

12

配列の計算を速くするサブルーチンパッケージ
"HISARP" について

日本IBM 加藤 隼一

§ 1. 主旨

配列処理の高速化のためにアセンブラ言語で多数のサブルーチンを作成し，これらに対応する FORTRAN サブルーチンと比較して，実行速度につき最高約 20 倍の能率が得られた。

§ 2. 配列処理の高速化

電子計算機のプログラムの中には，所要時間の大部分が配列の計算のために費やされ，しかもこれらの計算のほとんどが配列のごく基本的な処理の組み合わせになっているような種類のものは多い。筆者が数年来携わってきた構造解析関係の多くのプログラムはこのような傾向が顕著にみられる例である。数値計算に広く用いられている連立一次方程式の数値解法のプログラムなども，主要な部分は配列の基本的な処理によって占められている。

従って，計算を能率化するための方法として，モデル化の

段階での工夫や全体的なアルゴリズムの改良の他に、配列処理の高速化ということが重要になってくる。

配列の処理を高速化するための *approach* を次の3つに大別してみよう。

- ① 配列処理アルゴリズムの工夫 (例: 行列積の計算における *Winograd* の方法。FFT.)
- ② ソフトウェアによる高速化 (例: コンパイラーによる *optimization*)
- ③ ハードウェアによる高速化 (例: 配列処理のための専用演算回路やマルチプロセッサ)

このうち、②に属する *approach* について考えてみる。コンパイラーによる *optimization* の例として、IBM の FORTRAN H コンパイラーの場合は、使用者の選択 (OPT=0, 1, 2 のいずれか) により、コンパイルの速さ、目的プログラムの小ささ、目的プログラムの実行の速さのうちのいずれかを重視してコンパイルを行うようになっている。

FORTTRAN プログラムの

```
SPL=0.0
```

```
DO 100 I=1,N
```

```
100 SPL=SPL+X(I,I)
```

という部分をコンパイルする場合、DO ループの実行の途中

で、和を毎回記憶域 SPL に蓄える必要はなくレジスターに置いたままにすることができることや、配列要素 $X(I, I)$ のアドレスを求めるのに毎回直接 I の値にまでさかのぼる必要がないことなどを活かしてコンパイルを行えば、より効率のよい目的プログラムを生成することができる。

しかしふつう、コンパイラーによる *optimization* の結果にも、まだ高速化の余地がかなりある。そこで筆者は、FORTRAN プログラムの配列処理の部分を CALL ステートメントに置きかえ、予めアセンブラー言語で書いておいた高速サブルーチンに実行させるという方法をとった。これは②に属するもう一つの方法である。初めはコンパイラーの *optimization* の機能を単に補完するだけのものではあったが、その後いくつかの面でそれ以上のものとなりつつある。

§ 3. HISARP の概要

HISARP (High Speed Array Processor) は、筆者が構造解析のアプリケーションプログラムの開発に際して配列の高速処理のために作成し使用していた種々のサブルーチンを改良し、新たに系統的に作成して、一つのパッケージにまとめたものである。以前にまとめたもの (*version 1*) と、最近これに更に改良を加えて新たに整備しつつあるもの (*version 2*) とがある。

- 1) HISARP は、配列に関する種々の、ごく基本的な処理、演算を、高速で実行するためのサブルーチンの集まり（パッケージ）である。
- 2) FORTRAN プログラムから CALL できる。
- 3) アセンブラー言語で書かれた（マクロ命令で生成された）ものである。即ち、IBM OS/360 アセンブラー言語（Hレヤル）を使って定義したマクロ命令により、すべてのサブルーチンが系統的に生成された。（その副次的な効果として、サブルーチンの個数の多さ（*version 1* では現在 315 個）にもかかわらず、虫取りが容易であった。）
- 4) 各サブルーチンは
 - i) それぞれ単一の機能を有する。
 - ii) 実行時間に関して（特にシステム 360 モデル 195 向きに）*optimize* してある。
 - iii) 一部の例外を除いて *self-relocatable* かつ *re-entrant* である。
- 5) *optimize* に際しては
 - i) まず第一に、ループの内部を最優先した。
ただし *version 1* と *version 2* とではループの構造が異なる。（後述）
 - ii) ループの外部もできるだけ *optimize* した。その結

