ハイパフォーマンス
コンピューティング 62－11
（1996．8．28）

ＨＰＦ処理系における再分散解析機能の開発

佐藤　真琴　　根岸　清*　　小林　篤**

（株）日立製作所システム開発研究所
*（株）日立製作所ソフトウエア開発本部
**日立東北ソフトウエア（株）

概要

ＨＰＦ1.0の仕様を満たすデータ再分散指示文に対するデータ再分散解析機能を提案し、我々のＨＰＦ処理系に実装した。これは３つのデータフロー解析による到達する分散定義解析と分散形状の伝播からなる。また、アロケータブル配列への属性変更を基本とする再分散コード生成を実現し、再分散コード改良方式を提案した。ＮＡＳベンチマークのＦＦＴプログラムに対し、本解析の有効性と改良コードによる３０％の性能向上を確認した。

Development of Remapping Analysis in HPF Compiler

Makoto Sato　Kiyoshi Negishi*　Atsushi Kobayashi**

Hitachi,Ltd. Systems Development Laboratory
*Hitachi,Ltd. Software Development Center
**Hitachi Tohoku Software, Ltd.

Abstract

Remapping analysis for remapping directives satisfying HPF 1.0 specifications is proposed and implemented in our HPF compiler. It consists of reaching distribution analysis using three data flow analyses, and distribution shape propagation. The efficient remapping code generation that changes remapped array attributes into allocatable in principle is also implemented. Moreover, improved remapping codes are proposed to remove redundancies. The evaluation result for remapping codes to NAS/FT program indicates that remapping analysis is effective and improved codes decrease the execution time by 30 percent.

1　はじめに

高速計算のニーズは益々高まっており、超並列機に対する期待が大きい。一方、従来、超並列機の普及を妨げていたプログラミングの困難さは、ＨＰＦ(High Performance Fortran)[1]という、Fortranの並列向け拡張仕様により緩和されつつある。ＨＰＦや類似の言語に対するコンパイラは活発に研究され[2][4][10][11][12]、多くの商用ＨＰＦコンパイラもある[5][6][7][8][9]。

ＨＰＦには、プログラムの途中でデータを動的に分散できるデータ再分散指示文がある。これはＡＤＩやＦＦＴのようなデータ依存の方向が実行途中で変化するプログラムで使われることがある。

ＨＰＦ処理系は、データ再分散指示文を含むプログラムを効果的に並列化／最適化するため、プログラム中の任意位置でのデータ分散の方法（以後、データ分散形状と呼ぶ）を知る必要がある。

我々は、我々のHPF処理系で、プログラム中の任意位置でのデータ分散形状を並列化時に決定する、データ再分散解析機能を実現した。他にこれを実現したコンパイラは見当たらない。

関連研究では、実行時情報に基づいてデータ再分散するもの[8]、手続き間コンパイルする時、各手続き呼び出し位置での実引数の静的な分散形状を知るreaching decomposition analysis[3][4]がある。

2　データ再分散指示とＵＡＴ

2.1　データ再分散指示

再分散指示には、変数の分散方法を変える再分割指示文（redistribute）と、ある変数と別の変数との整列方法を変える再整列指示文（realign）がある[1]。

例　再分散指示の例を下記と図１に示す。

(a)　!hpf$ redistribute x(*,block(n)) onto p

ここで、ｘ：分散される変数。distributeeと呼ぶ。
　　ｐ：プロセッサの配置を表わす配列。
　　block(n)：連続ｎ要素単位の分割。

(b)　!hpf$ realign y(i,j) with x(i+1,j+1)

ここで、ｙ：整列する配列。aligneeと呼ぶ。
　　ｘ：整列される配列。align-targetと呼ぶ。

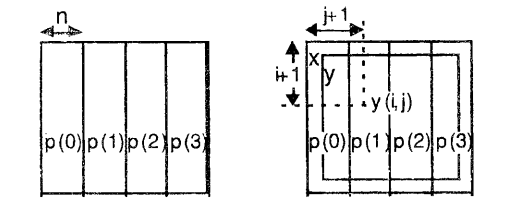


図１　再分割と再整列

2.2　ＵＡＴ

再分散解析で、ＨＰＦ規定に従って再分散指示を正しく解釈するために、次のＵＡＴが重要となる。

定義１　alignee xの最終align-target（UAT）
⇔xのalign-target y,　　if yが他に整列しない
　xのalign-target yのUAT,　if yが他に整列する

