数 値 解 析 28－5
（１９８９．３．１４）

ベクトル計算機による並列処理のシミュレーション

東芝ＣＡＥシステムズ　　東芝

福井　義成　　吉田　浩俊　　肥後野恵史

スーパーコンピュータのコンパイラでは条件分岐を含む“ＤＯループ”のベクトル化も可能になっているが，複雑な条件分岐をふくむ場合はまだベクトル化が不可能である．たとえば，偏微分方程式のように分布定数系としてモデル化したものはベクトル化しやすいが，ネットワークなどのように集中定数系としてモデル化したものはベクトル化しにくい．ここでは複雑な条件分岐を含むプログラムを高速に計算する一つの方法について述べる．ＣＲＡＹ　Ｘ－ＭＰにおけるマルチタスキングの経験を応用し，複雑な条件分岐をもつプログラムをベクトル化し高速化した．この方法は同じ条件のデータに対するクラスタリング法であり，ベクトル計算機で並列計算機をシミュレーションしているとも考えることができる．この方法で 2.14～ 2.63 倍の高速化ができた．

Simulation of parallel computation by vector computer

Yoshinari Fukui, Hirotoshi Yoshida

TOSHIBA CAE SYSTEMS, INC.
580-1, HORIKAWA-CHO, SAIWAI-KU, KAWASAKI 210, JAPAN

Shigefumi Higono

TOSHIBA CORPORATION
580-1, HORIKAWA-CHO, SAIWAI-KU, KAWASAKI 210, JAPAN

In this paper, we describe a vectorization technique for complicated conditional branch statements. Current compilers of super computer is capable of vectorizing some conditional branch statements, but has difficulty vectorizing complicated conditional branch statements. Our method is generalized vectorization method for complicated branch statements, without any needs for the global consideration for the data structure. This is based on simulating parallel computation by vector computer.

1. はじめに

スーパーコンピュータの普及には自動ベクトル化コンパイラの発達が重要な役割をはたしたと考えられる. 現在の自動ベクトル化コンパイラでは条件分岐を含む“ＤＯループ”のベクトル化も可能になっているが, 複雑な条件分岐場合（条件分岐のレベルが深い場合）はまだベクトル化が不可能である. ここでは複雑な条件分岐を含むプログラムをベクトル化し高速に計算する一つの方法について述べる. この方法は同じ条件のデータに対するクラスタリング手法の一種であり, ベクトル型スーパーコンピュータで並列計算機をシミュレーションしているとも考えることができる.

2. 高速化

スーパーコンピュータでは, ベクトル化することにより, 汎用計算機に比べて非常に高速に結果を得ることができる. ところが解析ソフトの中には複雑な条件分岐をもつためベクトル化が困難なものもある. 偏微分方程式のように分布定数系としてモデル化したものはベクトル化しやすいが, 制御系の問題や回路解析のように集中定数系としてモデル化したものはベクトル化しにくい.

計算の高速化について考えると, 一般に高速化の効果は高速化にかけた手間に依存する. しかし, ある程度以上高速化されたプログラムはそれ以上手間をかけてもあまり速くならない. 高速化にかけた手間と得られた結果（高速化）が最良になる点がある. このバランス点は計算の規模や頻度に依存する. また, 高速化にも色々なレベルが考えられる[1].

（１）無駄な計算の削除
（２）コーディングの変更
（３）データ構造の変更
（４）アルゴリズムの変更
（５）モデル化の変更

５つの項目では, 後になるほど高速化の効果が大きい. 新たにプログラムを作成する場合は, 上記のことを考慮してプログラムを作成すればよいが, すでに作成されたプログラムでは変更に必要とする手間も多くなる.

複雑な条件分岐をもつプログラムのベクトル化は, 問題ごとにプログラムの全体をよく分析し, ベクトル化しやすいように変更することによって可能である. しかし, これには各プログラムごとに多大の労力を必要とし, プログラムの大規模な変更にも対応しにくい面がある. そこで, 局所的な変更を組合せることにより, 複雑な条件分岐をもつプログラムに対してベクトル化が可能となる方法について述べる.

自然現象の多くのものは並列に現象が進行しており, 並列計算が可能なものが多い[2],[3]. ベクトル化も並列計算可能なものの処理の一方法であるが, このような問題には複数のプロセッサを同時に使用した並列処理が向いてる. ＣＲＡＹ　Ｘ－ＭＰでの並列処理（マルチタスク化）に関する報告はすでに行った[4].

