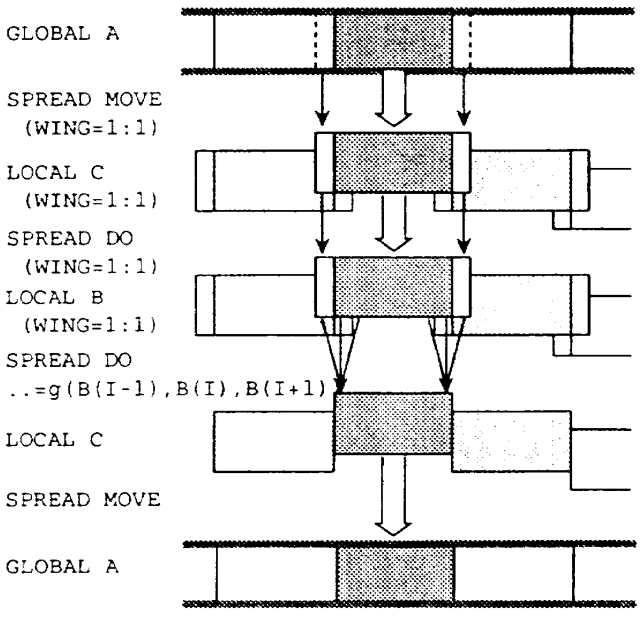

図１１　分割ローカル配列とデータの流れ

EQUIVALENCE文で指定できるのは，同じ分割方法が指定されているグローバル配列とローカル配列である。ただし，“袖”はローカル配列にのみ持ちうるため，EQUIVALENCE結合されるのは“袖”を除いた領域のみである。

EQUIVALENCEされた配列間においては，SPREAD MOVE構造による値のコピーは不要となる。しかしながら，ローカル配列の“袖”については，本体部分の値をコピーしてこなければならない。それを指示する記述を，WINGFIX文とする。

図１２のプログラムは，図１０のプログラムにEQUIVALENCE文を採り入れ，SPREAD MOVE構造の代わりをWINGFIX文で行わせた例である。配列ＡとＣとは，同一の分割方法を用いているため，領域の重ね合わせが可能である。

図１３は，図１２のプログラムにおけるデータの流れであり，WINGFIXによってローカル配列Ｃの袖に本体Ａの値を入れ，Ｂを経て計算結果をＣに代入し，それがすなわちグローバル配列Ａの値となっている様子を示している。DO 100のループが繰り返されると，前回の隣接プロセサによる計

```
      PROGRAM EXAMPLE
      REAL A(1000),B(1000),C(1000)
      PROCESSOR P(10)
      INDEX PARTITION Q=(P,1:1000)
      GLOBAL A(/Q)
      LOCAL B(/Q(WING=1:1)),C(/Q(WING=1:1))
      EQUIVALENCE (A,C)

      READ(1) A
      PARALLEL REGION
      DO 100 ITIME=1,100

        WINGFIX C
        SPREAD DO /Q(WING=1:1)
        DO 10 I=1,1000
          B(I)=f(C(I))
   10   CONTINUE
        END SPREAD
        SPREAD DO /Q
        DO 20 I=2,999
          C(I)=g(B(I-1),B(I),B(I+1))
   20   CONTINUE
        END SPREAD

  100 CONTINUE
      END PARALLEL
      WRITE(2) A
      END
```

図１２　第４段階のプログラム

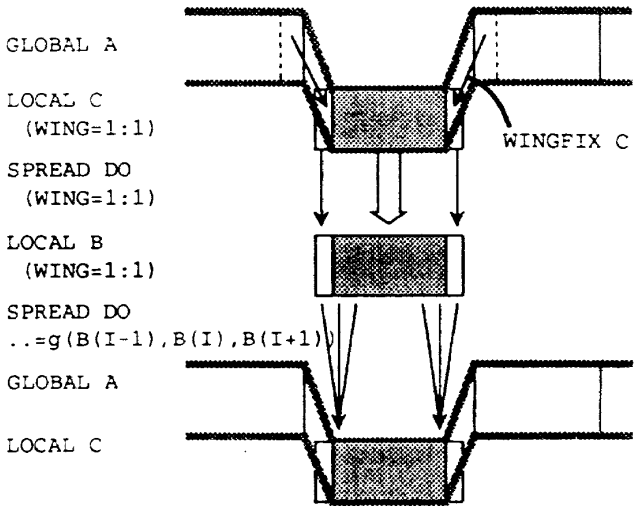

図１３　グローバルとローカルの重ね合わせ

計算結果が，WINGFIXよって再びローカル配列Ｃの袖に格納され，計算が続くことになる。

グローバル配列とローカル配列のEQUIVALENCE結合は，やや技巧的な方法ではあるが，データ転送量の削減には効果的である。そのためには，ＤＯループの分割と配列の分割を一致させることが必要である。

4．まとめ

分散メモリ型並列計算機用のプログラム開発は，今のところ，既存のFortranプログラムをベースとすることから始められると考えられる。この場合に対する，効果のある段階的な開発手順が提示できた。

今後は，各段階における並列効果の定量的な評価，並びに，自動並列化及びプログラム開発支援系の検討も行っていく予定である。

本報告は，平成３年２月に開始された航技研・富士通の共同研究「数値風洞の開発研究」に基づいている。

5．参考文献

[1] 岡田信，高村守幸：“ＣＦＤ向け並列計算機のソフトウェア”，航空宇宙研究所特別資料，ＳＰ－１３，P109，1990

[2] 福田正大，中村孝，吉田正廣，岡田信，中村修一：“数値風洞の言語処理ソフトウェア”，航空宇宙研究所特別資料，ＳＰ－１６，P115，1991