例　xがy、yがｚに整列し、zが他に整列しなければ、zはx、yのUATである。

ＨＰＦ規定

Ｋ１：[再]分割指示文のdistributeeは[再]整列指示文のaligneeになれず、[再]整列指示文のaligneeは[再]分割指示文のdistributeeになれない。

Ｋ２：再整列指示文はaligneeのみ分散し、これに整列する他の変数の分散形状は変えない。

Ｋ３：再分割指示文はそのdistributeeをUATとする変数も再分散する。

Ｋ１より、一手続き内の分散変数は、ある指示文のdistributeeかaligneeの一方になる。以後、分散変数をこの分類に従い、指示文とは独立に、単に、distributee、aligneeと呼ぶ。

aligneeとalign-target（またはUAT）をノードとし、両ノードをこの順に向き付けしたエッジで接続すると、有向グラフになる。これを整列（またはUAT）グラフと呼ぶ。

図２は整列グラフとこれをUATグラフに変換した例である。この図で、ｚが他の変数ｗに再整列しても、Ｋ２より、ｘ、ｙのUATはｕのままである。したがって、整列グラフでは解釈を間違うおそれがある。また、ｕを再分割すると、Ｋ３より、ｕをUATとするｘ、ｙ、ｚも再分散される。よって、再分散指示の解釈ではＵＡＴが重要である。

(a)整列グラフ　　(b)UATグラフ

図２　整列グラフとUATグラフ

図３は例題プログラムである。Ｓ１～Ｓ４は手続き入口での分散指示文、Ｓ５～Ｓ７は再分散指示文である。図４は図３のプログラムのフローグラフである。BB1～BB3は基本ブロックを表わす。Ｅ１～Ｅ３、Ｏ１～Ｏ３は各々、BB1～BB3の入口と出口である。図５は、図４のフローグラフ中の主な位置での、x, yとt1, t2とのUATグラフである。

```
S1: !hpf$ distribute t1(block)
S2: !hpf$ distribute t2(cyclic)
S3: !hpf$ align x with t1
S4: !hpf$ align y with x
          :
          if (...) then
S5: !hpf$  realign x with t2
          endif
          :
S6: !hpf$ realign y with x
          :
S7: !hpf$ redistribute t2(block)
```

図３　例題プログラム

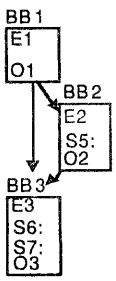
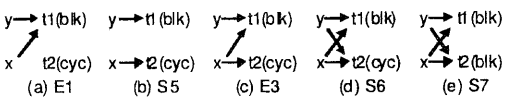


図４　ﾌﾛｰｸﾞﾗﾌ



blk：block分割、cyc：cyclic分割

図５　UATグラフ

3　データ再分散解析

3.1　解析アルゴリズム

ＨＰＦでは、allocate時のdistributeeの分散形状やaligneeのUATは、プログラム入口での分散形状やUATと同じ、と規定している。よって、分散形状は手続き入口、再分散指示文、allocate文で変化する可能性がある。これらを再分散位置と呼ぶ。

/* 到達する分散定義解析 */
（１）基本ブロック入口に到達する分散定義を求める
　　（１ａ）ＵＡＴ解析
　　（１ｂ）到達する整列定義解析
　　（１ｃ）到達する分散定義解析
（２）再分散位置に到達するデータ分散定義を求める

/*分散形状の決定*/
（３）再分散位置へ分散形状を伝播させる
（４）任意位置での変数の分散形状を求める

図６　データ再分散解析アルゴリズム

図６はデータ再分散解析アルゴリズムである。到達する分散定義解析と分散形状の決定から成る。

（１）は、基本ブロック入口に到達するビット集合を、繰返しアルゴリズム[13]で決定する。(1a)は基本ブロック入口でaligneeのUATを決定する。(1b), (1c)は基本ブロック入口に到達する、aligneeのalignee- targetが変化する直前の位置、及び変数の分散形状が変化する直前の位置を決定する。

（２）は、（１）の結果に基本ブロック内の再分散指示文やallocate文による変更を順次加えて、再分散位置でのビット集合を決定する。

（３）は、（２）の結果から、再分散位置で分散形状の変化する変数の分散形状を決定する。

（４）は、（３）の結果から、任意位置での分散形状を決定する。

3.2　到達する分散定義解析

本節では３つのデータフロー方程式系と、それに関連するデータフロー情報集合を説明する。

定義２

（１）VARS：変数を表わす集合。
（２）LOCS：再分散位置を表わす集合。
（３）nl：LOCSの要素数。
（４）nv：VARSの要素数。
（５）B(m, n)：要素が0か1のm行n列の行列。