マルチタスキングでは複数のプロセッサを使用し，多くのＣＰＵリソースを必要とするため，同じ問題に対して単一のプロセッサですむベクトル化を行なった．

3．　高速化の実例

スーパーコンピュータの普及には自動ベクトル化コンパイラの発達が重要な役割を果たした．しかし，自動ベクトル化コンパイラでも複雑な条件分岐場合（条件分岐のレベルが深い場合）はまだベクトル化が不可能である．回路解析プログラムＳＰＩＣＥ－ＧＴは，カリフォルニア大学で開発されたＳＰＩＣＥ２　Ｇ．６を改良強化したバージョンである．回路解析は基本的に集中定数系の解析であり，ＳＰＩＣＥにはベクトル化に関する考慮がまったくなされていない（プログラム中にも“ＤＯループ”はほとんど存在しない）．また，データ構造がリスト構造となっており，そのままではベクトル化はまったく不可能である．

回路解析の高速化の場合，３つの部分にわけて考えることができる．

（１）　回路行列の構成
（２）　回路行列を解く
（３）　その他

回路行列の構成の部分は，並列計算は可能であるが複雑な条件分岐をもつため現在ではベクトル化が困難な部分である．この部分をマルチタスキング化[4]の考え方を応用してベクトル化を行った（回路行列を解く部分はコード生成によって高速化を行った[5]）．　複数のＣＰＵで並列処理を行うことを考え，実際にはベクトル命令で複数個のＣＰＵによる並列計算をシミュレートする（ベクトル型スーパーコンピュータで並列計算機をシミュレーションしていると考えることができる）．

4．クラスタリングとベクトル化

スーパーコンピュータのコンパイラは比較的単純な条件分岐はベクトル化を行ってくれる（図１）．ベクトル化の対象となるのは，主に条件分岐が単純で比較的小さな“ＤＯループ”内で完結している場合である．しかし，複雑な条件分岐が広範囲にわたっている場合はベクトル化が困難である（図２）．このように複雑な条件分岐が広範囲にわたっている場合のベクトル化を考える．個々の計算は原理的には並列処理可能であることを前提にしており，処理すべきデータをベクトル化するため，並列処理すべき変数ごとにベクトル化の作業用領域（１次元配列）を用意する．条件分岐ごとに同じ条件のものを集め，クラスタリングを行う．条件分岐を処理するため（クラスタリング）のインデックス変数を用意し，条件分岐ごとにその中のポインターの順序を入れ換えることにより，クラスタリングを行う（図３，４）．

クラスタリングで２つのグループに分け，１つのグループ（グループ１）には処理を継続する．残りのグループ（グループ２）に対してはグループの情報はスタックに蓄えておき，グループ１に対する処理が完了したのち処理を再開する．

```
      DO 100 I=I1,I2,I3
        IF(A(I) .GE. 0.0) THEN
          B(I)=C(I)**2
        ELSE
          B(I)=D(I)
        ENDIF
  100 CONTINUE
```

図１．ベクトル化可能な条件判定の例

```
      K=0
   10 CONTINUE
        IF(K .GT. 10000) GO TO 1000
        N=NN(K)
        A=Z(N  )
        B=Z(N+1)
        C=Z(N+2)
        GO TO ( 100, 200, 300 ), IPOINT(N)
  100   CONTINUE
          CALL SUB1
          Z(N+3)=F
        GO TO 900
  200   CONTINUE
          CALL SUB2
          Z(N+4)=G
        GO TO 900
  300   CONTINUE
          CALL SUB3
          Z(N+5)=H
        GO TO 900
  900   CONTINUE
          K=K+1
        GO TO 10
 1000 STOP
```

図２．ベクトル化不可能な条件判定の例

インデックス変数の内容（条件分岐以前）

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17

↓

条件判定の真偽

T	F	F	T	T	F	T	T	F	T	T	F	T	F	F	T	F

↓

インデックス変数の内容（条件分岐後）

1	4	5	7	8	10	11	13	16	2	3	6	9	12	14	15	17

←－－－－Ｔのインデックス－－－→　←－－－Ｆのインデックス－－→

図３．クラスタリングの例

```
      IF(A .LE. 0.0) GO TO 200

                ‖
                ▽

      J=IVV1
      K=0
CDIR$ IVDEP
      DO 120 I=IVV1,IVV2
      IF(AA(LV1(I)) .LE. 0) THEN
        K     =K+1
        LV2(K)=LV1(I)
      ELSE
        LV1(J)=LV1(I)
        J     =J+1
      ENDIF
  120 CONTINUE
      DO 125 I=1,K
        LV1(I+J-1)=LV2(I)
  125 CONTINUE
```

図４．クラスタリング

以下のような手順でベクトル化を行う.

