ハイパフォーマンス
コンピューティング 87－15
（２００１．７．２６）

データ分散の図式表現と計算分散公式の提案及び評価

佐藤 真琴
日立製作所システム開発研究所

要旨

従来，ある１次元配列にHPFが規定する一般の規則的データ分散を適用し，この配列の要素で，その添字がループ制御変数の線型式となるものに従って一般の１重ループを分散する場合，非効率なコードしか生成できなかった．本論文では，データ分散を図式で表現し，この表現を用いて上記に対する計算分散公式を与えた．またこの公式における添字参照の周期性を示し，これからInspector-executor法によるテーブル検索法コードを与えた．このコードは日立SR2201上で実行時解決法よりも1.6から6.3倍高速であることがわかった．

Diagram Representation of Data Distribution and Computation-Partitioning Formula

Makoto Satoh
Systems Development Laboratory, Hitachi, Ltd.

Abstract

For any one-dimensional array of a general regular distribution of HPF and any single nested loop including an ON HOME directive with a general affine subscript of the array element in its HOME clause, the conventional computation-partitioning methods can not generate effective code. In this paper, we represent any regular data distribution of HPF by a diagram and provide a computation-partitioning formula for the above loop. Moreover, we prove the periodicity of array references in the formula and provide a table-lookup code using the Inspector-executor technique. This optimized code outperformed the run-time resolution code by 1.6 to 6.3 times on a Hitachi SR2201 supercomputer.

記号

p：プロセッサ数
b：ブロックサイズ
q：プロセッサ番号
L_T ：テンプレートの，ある次元の下限値
e：テンプレートの，ある次元の寸法
c：cyclic(b)分散におけるサイクルの数 $=\lceil e/(p*b)\rceil$
n：テンプレートのAXIS_TYPEがNORMALとなる次元数
v：アライン指示文でアラインソースが"*"となる次元数
r：テンプレートのAXIS_TYPEがREPLICATEDとなる次元数
s：テンプレートのAXIS_TYPEがSINGLEとなる次元数
f_a*i+f_b：アライン添字
i_a*i+i_b：ON HOME指示文のHOME節中の配列添字
l, u, m：分散対象ループの下限値，上限値，ストライド
$F(x) := f_a*i_a*x+f_a*i_b+f_b-L_T$
$h(j,q) := (p*b*j+b*q+L_T-f_a*i_b-f_b)/(f_a*i_a)$
$t(j,q) := (p*b*j+b*q+(b-1)+L_T-f_a*i_b-f_b)/(f_a*i_a)$
$h'(j,q) := (p*b*j+b*q+L_T-f_a*i_b-f_b-l*f_a*i_a)/(f_a*i_a*m)$
$t'(j,q) := (p*b*j+b*q+(b-1)+L_T-f_a*i_b-f_b-l*f_a*i_a)/(f_a*i_a*m)$

1．はじめに

分散メモリ型計算機や分散共有メモリ型計算機では，各プロセッサのデータアクセスを局所化させてプログラムの実行を高速化するデータ分散が重要である．本論文で考察の対象とするプログラムパターンは，HPF [1]による規則的なデータ分散が指示された１次元配列，並びに，ループ制御変数の線型式を添字とする配列要素をHOME節[1]に指定したON HOME指示文を含む１重ループである．このパターンを，本論文ではRDLS(1, 1)パターン（Regular data Distribution and Linear Subscript）と呼ぶ．このパターンは密行列を使った流体計算など多くの科学技術計算プログラムにおいて基本となるものである．

一般のRDLS(1, 1)パターンに適用可能な計算分散法として実行時解決法（RR）がある．しかし，その生成コードの実行性能は良くない[2]．

また，ループストライドが定数となるRDLS(1, 1)パターンに適用可能な計算分散法としてD system [3]がある．これはプロセッサ数やブロックサイズが定数の場合，オメガテストを用いるため，コンパイルに多くの時間がかかる可能性がある．他方，これらの値が定数でない場合はvirtual

processor[7]によるコード（VP）を生成する．VP は block-cyclic 分散ではテーブル検索法コード[5]より遅い[6].

また，テーブル検索法コードは高速であるが，特殊なパターンにしか適用できていない[5][6].

