数理解析研究所講究録
第648巻 1988年 77-86

複雑な条件分岐をもつ解析ソフトの高速化

東芝CAEｼｽﾃﾑｽﾞ　福井　義成 (Yoshinari Fukui)
吉田　浩俊 (Hirotoshi Yoshida)
東芝　肥後野恵史 (Shigefumi Higono)

1．はじめに

計算機による解析（ｼﾐｭﾚｰｼｮﾝ）が行われることが多くなり，その解析も非常に大規模なものになっている．このような要求に応えるため出現したのがスーパーコンピュータで，計算機の解析能力を特に高めたものとなっている．最近ではスーパーコンピュータも普及し，“普通の計算機”になりつつある．このような状況では「“スーパーコンピュータ”をどのように有効利用するか？」が問題となる．

2．高速化

各種の解析をスーパーコンピュータで行う場合，ベクトル化により，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，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．

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，成長曲線のように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．どこかに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と思われる．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次に述べる．

3．　高速化の実例

高速化を追求した例について述べる．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 ﾙｰﾌﾟ”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

回路行列を解く部分はコード生成によって約２６倍の高速化を行った[2]．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[3]の考え方を応用してベクトル化を行った．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．（この例は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．）

並列に処理できるデータをベクトル処理するため，関係する変数ごとにベクトル化の作業用領域（１次元）を用意する．また条件分岐を処理するためのインデックス配列を用意し，条件分岐ごとにその中のポインターの順序を入れ換え，クラスタリングを行う（図１）．クラスタリングで２つのグループに分け，グループ１の処理を継続する．グループ２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．処理を再開する場所を記憶するため，“ ASSIGN GO TO ”を利用する．以下のような手順でベクト

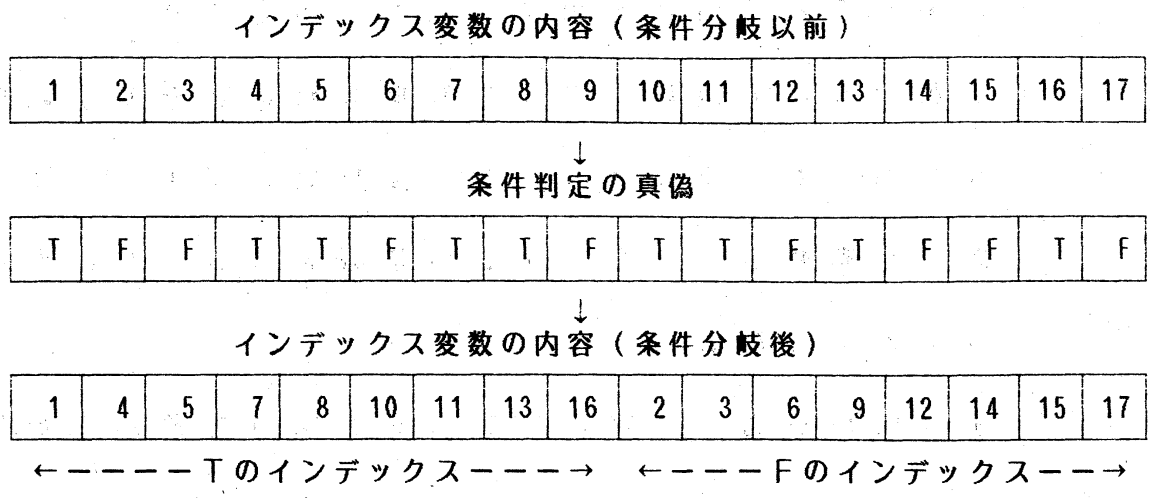

図１．クラスタリングの例

ル化を行う．

（１）条件分岐のためのインデックス配列（１次元）を用意する．　LV1

（２）インデックス配列のポインター（先頭と最後）を用意する．　IVV1, IVV2

（３）インデックス配列の未処理の部分を記憶しておくスタック配列（１次元：先頭と最後，処理を再開する場所）を用意する．　ISTK1, ISTK2, ISTKA

（４）ベクトル化を行う部分を決定する．

（５）ベクトル化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．（変数名に VV を付ける）

（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ﾙｰﾌﾟ”化する．

（７）単純な条件分岐はそのままベクトル化する．

（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（図２）．

（９）前の処理が完了したのち，スタックの先頭から情報を取り出し，処理を中断した場所から再開する．

```
      IF(A .LE. 0.0) GO TO 200
                    ||
                    V
      J=IVV1
      K=0
CDIR$ IVDEP
      DO 120 I=IVV1,IVV2
      IF(A VV(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
      IF(J .EQ. IVV1) GO TO 200
      IF(K .GT. 0) THEN
        ASSIGN 200 TO KGOTO
        NSTK       =NSTK+1
        ISTK1(NSTK)=J
        ISTK2(NSTK)=IVV2
        ISTKA(NSTK)=KGOTO
        IVV2       =J-1
      ENDIF
```

