VP 用チューニングプログラム

岡　田　康　利*・成　久　洋　之**

*岡山理科大学工学研究科修士課程電子工学専攻

**岡山理科大学工学部情報工学科

（1993年9月30日　受理）

1．はじめに

高速な演算処理能力を持つスーパーコンピュータは従来の逐次型のノイマン型コンピュータと異なりベクトル命令という高水準の命令を使用することによりハードウェアで並列処理を実行している。その意味でベクトル計算機とも言われている。従って，並列処理の可能な部分が多いほど高速処理が期待できるわけであるが，与えられた問題構造によってはこの並列処理可能な部分の比率の低いものも存在する。逐次型計算機でのプログラムの中で，並列処理可能な部分の割合をベクトル化率と呼ぶが，このベクトル化率の高いほどベクトル計算機では性能のよいものとなる。

科学技術計算に使用されてきた逐次型計算のためのFORTRANプログラムをVPで使用するに当たって，効率を上げようと思えばプログラマが自らの手でVP用に改造してやる必要があった。しかし，その作業は容易ではなく，時間もかかる。しかも，高い並列性を実現するには豊富な知識と経験が必要である。そのような事情から，自動的にしかも高速にベクトル用プログラムに変換してくれるチューニングプログラムがあれば，その利用価値は絶大である。本研究ではスカラー演算用に作成されたプログラムを自動的にVP用プログラムに変換するためのチューニングシステムを検討するものである。

2．ベクトル計算機の概要

ここでは，ベクトル計算機の基本構造とベクトル演算による性能向上について述べる。

2.1　ベクトル計算機の構造

ベクトル計算機は，ベクトル化できないスカラ部分の処理を行うスカラ処理ユニットと，ベクトルデータを処理するベクトル処理ユニットから成り立っている。スカラ処理ユニットではベクトル処理ユニットを含めた計算機全体の制御も行っている。ベクトル処理ユニットではパイプライン演算器によってベクトルデータの処理を行っている。

ベクトル演算器は，ステージと呼ばれる機能単位に分けられており，データが各ステージを通過していく事により演算が行われる。これらの処理を行うパイプラインは加算用，乗算用など，複数の物が用意されており，計算機の種類によってその数もまちまちである。

2.2 ベクトル演算による性能向上

汎用計算機の演算処理は，一つの演算処理が終了しないと次の演算を行う事ができないが，ベクトル計算機であれば，一つの演算が終了しないうちに次の演算を行う事ができる。

FORTRAN プログラム中の DO ループがベクトル命令に自動変換されること（これはベクトル化と呼ばれる）によって性能が向上する。ベクトル計算機上での FORTRAN プログラムの性能は，このベクトル化される部分の大きさに依存する。したがって，ベクトル化される部分が大きいものほど性能が良い。図１にプログラムの性能向上率について示す。

図１の縦軸はプログラム全体の処理時間である。ベクトル化された部分がベクトル計算機では高速に実行されるためにプログラムの性能が向上する。

プログラムをチューニングするには，ベクトル化される部分を大きくすれば良い。チューニング作業はプログラムの修正を伴う大変な作業である。しかし，ベクトル計算機の場合には，チューニングによる性能向上の割合も大きいので必要な作業ともいえる。

3．ベクトル化について

本研究では対象言語として FORTRAN を取り扱っているが，ベクトル化の対象となるのは，FORTRAN プログラムの DO ループの範囲である。DO WHILE ループであるとか，DO UNTIL ループ，IF～GOTO ループなどはベクトル化の対象とはならない。データの型においても対象となるのは REAL＊8，COMPLEX＊8，COMPLEX＊16，REAL＊4，INTEGER＊4，LOGICAL＊4 であり，LOGICAL＊4，INTEGER＊2，REAL＊16，COMPLEX＊32，CHARACTER などは対象とならない。文の要素では算術式，論理式，文関数の引用，組み込み関数の引用などが対象となり，外部関数の引用は対象とな