果, 処理される配列が小さくても効率のよさがあまり減殺されない。

6) 扱える配列のタイプは, 整数(1バイト長, 半語長, 1語長)と, 実数および複素数(いずれも1倍精度, 2倍精度, 4倍精度)とで, 合計9種類である。

7) システム360のモデル20以外およびシステム370のすべてのモデルにおいて使用可能。ただし4倍精度の演算を含むサブルーチンは4倍精度命令セットを備えたモデルでのみ使用可能。

§4. サブルーチン名

HISARPの機能の説明の一助に, HISARPのサブルーチンの名づけ方を記す。サブルーチンの名前を5文字または6文字とし, 次のように3つの部分から構成する。

サブルーチン名 = *type* · *mode* · *operation*

*type*は, 処理される配列のタイプを表わすためのもので, 1文字または2文字からなる。

type = B/H/I/S/D/Q/CS/CD/CQ

B: バイト整数

H: 半語長整数

I: 1語長整数

S: 1 倍精度実数

D: 2 倍精度実数

Q: 4 倍精度実数

CS: 1 倍精度複素数

CD: 2 倍精度複素数

CQ: 4 倍精度複素数

mode は, 配列中の要素のうち, どのような並び方のものを処理対象として扱うかを表わすための 1 文字である。

mode = M/V/T

M: *matrix mode* (マトリックスの全体または一部を扱う。ただし配列中のマトリックス要素の排列は FORTRAN 流に *columnwise* であるものとする。)

V: *vector mode* (連続したいくつかの要素を扱う。)

T: *step mode* (一定個毎の要素を扱う。)

なお, マトリックスの場合, V *mode* で列を, T *mode* で行をそれぞれ扱うことができる。

operation は, 処理や演算の種類を表わすためのもので, 3 文字からなる。

operation = INI/SCL/ADD/SUB/MOVB/XCH/SNA/
ACC/ASQ/IPR/PIV/.....

INI: 初期値化 $x_a \leftarrow c$

SCL: スカラー倍	$x_\alpha \leftarrow Cx_\alpha$
ADD: 加算	$x_\alpha \leftarrow x_\alpha + y_\alpha$
SUB: 減算	$x_\alpha \leftarrow x_\alpha - y_\alpha$
MOV: 移動	$x_\alpha \leftarrow y_\alpha$
XCH: 交換	$x_\alpha \leftrightarrow y_\alpha$
SNA: スカラー倍を加える	$x_\alpha \leftarrow x_\alpha + Cy_\alpha$
ACC: 合計	$a \leftarrow \sum_\alpha x_\alpha$
ASQ: 平方和	$a \leftarrow \sum_\alpha x_\alpha^2$
IPR: 内積	$a \leftarrow \sum_\alpha x_\alpha y_\alpha$
PIV: 絶対値最大要素を探す。	

(ただし, 1 または 2 次元の添字 α の集合は *mode* による。
mode と *operation* との特別な組み合わせ (または融合した
 もの) として, 現在次の 2 つがある。

MPRD: マトリックス $\times$ マトリックス ($P \leftarrow {}^tXY$)

MVPR: マトリックス $\times$ ベクトル ($\vec{P} \leftarrow {}^tX\vec{y}$)

version 1 のサブルーチンの現在の個数 315 は, 以上のす
 べての組み合わせの数 $9 \times (3 \times 11 + 2)$ である。

§ 5. *version 1* / *version 2*

version 1 と *version 2* との相異は, サブルーチン内のル
 ープ (2 重以上のループになっている場合には一番内側のも

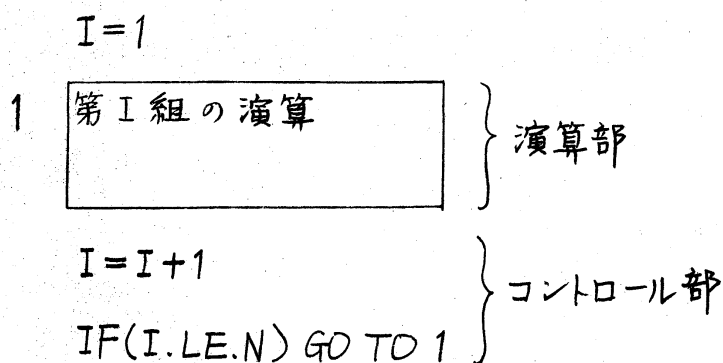
の。以下でも同様。)の構造の相異に基づいている。

version 1 のループの構造は, FORTRAN の DO ループをそのまま *optimize* した形になっている。すなわちループは, データそのものの演算のための 1 組の命令群 (演算部) と, 配列要素のアドレスの更新やループコントロールのための 1 組の命令群 (コントロール部) とで構成されている。

version 2 のループは, m 組の演算部と 1 組のコントロール部とで構成されており, ループを 1 回まわる毎に m 回分の演算が実行される。 m を詰め込み係数と呼ぼう。このループでの演算の必要実行回数を N とすれば, m 組の演算部のうち後から $\text{mod}(N-1, m)+1$ 番目の組からループの実行を開始すればよい。ループを抜け出すのは必ずループの最後からである。