３方程式のデータフロー情報集合U、A、Dは、各々２次元のビット集合である：

$$U \subset B(nv, nv);\quad A,\ D \subset B(nv, nl)$$

Uは、要素（alignee, aligneeのUAT）のみが１となる集合であり、UATグラフを表現する。

A、Dは、各々、要素（alignee, aligneeの到達する整列定義位置）、要素（変数, 変数の到達する分散定義位置）のみが１となる集合である。

図７は図６の(1a), (1b), (1c)に対するデータフロー方程式である。ここで、＊を任意の記号とすると、

$$U\_* \subset B(nv, nv);\quad A\_*, D\_* \subset B(nv, nl)$$

また、U_IN, U_OUTは、基本ブロック入口、出口でのUを表わす。A、Dについても同様である。その他の記号は以下で定義する。

定義３

（１）AL：全aligneeからなる集合。
（２）DI：全distributeeからなる集合。
（３）L(v)：手続き入口∪allocate位置∪alignee (distributee) vの再整列（再分割）指示位置。
（４）S[n]：基本ブロックｎ中の文からなる集合。
（５）pred[n]：nに分岐する全基本ブロック
（６）last(v,n)：alignee(distributee) vのｎにおける最後の再整列（再分割）位置。
（７）plural(v,d,n)：n外にあり、そこでのvのUATが唯一でないか、dでない、LOCSの部分集合。

(1) $U\_PRE[n] = \{(v, *) \mid v \in DI \text{ or } v \in NA(n)\}$
　　$U\_A[n](x) = \{(v, w) \mid v \in CA(n), a = iuat(v, n) \in AL, (a, w) \in x\}$
　　$U\_D[n] = \{(v, d) \mid v \in CA(n), d = iuat(v, n) \in DI\}$
(2) $U\_OUT[n] = (U\_IN[n] \cap U\_PRE[n]) \cup U\_A[n](U\_IN[n]) \cup U\_D[n]$
　　$U\_IN[n] = \bigcup_{p:pred[n]} U\_OUT[p]$
　　(a) UAT解析

(1) $A\_PRE[n] = \{(v, *) \mid v \in NA(n)\}$
　　$A\_DEF[n] = \{(v, last(v,n)) \mid v \in CA(n)\}$
(2) $A\_OUT[n] = (A\_IN[n] \cap A\_PRE[n]) \cup A\_DEF[n]$
　　$A\_IN[n] = \bigcup_{p:pred[n]} A\_OUT[p]$
　　(b) 到達する整列定義解析

(1) $D\_PRE[n](x) = D\_NoD[n] \cup D\_NoA[n](x) \cup D\_PL[n](x)$
　　$D\_NoD[n] = \{(v, *) \mid v \in ND(n)\}$
　　$D\_NoA[n](x) = \{(v, *) \mid v \in NA(n), (v, d) \cap x = \phi \text{ for } \forall d \in CD(n)\}$
　　$D\_PL[n](x) = \{(v, plural(v, d, n)) \mid v \in NA(n), \exists d \in CD(n), \text{s.t. } (v, d) \in x\}$
　　$D\_DEF[n](x) = \{(v, l) \mid l = last(v, n) \text{ if } L(v) \cap S[n] \neq \phi ; l = last(d, n) \text{ if } L(v) \cap S[n] = \phi, \exists d \in CD(n), \text{s.t. } (v, d) \in x\}$
(2) $D\_OUT[n] = (D\_IN[n] \cap D\_PRE[n](U\_IN[n])) \cup D\_DEF[n](U\_IN[n])$
　　$D\_IN[n] = \bigcup_{p:pred[n]} D\_OUT[p]$
　　(c) 到達する分散定義解析

(1) データフロー情報集合，(2) データフロー方程式，＊：任意の要素

図７　データフロー方程式

(8) NA(n)：＝{ v∈A ｜ L(v)∩S[n]＝φ }
(9) CA(n)：＝{ v∈A ｜ L(v)∩S[n]≠φ }
(10) ND(n)：＝{ v∈D ｜ L(v)∩S[n]＝φ }
(11) CD(n)：＝{ v∈D ｜ L(v)∩S[n]≠φ }