本論文の目的は，コンパイル時に値の決まらない変数を含む，一般の RDLS(1, 1)パターンに対して，短いコンパイル時間で高速なテーブル検索法コードを与えることである．このために，我々は，一般の RDLS(1, 1)パターンを，複雑性を視覚化する目的で図式表現し，これから計算分散公式（TH1)を得た．次に，TH1 における参照添字の周期性を示し，最初の周期における参照添字取得コードを Inspector とし，取得した参照添字から一般の参照添字を得て計算を行なうコードを Executor とするテーブル検索法コードを導いた．最後にこのコードから従来タイプのテーブル検索法コードを導いた．我々のコードはプログラム中の変数を使って明示的に表わせるため，コンパイル時間は短い．尚，本論文では，対象配列は共有メモリ上にあるものとし，その配列添字はデータ分散前の添字と同じ（グローバルアドレス表現）とする．

本論文の残りは以下となる．第２章ではデータ分散の図式表現を示し，第３章ではこれを用いて導いた計算分散公式を与える．第４章ではこの公式の参照添字の周期性を示してテーブル検索法コードを与え，第５章では提案コードの評価を示し，第６章で本研究のまとめを行なう．

2. データ分散の図式表現

本章では HPF [2]の ALIGN や規則的データ分散を指示する DISTRIBUTE を，完全系列を含む図式[4]で表現する．

2. 1 テンプレートのデータ分散の図式表現

本節では，以下の１次元のテンプレート T に対する DISTRIBUTE を図式で表現し，テンプレートを含むある区間をデータ分散に関連した小区間の和として表現する．

```
!HPF$ PROCESSORS  P(p)
!HPF$ TEMPLATE T(L_T : L_T +e–1)                (1)
!HPF$ DISTRIBUTE T(CYCLIC(b)) ONTO P
```

2. 1. 1 標準テンプレートへの埋めこみ

まず，テンプレート T の要素数が式(1)中の p や b で割りきれないことによる扱いにくさを解消するため，T を区間 $I_{pbc}=[0:p*b*c-1]$ (但し，$c=\lceil e/(p*b)\rceil$)へ埋めこむ．

定義1　テンプレート T に対して，各次元の下限値が 0 となるように添字範囲を平行移動したものを T_0 とする．また，I_{pbc} を T に対する標準テンプレートと定義する．　□

この時，式(1)の T に対する T_0 の宣言は

```
!HPF$ TEMPLATE T0 (0 : e–1)
```

となる．以降では，テンプレートや配列と，それらの添字から成る集合とを同一視する．すると，T から T_0 へ写像

$$T \ni x \rightarrow x - L_T \in T_0 \qquad (2)$$

が定義できる．次に，$p*b*c \geq e$ より，T_0 は標準テンプレート I_{pbc} の部分集合とみなせ，T_0 から I_{pbc} の中へ写像

$$T_0 \ni x \rightarrow x \in I_{pbc} \qquad (3)$$

が定義できる．最後に，I_{pbc} を名前とするテンプレートを考え，それに対して T と同じ HPF 指示文を指示する．

```
!HPF$ DISTRIBUTE I_pbc(CYCLIC(b)) ONTO P        (4)
```

この時，指示文(1)による $x \in T$ のマッピングと式(2)と(3)の合成の結果 $x - L_T \in I_{pbc}$ の指示文(4)によるマッピングは等しくなる．よって，以降では，テンプレート T のデータ分散のかわりに，I_{pbc} に対するデータ分散を考える．

2. 1. 2 標準テンプレートの図式表現と区間表現

指示文(4)は，データ分散前の標準テンプレート I_{pbc} 上に[0 : b– 1]を初めに b 個の連続要素から成る区間（各々をブロック区間と呼ぶ）が p 個だけ連続的に並び，これら b*p 個の連続要素列が c 個並ぶことを表わす．そこで，ブロック区間を I_b=[0 : b– 1]，b*p 個の連続要素列を I_{pb}= [0 : b*p–1]，p 個のプロセッサ列を I_p= [0 : p– 1]でモデル化する．以下，我々は，I_b，I_p，I_{pb} を図式[4]で表現する．この時，次の方針を定めた．