従って, HISARP のループの構造を FORTRAN 風に表現すれば次のようになる。

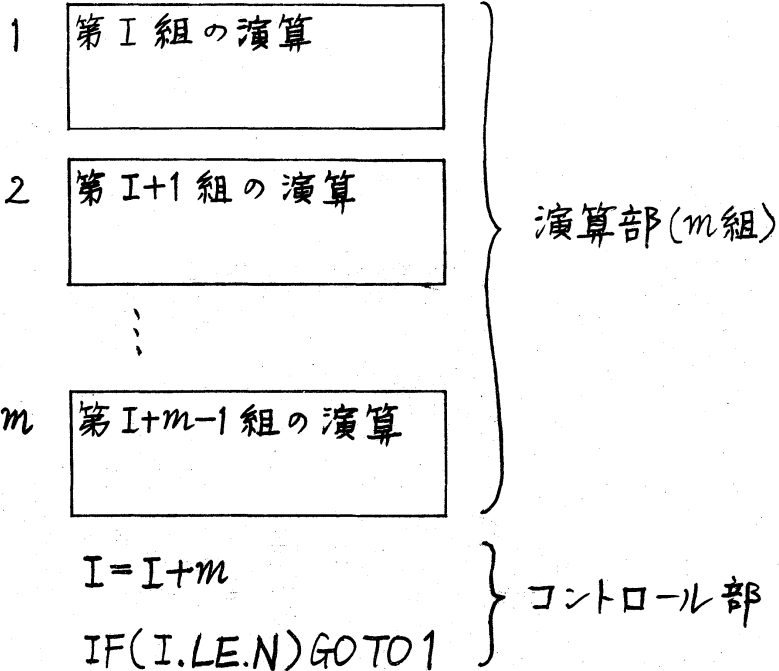
version 1:



version 2:

$$L = m - \text{MOD}(N-1, m)$$

$$I = 2 - L$$

$$\text{GO TO } (1, 2, \dots, m), L$$


HISARPを受け容れるシステム 360, 370 の各モデルの中でも、特に適当なモデル 195 は、いわゆるパイプラインコンピュータの 1 つであり、本文の内容に関連して次のような特徴を持っている。

- 1) 命令を先取りして一時蓄えておくための、容量 8 ダブルワードの棚が CPU 内にあり、1 つのループ全体がこの棚に納まると処理効率が上る。(ループモード)
- 2) 主記憶域へのアクセスタイムを縮めるために、容量が 32K バイト = 4K ダブルワード (ただし $K = 2^{10} = 1024$)

の高速バッファを有する。

3) CPU内での機能が分化し、演算の並行処理能力が高いので、個々の演算が速いだけでなく、全体としての実効速度が極めて大きくなり得る。

4) ブランチ命令は、他の命令に比較して割合遅い。

version 1 のほとんどのループはループモードで実行される。*version 2* のループも原則としてループモードで実行されるようにし、その範囲内で詰め込み係数 m を最大にした。

一般に、*version 2* は *version 1* に比較して、次のような特徴を持ち、より高速になっている。

1) ループ実行中の演算の並行処理の程度が高い。

2) ブランチ命令の回数が少い。

§6. 実行時間の比較

HISARP に含まれる 2 種類のサブルーチン DVACC (配列の要素の合計), DMPRD (行列積) (いずれも 2 倍精度実数演算ルーチン。§4 参照。)につき、対応する FORTRAN プログラムを作成し、(プログラムリスト参照)

② FORTRAN G コンパイラでコンパイルしたもの。

④ FORTRAN H コンパイラでコンパイルしたもの。
(OPT=2 を指定して optimize。§2 参照)

③ HISARP *version 1*

④ HISARP version 2

の能率テストをモデル 195 で行い、次の要領で単位演算当りの平均所要サイクル数 C を算出して比較した。(グラフ参照)

$$C = T_{EX} / I / J / T_c$$

$$J = \begin{cases} N & (\text{DVACC の場合。 } N \text{ は要素の個数。}) \\ N^3 & (\text{DMPRD の場合。 } N \text{ は行列の次数。}) \end{cases}$$

T_{EX} : 実行時間 (CALL ステートメントの時間も含む) の測定値

I : 繰り返し CALL 回数

J : CALL 1 回当りの単位演算の実行回数

T_c : モデル 195 のマシンサイクルタイム (54 nsec.)