図７を説明するために以下の定義をする。

定義４　nにおけるvの局所UAT：luat(v,n)
⇔n中の再整列指示文のみによって決定される、
nの出口におけるvのUAT。

次は定義より明らかである。

補題１　$UAT_{in}(v,n)$、$UAT_{out}(v,n)$を、基本ブロックnの入口、出口におけるvのUATとすると、

$$UAT_{out}(v,n)=UAT_{in}(luat(v,n),n)$$

図７の説明

（ａ）U_OUT[n]は全ての変数vに対する(v, $UAT_{out}(v,n)$)の和集合である。vがaligneeかつ、そのUATがnで変化する（即ち、v∈CA(n)）なら、補題１より、d=luat(v,n)とすると、$(v,UAT_{out}(v,n))=(v,UAT_{in}(d,n))$となる。dがaligneeなら、$UAT_{in}(d,n)=\{w\in VARS \mid (d,w)\in U\_IN[n]\}$。dがdistributeeなら、$UAT_{in}(d,n)=\{d\}$。両者はU_A, U_Dに対応する。

（ｂ）vがaligneeかつそのUATがnで変わる時のみvのA_OUTは変わる。これはA_DEFに対応する。

（ｃ）vの分散形状がnで不変ならばvに対するD_INの値は不変である（D_NoD、D_NoAに対応）。vがaligneeかつそのUATdがnで不変だが、dの分散形状はnで変わると仮定する。もし(v,loc)∈D_INとなる位置locでvのUATが複数なら、(v,loc)はnで保存される（D_PLに対応）。

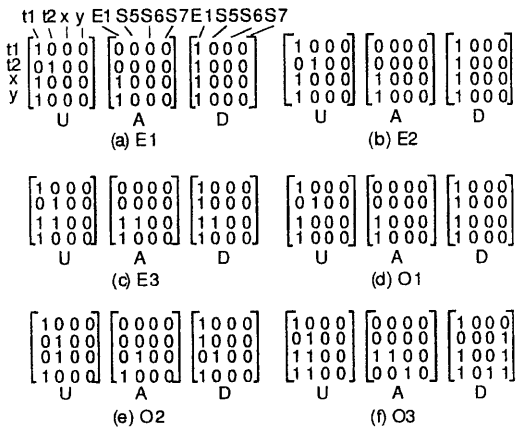


図８　基本ブロック入口･出口での情報集合

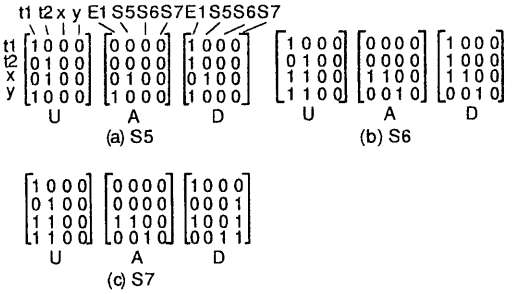


図９　再分散指示位置での情報集合

図８、図９は図３のプログラムに対する、基本ブロック入口･出口、及び再分散指示位置でのデータフロー情報集合である。本プログラムはループがないので、BB1, BB2, BB3の順に解析すれば繰り返し処理はなく、１回で終了する。両図はこの順で解析した結果である。

３.３　分散形状の伝播と決定

３.３.１　分散形状の伝播

データ分散形状は、UATへのalign方法を表すalignment情報と、UATの分割種別を表すdistribution情報から決まる。本方法では、後者を格納するテーブル（DistTbl）と前者を格納するテーブル（AlgnTbl）を作り、AlgnTblはそのUATのDistTblを指すことでデータ分散形状を表現する。両テーブルはトランスレート時にのみ用いる。

DistTblは再分割指示位置のdistribution情報から作る。一方、AlgnTblは再整列指示位置のalignment情報と、この再整列指示位置に到達する分散定義等の位置におけるDistTblやAlgnTblから作る。

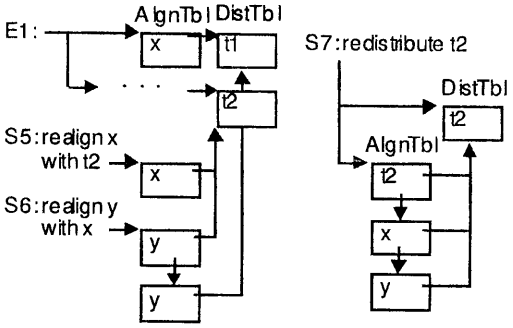
図１０(a)は再整列指示位置Ｓ５、Ｓ６におけるAlgnTblである。以下のように作成する。