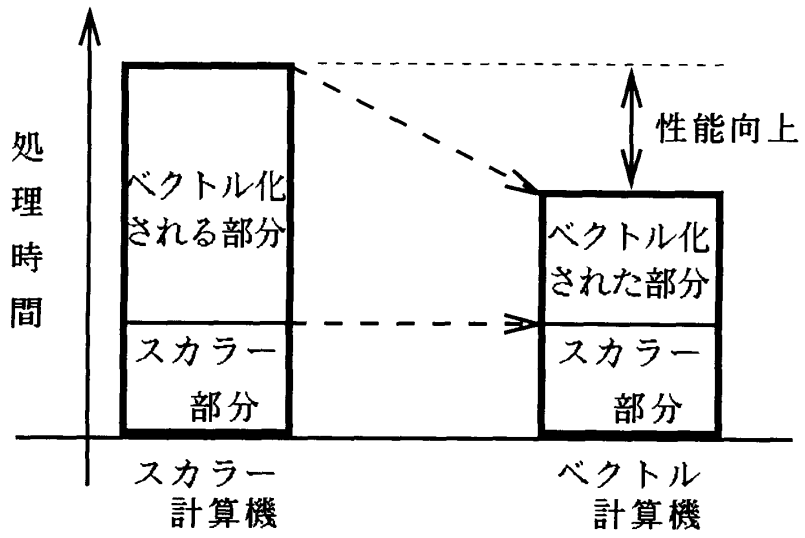

図１　プログラムの性能向上率

らない。文の種類で言うと算術代入文，論理代入文，算術 IF 文，ブロック IF 文，ELSE IF 文，ELSE 文，ENDIF 文，単純 GOTO 文，CONTINUE 文などが対象となり，CALL 文，入出力文，計算型 GOTO 文，ASSIGN 文，PAUSE 文，STOP 文，RETURN 文などは対象外である。データの定義・引用に関してはベクトル化できる順序，つまり回帰演算となっていなければベクトル化の対象となるが，回帰的なデータの定義・引用が行われている物については対象とならない。以上の事をまとめた物を表1に示す。

もちろんこれは対象とするハードとコンパイラの制限によるところが大きい。今回使用したのは富士通 FACOM VP-30E と，そのシステムを利用するための FORTRAN 77/VP コンパイラである。

3.1 ベクトル化の推進

(1) ベクトル化された部分をさらに高速化する。

・ベクトル長を大きくする

・演算機を並列利用できるよう DO ループを一元化する

・メモリアクセスの最適化をはかる

表1 ベクトル化の対象となる範囲

項　目	ベクトル化対象項目	ベクトル化非対象項目
対象範囲	DO ループ	DO WHILE ループ DO UNTIL ループ IF/GOTO ループ
データの型	LOGICAL＊4 INTEGER＊4 REAL＊4 REAL＊8 COMPLEX＊8 COMPLEX＊16	LOGICAL＊1 INTEGER＊2 REAL＊16 COMPLEX＊32 CHARACTER
文の要素	算術式 論理式 文関数の引用 組み込み関数の引用	外部関数の引用
文	算術代入文 論理代入文 算術 IF 文 ブロック IF 文 ELSE IF 文 ELSE 文 ENDIF 文 単純 GOTO 文 CONTINUE 文	CALL 文 入出力文 計算型 GOTO 文 ASSIGN 文 PAUSE 文 STOP 文 RETURN 文
データの定義/引用	ベクトル化できる順序（回帰演算でないこと）	回帰的なデータの定義/引用