[方針１] 完全系列[4]を多く含める．

[方針２] 添字の連続性をモデル化する．

これらは，将来，既存の定理の適用や，通信データの連続性による通信最適化への応用を考えたためである．

Z 加群 Z_b= {0, 1, ..., b - 1}，Z_p，Z_{pb} には，写像"mod (b)"，"b 倍"等で結ばれた完全系列が考えられ，これのアナロジーより，区間 I_b，I_p，I_{pb} 間にも以下の２つの完全系列を考える．

$$0 \rightarrow I_b \underset{\lfloor /p \rfloor}{\overset{*p}{\rightleftarrows}} I_{pb} \overset{\text{mod } p}{\rightarrow} I_p \rightarrow 0 \quad \text{(exact)} \qquad (5)$$

$$0 \rightarrow I_p \underset{\lfloor /b \rfloor}{\overset{*b}{\rightleftarrows}} I_{pb} \overset{\text{mod } b}{\rightarrow} I_b \rightarrow 0 \quad \text{(exact)} \qquad (6)$$

図式(5)では，I_p の１つの元に写される I_{pb} の元が I_{pb} において不連続（例えば，I_p の元 0 に対して，0, p, 2*p, ...）であるので，[方針２]に反する．

一方，図式(6)では，mod b で[0 : b - 1]に写される区間[0 : b - 1]，[b : 2*b - 1]，[2*b : 3*b - 1]，...はブロックサイズ b の連続要素に対応し，I_p の各元が写される I_{pb} の元 0, b, 2*b, ... はこれらのブロック区間の先頭要素なので，ブロック区間の他の要素も同じプロセッサに対応すると考えると，データ分散を自然に表現できる．また，上記のモデル化により，ブロック区間中の任意の元からそれがマッピングされるプロセッサ番号を得る写像は I_{pb} から I_b への $\lfloor /b \rfloor$ で与えられ，これは HPF の定義と合う．以上より，我々は，図式(6)を採用する．図式(6)から以下の区間表現を得る．

$$I_{pb} = \sum_{q=0}^{p-1} (b*q + I_b). \qquad (7)$$

図１は b=2, p=3 での図式(6)による区間間の関係を示す．

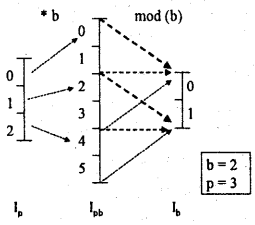

図1: I_p, I_{pb}, I_b間の関係.

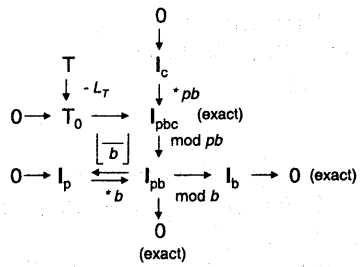

図2: テンプレートのデータ分散の図式表現

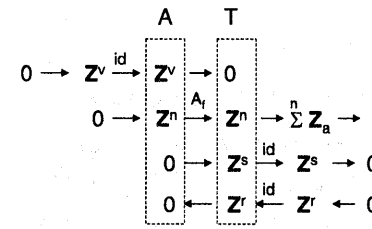

図3:アラインメントの機能毎の図式表現

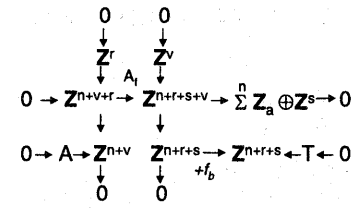

図4:アラインメントの図式表現

同様にして，以下の図式および区間表現を得る.

$$0 \to I_c \xrightarrow{*(p*b)} I_{pbc} \xrightarrow{\bmod (p*b)} I_{pb} \to 0 \quad \text{(exact)} \qquad (8)$$

$$I_{pbc} = \sum_{j=0}^{c-1} \sum_{q=0}^{p-1} (p*b*j + b*q + I_b). \qquad (9)$$

以下，このjの値を分散周期と呼ぶ.

式(2)と(3)及び図式(5)と(8)より図2の図式を得る．尚，以降の節では，"(exact)"は省略する.

2. 2 アラインメントの図式表現

本節では，$\dim_A$次元配列Aの，$\dim_T$次元テンプレートTへのALIGNを図式で表現する.