Ｓ５　align-target t２に到達する分散定義は図９(a)のDよりＥ１である。t２はdistributeeなので、Ｓ５の整列情報のみからAlgnTblを作成し、これはＥ１におけるt２のDistTblを指すようにする。

Ｓ６　align-target xの到達する分散定義は、図９(b)のDよりＥ１またはＳ５である。Ｅ１（Ｓ５）におけるxのAlgnTblは上で作成済みなので、この整列情報とＳ６の整列情報を合成してAlgnTblを作る。これはＥ１（Ｓ５）でのxのAlgnTblと同じDistTblを指すようにする。

図１０(b)は再整列指示位置Ｓ７におけるDistTblとAlgnTblである。以下のように作成する。

Ｓ７　まず、t２からt２自身へのnatural templateに対するAlgnTblを作成する。次に、t２をUATとするaligneeは、図１１(c)のUより、ｘ、ｙとわかる（t２は除く）。ｘ（またはｙ）の到達する整列定義は図９(c)のAよりＥ１，Ｓ５（またはＳ６）であり、これらの位置でのｘ（またはｙ）のUATは図８(a)、図９(a)のUよりt１、t２(又はt１とt２)なので、Ｓ５（またはＳ６）がｘ（またはｙ）がt２に再整列する位置となる。Ｓ５（またはＳ６）でのｘ（またはｙ）のAlgnTblは図１０(a)で作成済みなので、それとＳ７のDistTblから、Ｓ７でのAlgnTblを作成する。両者はＳ７のDistTblを指す。



（ａ）再整列指示位置　（ｂ）再分割指示位置

図１０　AlgnTblの作成

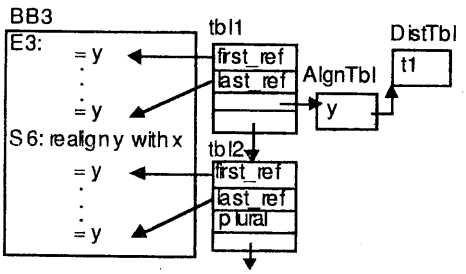


図１１　分散形状テーブル

3.3.2　分散形状の決定

変数毎に、基本ブロック内で分散形状が不変な範囲に一つずつ、分散形状テーブルを作成する。このテーブルはこの範囲の分散形状を表すAlgnTblと、この範囲内の最初と最後の変数の参照点を指す。与えられた参照点が範囲内にあるかを判定することで、参照点での分散形状が得られる。

図１１は基本ブロックBB3におけるｙの分散形状テーブルtbl１、tbl２を示す。Ｅ３とＳ６の間では、ｙの分散形状は図５(ｃ)より唯一に定まるので、tbl１は唯一のAlgnTblを指す。Ｓ６－Ｓ７間では、ｙのAlgnTblは図１０(a)のＳ６が表すように２つなので、tbl２は分散形状が一意でないことを表すpluralという領域を指す。

4　データ再分散コード生成

4.1　変数属性の変更

再分散配列の属性はアロケータブル配列かポインタ配列に変更すべきである。配列要素数がＰＥ数やblock長で割れない場合、再分散前後で、１ＰＥ当たりの要素数が変わることがあるからである。

本処理系では再分散配列の属性は、原則的にアロケータブル配列にした。生成プログラムをコンパイルする際、ポインタ配列では最適化しにくいためである。

```
      real :: a(100)                    allocatable::a(:),t(:)
!hpf$ processors p(4)                   [構造体Aの各成分の値設定]
!hpf$ dynamic a                         allocate(a(A%l:A%u))
        :                                       :
!hpf$ redistribute                      [構造体Tの各成分の値設定]
      a(cyclic) onto p                  allocate(t(T%l:T%u))
                                        call remap(a,A,t,T)
                                        deallocate(a)
                                        A=T
                                  R1: allocate(a(A%l:A%u))
                                  R2: a=t
                                  R3: deallocate(t)
```

入力プログラム　　出力プログラム

図１２　再分散コード

4.2　データ再分散コード

次の３つの場合に再分散コードを生成する。

(1)手続き入口で仮引数に対して分散指示がある

(2)再分散指示位置

(3)実引数の分散形状とインターフェースブロックで記述（descriptive）[1]の形で宣言された分散形状が異なる

図１２は(2)の、再分散指示に対する再分散コードである。ここで、remap(a, A, t, T)は、構造体Aが表わす分散形状を持つ配列aを、構造体Tが表わす分散形状に再分散した結果を、配列ｔへ格納することを表わす。