（１）　条件分岐のためのインデックス変数（１次元配列）を用意する.

（２）　インデックス変数の中で処理の対象となる部分を示すポインター変数（スカラー，処理の先頭と最後）を用意する.

（３）　インデックス変数の中の未処理の部分を蓄えておくスタック変数（１次元配列：処理の先頭と最後および処理を再開する場所）を用意する.

（４）　ベクトル化を行う部分を決定する.

（５）　ベクトル化の対象となった部分で値が変更されている変数に対して作業用の変数を用意する.

（６）　文番号および条件分岐で区切られる部分（ブロックと呼ぶ）を“ＤＯループ”化する（図５，６）.

（７）　現在でもベクトル化できる比較的単純な条件分岐はそのままコンパイラにまかせてベクトル化する（図７，８）.

（８）　複雑な条件分岐はすべて２方向分岐に変換する．条件分岐ごとにクラスタリングを行い，グループ１の実行を継続し，グループ２の情報をスタックに保存しておく.

（９）　先に処理したの実行がすべて完了したのち，スタックに保存されたグループの先頭から情報を取り出し，そのグループが処理を中断した場所から処理を再開する.

表１のようにベクトル化をおこなった部分で２倍以上の高速化が実現できた．全体でも２倍程度高速化されている.

5．まとめ

複雑な条件分岐をもつプログラムをベクトル化する方法について述べてきたが，この方法では全体的な考慮をせずに部分的な変更でベクトル化が可能である．回路解析での例では２倍以上の高速化ができた．今後，複数ＣＰＵによる並列処理が多くなるであろうと考えられる．本方法はもともと複数のＣＰＵで並列処理するのをベクトル命令でシミュレートしているため，処理し残したグループをスタックから取り出す部分を少し変更するだけで，複数のＣＰＵで並列に処理可能となる.

```
A=B**2
C=D+H
E=F+G
```

図5．ベクトル化前のプログラム

```
      DO 300 I=IVV1,IVV2
        AA(LV1(I))=BB(LV1(I))**2
        CC(LV1(I))=DD(LV1(I))+HH(LV1(I))
        EE(LV1(I))=FF(LV1(I))+GG(LV1(I))
  300 CONTINUE
```

図6．ベクトル化後のプログラム

表1．ベクトル化による高速化

モデル	要素数	モード	ベクトル化部分のみ		全体	
			秒	比率（倍）	秒	比率（倍）
1	15	スカラー	34	1．11	41	1．08
		ベクトル	31		38	
2	45	スカラー	34	1．86	37	1．74
		ベクトル	18		21	
3	150	スカラー	34	2．42	36	2．25
		ベクトル	14		16	
4	450	スカラー	36	2．54	39	2．26
		ベクトル	14		17	
5	1500	スカラー	120	2．63	138	2．19
		ベクトル	46		63	
6	1884	スカラー	189	2．14	261	1．62
		ベクトル	88		161	
7	3000	スカラー	242	2．58	300	1．96
		ベクトル	94		153	

```
      IF(A .LT. 1.0) GO TO 20
        B=1.0
      GO TO 40
   20 CONTINUE
        B=2.0
   40 CONTINUE
```

図7．単純な条件分岐の例

```
      DO 400 I=IVV1,IVV2
        IF(AA(LV1(I)) .GE. 1.0) THEN
          BB(LV1(I))=1.0
        ELSE
          BB(LV1(I))=2.0
        ENDIF
  400 CONTINUE
```

図8．単純な条件分岐をベクトル化した例

参考文献

[1] Metcalf, M. : FORTRAN OPTIMIZATION, ACADEMIC PRESS, 1982

[2] 川合俊雄: シミュレーション的自然観と並列計算機, ｼﾐｭﾚｰｼｮﾝ, Vol.5, No.2, pp.64-70, 1986.

[3] 星野力: ＰＡＸコンピュータ, オーム社, 1985.

[4] 福井, 大吉, 加藤, 渡辺: マルチプロセッサ・システムにおける回路解析コードの並列処理, 情報処理学会数値解析研究会資料, 17-5, 1986.

[5] 福井: 問題の性質を利用した大規模スパース行列の高速解法, 情報処理学会数値解析研究会資料, 20-3, 1987.