(2) DO ループ内のスカラ部分をベクトル化する。

・外部手続き→外部手続きはベクトル化されないので外部手続きをそのルーチン内に組み込む

・回帰演算→回帰演算はベクトル化されないので，データの依存関係に回帰性がなければ，最適化制御行を使って回帰性のない事を明らかにする

(3) ベクトル化対象外の物をベクトル化する。

・データの型変換を行う

・IF～GOTO ループの DO ループ化（DO WHILE ループ，DO UNTIL ループについても同じ）

3.2 具体的なチューニング項目

以上のような事に基づいて，考えられるチューニング項目を列挙すると，以下のようになる。

(1) 外部関数，サブルーチンの展開

(2) 最も内側の DO ループベクトル長の増大化

(3) データの型変換

(4) メモリアクセスの最適化

(5) IF～GOTO ループの DO ループ化（DO WHILE，DO UNTIL）

(6) 回帰変数の解除

(7) 多重ループの解除

(8) DO 文の繰り返し回数の指定

(9) IF 文の真率の指定

(10) DO 文の分割

3.3 チューニング技法

具体的にどのようにチューニングを施すか，その技法について以下に説明する。

(1) 外部手続きの組み込みによるベクトル化

CALL 文や関数（ベクトル化される組み込み関数以外）はそのままではベクトル化されないので，呼び出したルーチン内に組み込む。

(2) 回帰演算の対処

DO ループ内の実行文はベクトル化によって，それぞれ単独の DO ループを構成したように実行順序が変更される。文の実行順序の変更によって定義引用の順序が変わる場合は実行結果が変わるのでベクトル化の対象にならない。実行順序の変更によって定義引用の順番が変わるデータを回帰的データと呼び，回帰的データの定義と引用を回帰的参照と呼ぶ。

文の実行順序が変更されても次に示すようなデータと引用関係の場合は実行結果は変わらず，ベクトル化の対象となる。

・実行順序が変更される文中に，同一データの定義と引用がない。

・同一データの定義と引用がある場合には，文の実行順序が変更されてもそのデータの定義と引用の順序が変更されない。

(3) ベクトル長を大きくする

ベクトル化するループの要素数は可能な限り多くなるように改める。要素数は，DO ループの繰り返し回数によって定まる。

(4) ベクトル演算の数を増やす

DO ループ内の演算の数が多ければそれだけベクトル長が大きくなるので，そのようにつとめる。

(5) メモリアクセスの最適化を図る

a．行方向のアクセスは列方向のアクセスに変換した方がメモリ上を連続アクセスできる。

b．メモリ上連続したデータであっても，負の方向にアクセスする場合は正方向に変換した方がよい。

c．2次元配列を計算に進める場合，配列の寸法を変更してメモリバンクの競合を回避した方がよい。その寸法の計算方法を表2に示す。

(6) 最適化制御行を使ってベクトル化された部分を更に高速化する

a．IF 文の真率を指定する。

b．DO 文の繰り返し回数を指定する。

(7) データの型変換を行う

ベクトル化されるデータの型は決まっているので，もしプログラム上問題がなければベクトル化されないデータの型を変換する。

4．チューニングプログラム

今回作製したチューニングプログラムは，外部関数・サブルーチン展開とデータの型変換，メモリアクセスの最適化に関するものであり，それぞれについての具体的な手順等は以下の様になる。

表2　配列寸法の計算方法

type＊m	REAL＊4 INTEGER＊4 LOGICAL＊4	REAL＊8 COMPLEX＊8 COMPLEX＊16
K	4 n + 2	2 n + 1

type＊m A (K, N)