能率テストは、同一の引数でサブルーチンを繰り返し CALL し、実行時間が 1 秒のオーダーになるようにした。このようなテストも、各サブルーチンにつき、 N のいくつかの値について行った。なお、「単位演算」は、DVACC の場合は要素 1 個分の加算を、また DMPRD の場合は 1 対の要素の乗算および加算を意味し、 C には一切のオーバーヘッドの負担分が含まれている。

version 2 の DVACC は、詰め込み係数 $m=14$ で、4 個の浮動小数点レジスターに交互に加算を行っているので、 C の値が、オーバーヘッドを含んでいてなおかつ、加算単独の所要サイクル数 2 より小さくなった場合がある。(グラフ参照)

これは、モデル 195 の浮動小数点加算器が、互いに独立な加減算なら 1 サイクル毎に処理を開始することができる、という理由による。ただし複数個のレジスターを使うので、加算の結合の組み合わせが異なってくるために、DVACC に関しては、結果の値に含まれる丸め誤差について、2 つの version の間で差が生じる。

version 2 の DMPRD は、詰め込み係数 $m=5$ であり、 C の値が、乗算と加算の所要サイクル数の単純和より小さくなった場合がある。これは加算と次の乗算とが並行処理された結果である。

C の値に含まれているオーバーヘッドの負担分は N の値の増加とともに減少し、従って C は単調減少の傾向を示すはずであるが、テスト結果では、DVACC の場合 $N=3000$ から $N=10000$ の途中で、また DMPRD の場合 $N=60$ から $N=80$ の途中で、 C の値が増加している。これは、繰り返し取り出されるデータが高速バッファの容量 (4096 ダブルワード) を越すと、高速バッファの効力が低下することに起因している。
(グラフ参照) (Note 参照)

DMPRD (行列積 $P \leftarrow {}^tXY$) の場合、 X 全体と Y の 1 列分とが高速バッファに納まらなくなる (すなわち N^2+N が 4096 を越える) 時に、 C の値が増加するはずであり、テスト結果と一致する。 X, Y 全体が高速バッファに納まらなくなる時

も、Cが増加するが、わずかに過ぎない。

Cの値から、②, ③, ④, ⑤の実行時間の比を求めると、ほぼ次の如くである。

DVACCの場合 20.4 : 3.7 : 3.7 : 1 (N=3000の時)

DMPRDの場合 17.2 : 2.9 : 1.6 : 1 (N=40,60の時の平均)

☆ プログラムリスト

(比較のために使われた FORTRAN プログラム)

```

SUBROUTINE DVACC (ACC,X,ICT)
REAL*8 ACC,X(1)
ACC=0.000
DO 100 I=1,ICT
100 ACC=X(I)+ACC
RETURN
END

SUBROUTINE DMPRD (P,X,Y,ICIMP,ICIMX,ICIMY,ICT,JCT,KCT)
REAL*8 P(ICIMP,KCT),X(ICIMX,JCT),Y(ICIMY,KCT),W
DO 200 K=1,KCT
DO 200 J=1,JCT
W=0.000
DO 100 I=1,ICT
100 W=X(I,J)*Y(I,K)+W
200 P(J,K)=W
RETURN
END

```

☆ Note: 80次の行列の積を求める場合、次のように2度に分けて処理すれば、高速バッファの機能を活かせる。

```

DIMENSION P(80,80),X(80,80),Y(80,80)
CALL DMPRD(P,X,Y,80,80,80,80,40,80)
CALL DMPRD(P(41,1),X(1,41),Y,80,80,80,80,40,80)

```

☆ 単位演算当りの平均所要サイクル数C

- (a): FORTRAN G コンパイラでコンパイルしたもの。
 (b): FORTRAN H コンパイラでコンパイルしたもの。
 (c): HISARP version 1
 (d): HISARP version 2