4.3　データ再分散コード改良方式

図１２の出力プログラムは冗長性をもつ。ａがポインタ配列ならR1, R2, R3は"a=>t"となり、高速

に実行されるはずである。このため、allocatable配列にポインタ代入に似た処理を行う非Fortranルーチンrenameを導入する。

rename (x, y)は、yの先頭アドレスをxの先頭アドレスに等しくし、xを未割付状態、yを割付状態にする。データフロー解析では、この関数に対しR1, R2, R3があるかのごとく解析すればよく、ポインタのようなエリアスの問題は発生しない。

5 評価結果

表１と図１３は、ＳＲ２２０１でＮＡＳベンチマークのＦＦＴプログラムを、配列サイズ64^3で実行した結果である。主要ループを含む同じサブルーチン３つをインライン展開し、最初のループ前後に再分散指示を挿入した。

表１ NAS/FTの実行結果

	1	2	4
ra	1	0.64	0.35
opt-ra	0.71	0.43	0.25
non-r	0.68	>11	

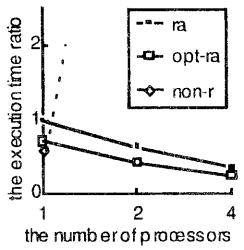


図１３ NAS/FTの実行結果グラフ

raは4.2節のコードを、opt_raは4.3節の改良コードを用いた。後者は前者のトランスレート結果を人手で修正した。non_rは再分散指示を挿入しない場合である。比opt_ra/raは、1,2,4台に対し、0.71、0.68、0.7である。

6 おわりに

- ＨＰＦ1.0仕様の再分散指示文をトランスレート時に解析する再分散解析法を提案し、我々のＨＰＦ処理系に実装した。
- 生成プログラムのコンパイル時の最適化促進のため、分散配列属性を基本的にアロケータブルにした。また、生成コード改良方式を提案した。
- ＮＡＳベンチマークのＦＴに対し再分散解析と改良コードの有効性を確認した。改良コードは改良前コードに対し実行時間を３０％短縮した。

謝辞

本研究にご協力いただいた、日立製作所ソフトウエア開発本部の吉川聡氏、西谷康仁氏、中島恵氏、会田一弘氏、布広永示氏、日立東北ソフトウエアの黒沢隆氏、浦川純一氏、日立製作所システム開発研究所の飯塚孝好氏、佐藤茂久氏、菊池純男氏、十山圭介氏に、感謝します。

参考文献

[1]High Performance Fortran Language Specification, ver.1.0, CRPC, Rice Univ., Jan. 1993

[2]S.Hiranandani et al. Evaluating of Compiler Optimizations for Fortran D, J. of Parallel and Distributed Computing, 21, pp.27-45, 1994.

[3]M.W. Hall et al. Interprocedural Compilation of Fortran D for MIMD Distributed- Memory Machines, Supercomputing '92, pp. 522-534, 1992.

[4]B. Chapman et al. Dynamic Data Distribution In Vienna Fortran, Supercomputing 93, pp. 284-293, 1993.

[5]J. Levesque. Using High Performance Fortran on the Intel Paragon, Intel Users Group Meeting, May 1994.

[6]R. Babb et al. Retargetable High Performance Fortran Compiler Challenges, COMPCON 93,pp.137-146, 1993.

[7]M.Gupta et al. An HPF Compiler for the IBM SP2, CPC95, pp. 22-39, June, 1995.

[8]蒲池 他. HPF処理系の実現とCenju-3での評価, JSPP'95, May, 1995.

[9]進藤 他. FLoPS：分散メモリ型並列計算機を対象とした並列化コンパイラ, JSPP'95, May, 1995.

[10]Bozkus.Z et al. Compiling Fortran 90D/HPF for Distributed Memory MIMD Computers, J. of Parallel and Distributed Computing, 21, pp.15-26, 1994.

[11] E. Su et al. Advanced Compilation Techniques in the PARADIGM Compiler for Distributed -Memory Multicomputers, ICS95, pp. 424-433, July 1995.

[12]T. Brandes. Optimization Startegies within the High Performance Fortran Compilation System ADAPTOR, CPC95, pp. 49-64, June, 1995.

[13]H.Zima et al. Supercompilers for Parallel and Vector Computers, Addison-Wesley,1991.