4.1 外部関数・サブルーチン展開用のプログラム

手　順：

step 1 — 字句解析，構造解析を行い，必要な情報を収集する

step 2 — 外部関数，サブルーチンが定義・引用されているところを見つけだす

step 3 — 仮・実引数をストックする

step 4 — 仮引数を実引数に置き換え，展開する

step 5 — 引用部（行）を削除するなどの後処理

仕　様：

・扱える最大ステップ数……1024ステップ

・扱える最大 FUN & SUB 数……合計64個

・扱える最大引数並び……16個

制　限：

・COMMON 文には対応していない

4.2 データの型変換用のプログラム

手　順：

step 1 — ベクトル化されないデータ型でなおかつ変換が可能である型宣言部を見つけだす（MAIN，SUB，FUNCTION 全てについて）

step 2 — 対応するベクトル化可能な型に置き換える

仕　様：

・扱える最大ステップ数……1024ステップ

・扱える最大コンバート数……制限無し

・変換されるのは

LOGICAL＊1 → LOGICAL＊4

INTEGER＊2 → INTEGER＊4

である。

4.3 メモリアクセスの最適化用のプログラム

手　順：

step 1 — データの型宣言部を見つけだす

step 2 — 2次元配列のみを選び，その配列の種類を調べる

step 3 — 配列の要素数をストックする

step 4 — 種類ごとに最適な配列の要素数を計算する

step 5 — 計算結果を元に要素数を修正する

仕　様：

・扱える最大ステップ数……1024ステップ

・扱える最大コンバート数……制限無し

・変換されるのは2次元配列の

REAL＊8, COMPLEX＊8, COMPLEX＊16, REAL＊4, INTEGER＊4, LOGICAL＊4

である。

5．結果及び考察

チューニングプログラム及び性能評価用のサンプルプログラムの開発に当たっては NEC の PC-9801を使用した。開発言語は FORTRAN で MS-FORTRAN ver 5.1にてコンパイルを行った。そのほかにプログラムの性能改善に FORTUNE と，チューニング作業を支援するシステムである VECTUNE を使用した。

5.1 サンプルプログラムを用いた変換結果

外部関数，サブルーチンに対する処理は多少違うので別々にサンプルプログラムを作成して変換した。どちらもきちんと展開されており，引き数の引き渡しなども問題なく処理されていた。残りのデータの型変換，メモリアクセスの最適化のプログラムは，ただのサーチ&スワップなので問題なく動作した。変換速度は NEC PC-9801 RA で1秒以内であった。

作成したチューニングプログラムで試しにサンプルプログラムをチューニングして，チューニング前とチューニング後の VP での処理時間を実測したものを表3に示す。また，その際のチューニング前とチューニング後のプログラムを次に示す。

表3 チューニング前後での処理時間

	チューニング前	チューニング後
処理時間(cost)	18002	1594

チューニング前：

```
      INTEGER I
      DO 100 I= 1, 1000
        S=S+MENSEK (I)
  100 CONTINUE
      WRITE (6, *) 'SUM=', S
      STOP
      END
      FUNCTION MENSEK (A)
        INTEGER A
        MENSEK=3.1415*A*A
      END
```

→

チューニング後：

```
      INTEGER I
      DO 100 I= 1, 1000
        MENSEK=3.1415*I*I
        S=S+MENSEK
  100 CONTINUE
      WRITE (6, *) 'SUM=', S
      STOP
      END
```

5.2 考　　察

表3より，チューニングによって処理時間が短縮されたことがわかる。今回作成した(1), (3), (4)の項目によって期待できる性能向上率は対象プログラムに占める外部関数・サブルーチンの実行コスト率などによるので予想できないが，実際のプログラムでは外部関数・サブルーチンの利用率が高いためかなりの効果が上がると思われる。

今回は，構造解析を比較的必要としない，いわゆる実現がより容易な項目についてプログラムを作成した。全体として言える事は，チューニング項目のうち，実現可能な項目としては(1)～(4)ぐらいであり，(5)～(10)については，実現はかなり困難である。(2), (7), (8), (10)の項目はDOループの構造解析を行うという点では共通しており，(6)もその上で変数の依存関係を解析すれば，実現は可能と思われる。

Tuning Program for VP Computer

Yasutoshi OKADA*, Hiroyuki NARIHISA**

*Graduate School of Engineering

**Faculty of Engineering

Okayama University of Science,

Ridaicho 1-1, Okayama 700, Japan

(Received September 30, 1993)

In order to enhance the power of high speed calculation applicable to the weather forcasting or newclear simulation, supercomputer has been appeared. In our country, there are various kinds of supercomputers. However, almost domesticmade supercomputer are vector processing computer. Therefore, we must provide vertorized algorithm for using supercomputer obtaining high speed computing. Even if we use supercomputer, a few vectoring rate program is equivalent to the originally sequential conventional von Neumann computer program. At this point of view, it is necessary to increase the vectorizing rate of VP algorithm as hard as one can for sake of increasing VP usage efficiency. However, making these high efficient VP algorithm is not so easy especially for average supercomputer users. From these circumstances, it is very convenient if we can zse a automatically transformation program from a usually used sequential type FORTRAN program to a VP applicable vectorized program.

In this paper, we present a VP tuning program which a sequential FORTRAN program is automatically transformed to a vectorized program.