定義2 テンプレートTのAXIS_TYPEがNORMALな次元にアラインする配列Aの次元のAXIS_TYPEをNORMAL，配列Aの，それ以外の次元のAXIS_TYPEをVANISHEDと定義する. □

この時，AXIS_TYPEがNORMAL，REPLICATED，SINGLEとなるテンプレートTの次元数を，各々，n, r, s, AXIS_TYPEがVANISHEDとなる配列Aの次元数をvとすると，

$$n + v = \dim_A, \quad n + r + s = \dim_T \qquad (10)$$

となる．よって，配列A，テンプレートTの各次元の添字から成る集合をZの部分集合と見なすと，配列A，テンプレートTは，各々，Z^{n+v}, Z^{n+r+s}の部分集合と見なせる.

また，ALIGN指示文における，配列A（またはテンプレートT)の添字ベクトルで，AXIS_TYPEがNORMALまたはSINGLE以外の次元の要素が0となるものをI（またはJ)とすると以下が成立する.

$$J = A_f * I + f_b. \qquad (11)$$

但し，A_fは行列，f_bは定数ベクトルで，Tの AXIS_TYPEがNORMALまたはSINGLEとなる次元の定数項を同じ次元に含み，その他の次元は0である.

図3は配列AおよびテンプレートTの，AXIS_TYPEがVANISHED, NORMAL, SINGLE, REPLICATEDとなる次元を別々に考察して得られた完全系列である．配列AまたはテンプレートTは各々，Z^{n+v}, Z^{n+r+s}の部分集合と見なせるので，図3中の点線で囲まれたZ加群の直和に含まれる．ここで，idは恒等写像を表わす．但し，図3は式(11)の項f_bを含んでないことに注意．これは後（図4）で含める．以下，図3の各図式を説明する.

Z^v：テンプレートTにアラインするZ^vの元がないことと，テンプレートTの任意の要素に対してZ^vの全ての元が対応することを表現する.

Z^r：写像の向きが逆であること以外はZ^vと同じ.

Z^s：0からZ^sへの写像が単射であることから，テンプレートTで1つの定数ベクトルに対応することを表現する.

Z^n：A_fは次元毎には定数倍写像なので，それをa倍と表わすと，Z_aを用いた上記完全系列を得る.

図4は図3の完全系列をつないで得た図式である.

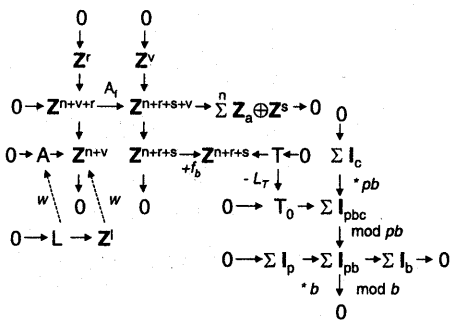

図5：一般の規則的データ分散の図式表現

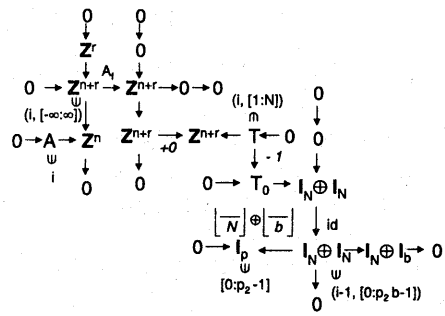

図6：REPLICATE の図式表現例

2. 3 一般の場合の図式表現

図5は多次元配列 A に対する HPF の規則的データ分散の図式表現である．これは図2の図式を次元毎に直和を取り，図4の図式とつないで得られる．ここで，A_fはZ^v やZ^rに対応する次元は恒等写像となるように変更・次元拡張した行列，L はループ繰り返し空間，w は添字関数を表わす．

2. 4 例

本節では REPLICATED [1]を図5で表現する．以下の HPF 指示文を考える．

```
!HPF$ TEMPLATE T(N, N)
!HPF$ ALIGN A(I) WITH T(I, J)                    (12)
!HPF$ DISTRIBUTE T(*,BLOCK(b)) ONTO P(p)
```

図6は，指示文(12)を表現する図式である．まず，左下の配列 A の任意の元 i に対するZ^{n+r}の元は(i, [-∞ : ∞])，T の元は(i, [1 : N])，$I_N \oplus I_N$の元は(i – 1, [0 : p_2*b – 1])となり，I_pの元はI_pの全範囲[0 : p_2 – 1]となる．したがって，配列 A の任意の元はプロセッサ全体に割りつけられる，即ち，レプリケートされる．

3. 計算分散

本章では，RDLS(1, 1)パターンに対する計算分散公式を示す．尚，本章における計算分散は，グローバルなアドレス表現に対するものである．

定理1　以下のプログラム片を考える．

```
!HPF$ PROCESSORS P(p)
!HPF$ TEMPLATE T(L_T : L_T+e-1)
!HPF$ ALIGN A(i) WITH T(f_a*i+f_b)
!HPF$ DISTRIBUTE T(CYCLIC(b)) ONTO P
L1:      do i = l, u, m
!HPF$ ON HOME(A(i_a*i + i_b))
           Φ (i)
         enddo
```

ここで，Φ(i)はループボディ中のコードを表わす．このループ L1 を計算分散した結果は下記コード C1 となる．

```
C1:  if (i_a * f_a = 0)
       if (q = ⌊ {( f_a * i_b + f_b – L_T) mod (p*b)} / b ⌋)
         do i = l, u, m
           Φ (i)
         enddo
       endif
     else
       S1
L2:    do j = Lc, Uc, sign (f_a * i_a *m)
         S2(j, q)
L3:      do i = L(j, q), U(j, q), m
           Φ (i)
         enddo; enddo
     endif
```

ここで，S1 は以下のコードである．

Lc = ⌊ F(l) / (p*b) ⌋; Uc = ⌊ F(u) / (p*b) ⌋

但し，$F(x) = f_a * i_a * x + f_a * i_b + f_b - L_T$.

また，S2(j, q)は以下のコードである．

$h(j, q) := (p*b*j + b*q + L_T - f_a * i_b - f_b) / (f_a * i_a)$

$t(j, q) := (p*b*j + b*q + (b-1) + L_T - f_a * i_b - f_b) / (f_a * i_a)$

$h'(j, q) := (p*b*j + b*q + L_T - f_a * i_b - f_b - l*f_a * i_a) / (f_a * i_a * m)$

$t'(j, q) := (p*b*j + b*q + (b-1) + L_T - f_a * i_b - f_b - l*f_a * i_a) / (f_a * i_a * m)$

if ($f_a * i_a * m > 0$)　$L(j, q) = \max (m * \lceil h'(j, q) \rceil + l, l)$

　　$U(j, q) = \min (t(j, q), u)$

else　$L(j, q) = \min (m * \lceil t'(j, q) \rceil + l, l)$

　　$U(j, q) = \max (h(j, q), u)$

但し，q はプロセッサ番号を表わす．　□

証明概略：図5を用い、式(9)に ALIGN 指示文の添字対応，逆添字関数を適用し，及び標準ループ（下限値 0，ストライド 1）との対応も考慮してループ分散を得、最後にループ L1 との交わりを取る。□

定理1より，テンプレート T の次元の AXIS_TYPE が NORMAL または SINGLE の場合の計算分散が得られた．一方，配列 A の次元の AXIS_TYPE が VANISHED の場合，対応するテンプレートの次元はないので，その次元はデータ分散されない．よって，その次元に制御変数を含むループは元のままである．また，テンプレート T の次元の AXIS_TYPE が REPLICATED の場合，対応する配列 A の次元はないので，この次元はループ分散に影響しない．

以上により，RDLS(1, 1)パターンに対する計算分散は完全に記述された．

4. 最適化計算分散コード

4. 1 参照添字の周期性

定理1における計算分散後の内側ループ L3 の上下限値 L(j,q)，U(j,q)の max または min の第1項 L'(j)，U'(j)を考える．これらは，$f_a * i_a * m > 0$ の場合，以下となる．

$$L'(j) = m*\lceil h'(j, q)\rceil + l, \qquad U'(j) = t(j, q). \qquad (13)$$

ここで，

$$\lambda = \mathrm{LCM}(p*b, f_a * i_a * m), \qquad (14)$$

$\alpha = \lambda / (f_a * i_a * m)$,

$\beta = \lambda / (p*b)$

とすると，以下の周期性が成立する．

$$L'(j+\beta) = L'(j) + m*\alpha, \qquad (15)$$

$$U'(j+\beta) = U'(j) + m*\alpha.$$

$f_a * i_a * m < 0$ の場合もこれと同じ結果を得る．よって，定理1における内側ループは，制御変数 j に関して周期 β を持ち，β毎に i の値は $m*\alpha$ だけ増加する．よって，ループ制御変数 i の値は，[l : l+m* α −1 : m]の範囲の i の値に周期 β 毎に $m*\alpha$ を加えた値として得られる．

4. 2 テーブル検索法コード

4.1節の周期性により以下のテーブル検索法コードを得る．ここで，Inspector ループは実行時解決コードである．

系1 定理1と同じ仮定で計算分散コードは以下となる．

```
ne = 0                                          ! inspector
do i = l, l+m*α−1, m
   if (φ(i)= q) ind (ne) = i;   ne = ne + 1
enddo
NE = ne;   NC = ⌊ (u−l+1) / (m*α) ⌋
do k = 0, NC− 1                                 ! executor
   do jc = 0, NE − 1
      Φ (ind (jc) + k*m*α)
enddo; enddo
do jc = 0, NE − 1                               ! residue loop
   i = ind (jc) + NC*m*α
   if ((u−i)* sign (f_a * i_a *m) ≥ 0) Φ (i)
enddo
```

ここで，$A(i_a * i + i_b)$が存在するプロセッサ番号 φ(i)は

$\phi(i) = \lfloor (F(i) \bmod (p*b)) / b \rfloor$. □

```
C      ia = −2, ib =7,      fa = −5, fb =1
C      l = 10000*(−m)*np, u = −m
C      al = ia*l+ib,        au = ia*u+ib
C      tl = fa*au+fb,       tu = fa*al+fb
       integer A (al : au)
!HPF$ PROCESSORS P (np)
!HPF$ TEMPLATE T (tl : tu)
!HPF$ DISTRIBUTE T(CYCLIC(b)) onto P
!HPF$ ALIGN A (i) WITH T (fa*i + fb)
       do i = l, u, m
!HPF$ ON HOME (A (ia*i + ib))
          A (ia*i+ib) = 100
       enddo
```

図7: 例題プログラム.

尚，

$$\mathrm{Next}(\mathrm{ind}(i)) = \begin{cases} \mathrm{ind}(i+1) & \text{if } 0 \le i < NE-1 \\ \mathrm{ind}(0) & \text{if } i = NE-1 \end{cases}$$

$$\Delta M(\mathrm{ind}(i)) = \begin{cases} \mathrm{ind}(i+1) - \mathrm{ind}(i) & \text{if } 0 \le i < NE-1 \\ \mathrm{ind}(0) - \mathrm{ind}(i) + m*\alpha & \text{if } i = NE-1 \end{cases}$$

とすると，系1の executor ループから以下の従来タイプのテーブル検索法コード[5][6]が導ける．

```
i = ind(0);   locOffset = ind(0)
while (i < NC*m*α)
  Φ (i)
  i += ΔM(locOffset);   locOffset = Next(locOffset)
endwhile
```

5. 評価

提案した種々の計算分散方法の効果を調べるために，図7で示したプログラムを，定理1，系1および実行時解決法コードで書きなおし，ブロックサイズとループストライド6つの組に対して実測した．

測定マシンは分散メモリ型並列計算機である日立 SR2201，プロセッサ数 np は8である．OS は HI-UX/MPP 02-03，Fortran90 コンパイラは OFORT90 V02-06-/E であり，最適化オプションは O(SS)である．

表1及び図8は SR2201 上での実測結果である．IE は系1のテーブル検索法コード，TH1 は定理1による変換結果コード，RR は実行時解決法コードを表わす．表中の値は8プロセッサでの実行時間の平均である．尚，表1中の，第1行目と第2行目のプログラムでは，各プロセッサで参照された配列要素数は各々，10000±1，10000±7 の範囲内の数となったが，それ以外ではちょうど 10000 だったので，各プロセッサの負荷はほぼ均一である．

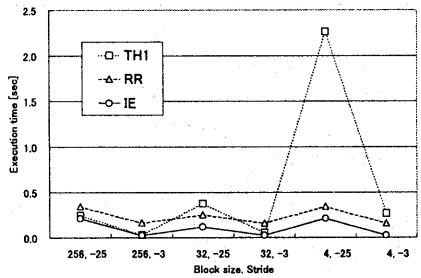

図8：SR2201 上での実測結果 [sec].

表1：SR2201 上での実測結果 [sec].

b, m	IE	TH1	RR	RR /IE	5*m/4*b
256, -25	0.214	0.249	0.346	1.62	0.12
256, -3	0.027	0.030	0.163	5.99	0.01
32, -25	0.116	0.376	0.251	2.16	0.98
32, -3	0.025	0.056	0.162	6.37	0.12
4, -25	0.212	2.257	0.347	1.63	7.8
4, -3	0.025	0.267	0.162	6.32	0.94

b: block size, m: stride

表2：Inspector-executor 法の内訳 [sec].

b, m	ALL	EXE	α	NE
256, -25	0.214	0.117	1024	128
256, -3	0.027	0.025	1024	128
32, -25	0.116	0.117	128	16
32, -3	0.025	0.025	128	16
4, -25	0.212	0.212	16	2
4, -3	0.025	0.025	16	2

b: block size, m: stride

表2はIEの実行時間の内訳である．ALLはIE全体の実行時間を，EXEはExecutorのみの実行時間を表わす．また，αはInspectorループの実行回数を，NEはInspectorが検出した配列要素数を表わす．図7のプログラムでは各々，ブロックサイズの4倍，ブロックサイズの1/2となる．

表1，表2，及び図8より以下のことがわかった．

(1) TH1はブロックサイズが大きい場合はRRより高速だが，RRより遅い場合がある．
定理1のループL2は，ほぼループ下限値に対応する分散周期からほぼループ上限値に対応する分散周期までの全周期をたどるので，計算分散前のループで1ストライド分進む間に複数回の分散周期が現れる大ストライドの場合はRRよりもループ繰返し数が多くなる．図7のプログラムでは分散周期数は逐次ループの繰り返し数のほぼ(5*m)/(4*b)倍である．表1の5行目よりこの値は7.8となり，他の行が1から0.01程度なのに比べて分散周期数が大きい．このため，5行目のTH1は実行時間が長くなったと考えられる．

(2) IEはRRの1.6倍から6.3倍高速である．
IEの方がRRよりループ繰り返し数が少なく（約1/p），1ループ当たりの計算量も少ないためと考えられる．

(3) Inspectorの実行時間は短い．
表2よりInspectorが検出した配列要素数は2から128要素の範囲であり，ほぼ10,000の全参照要素数に比べると少ないためである．その時間はブロックサイズbが256以外の場合は誤差に含まれる．また，ブロックサイズbが大きい場合はRRの代わりにTH1をInspectorに使うと高速になると予想できる．

(4) ストライドが-25より-3の時の方が実行時間は短い．
ストライドが-3の場合，キャッシュラインの再利用が行われたためである．

6. おわりに

HPFが規定する一般の規則的データ分散とループ制御変数の線型式を添字とする配列要素をループ中のON HOME指示文のHOME節に指定した多重ループをRDLSパターンと呼ぶ時，以下を示した．

(1) RDLSパターンに対するデータ分散を図式で表現した．

(2) 1次元配列と1重ループから成るRDLSパターンに対する計算分散公式を与えた．

(3) 計算分散公式における参照添字の周期性を示し，これよりテーブル検索法コードを与えた．

(4) 複雑なサイクリック・ブロック分散を持つプログラムに提案した種々の方法を適用して日立SR2201上で実測し，テーブル検索法コードが実行時解決法よりも1.6から6.3倍高速であることがわかった．

謝辞

日立製作所ソフトウエア事業部の西谷康仁氏には，SR2201の使用に際してご協力いただいた．

参考文献

[1] High Performance Fortran Language Specification Version 2.0, Rice Univ., 1997.

[2] S. Hiranandani et al. Compiling Fortran D for MIMD Distributed-Memory Machines, *Communications of the ACM*, Vol. 35, No. 8, pp. 66-80, Aug. 1992.

[3] V. Adve et al. Using Integer Sets for Data-Parallel Program Analysis and Optimization, PLDI'98, pp. 186-198, 1998.

[4] 河田敬義，ホモロジー代数学，岩波数学選書．

[5] S. Chatterjee et al. Generating local addresses and communication sets for data-parallel programs. PPoPP'93, pp. 149-158, San Diego, CA, May, 1993.

[6] K. Kennedy et al. Efficient Address Generation for Block-Cyclic Distributions, ICS '95, pp. 180-184, 1995.

[7] S. Gupta et al. Compiling Array Expressions for Efficient Execution on Distributed-Memory Machines, *J. of Parallel and Distributed Computing*, Vol. 32, pp. 155-172, 1996.