図２．ベクトル化を行った条件分岐

表１．ベクトル化による高速化の効果

ﾓﾃﾞﾙ	要素数	ﾓｰﾄﾞ	MOSFETのみ		全体	
			秒	比率	秒	比率
1	15	ｽｶﾗｰ	34	1.11	41	1.08
		ﾍﾞｸﾄﾙ	31		38	
2	45	ｽｶﾗｰ	34	1.86	37	1.74
		ﾍﾞｸﾄﾙ	18		21	
3	150	ｽｶﾗｰ	34	2.42	36	2.25
		ﾍﾞｸﾄﾙ	14		16	
4	450	ｽｶﾗｰ	36	2.54	39	2.26
		ﾍﾞｸﾄﾙ	14		17	
5	1500	ｽｶﾗｰ	120	2.63	138	2.19
		ﾍﾞｸﾄﾙ	46		63	
6	1884	ｽｶﾗｰ	189	2.14	261	1.62
		ﾍﾞｸﾄﾙ	88		161	
7	3000	ｽｶﾗｰ	242	2.58	300	1.96
		ﾍﾞｸﾄﾙ	94		153	

適用した結果を表１に示す．ベクトル化をおこなった部分で２倍以上の高速化され，全体でも２倍程度高速化されている．

４．　高速化の限界

現在のスーパーコンピュータが仮定している条件を満足している問題はそのまま高速に計算することができるが，仮定を満足していない問題は高速に計算できるとは限らず，むしろ高速に計算できないことが多い．高速化するためにはプログラムを変更しなければならない．スーパーコンピュータでも負荷が大きい計算は高速化をおこなう価値があるが，負荷が大きくない計算は高速化をする必要はない．（時代の最先端を行くような結果を得るには，現在のスーパーコンピュータの能力の限界付近での計算も必要になるであろう．）

現在のスーパーコンピュータは機種ごとに特性が異なる．そのため，最高の性能を発揮させるためには，スーパーコンピュータごとに方法を変えなければならない．計算機ごとに高速化のやり方を変更するのは，ユーザにとって望ましいことではない．したがって，計算機ごとに異なる高速化の方法を適用するのではなく，すべてのスーパーコンピュータに共通の性質に着目して高速化を行うことが望ましい．現在のパ

イプライン方式のスーパーコンピュータに共通な性質としては，

（１）　ベクトル計算が速い

（２）　プロセッサに比べメモリが遅い

（３）　メモリへのアクセスは連続に行う方が速い

（４）　条件分岐が遅い

などが考えられる[4,5]．この方法では機種ごとに合った高速化の方法に比べると少し性能は落ちるが，どのベクトル型スーパーコンピュータでも高速化され，適用した手間に比べて効果が大きい．機種ごとに合った高速化が山の頂上を目指すものとすれば，共通の性質に対する高速化は山の８合目か９合目を目指すものと言える．今後，高速化のため各種の並列化されたスーパーコンピュータが出現するであろうが，パイプライン方式のスーパーコンピュータも引続き大型の数値計算の分野では重要な地位を占めると考えられる．そのため，パイプライン方式のスーパーコンピュータに共通の性質に対する高速化は使用する計算機が変更になってもプログラムを変更する必要がなくなり，ソフトウェアの蓄積が行える．

５．　おわりに

現在のスーパーコンピュータで十分にベクトル化されたプ

ログラムも，並列処理なしに現在の１００～１，０００倍以上の性能を出すことは困難である．今後の方向として，複数のプロセッサによる並列処理が不可欠である[6]．現在の計算機環境で考えると，プログラム変更に時間を必要とするベクトル化や並列処理よりも何もしないで済む速いスカラー計算機が望ましい．計算をすることが目的のユーザにとって，プログラムをベクトル化や並列化することに時間をかけることは望ましくない．しかし，モデル化まで戻って考えると，科学技術計算では並列計算可能な問題が多い[7]．逐次処理のプログラムは，人間が計算機に処理の手順を与えているのであるが，より抽象化されたレベルで何をすべきかということを計算機に示してやれば，ベクトル化・並列化もより容易にできる．例えば，２つの正方行列の積を計算する場合，３重のＤＯループ構造を与えるよりも行列定義と行列演算で記述すれば，使用している計算機ごとに能力を最大に発揮させることは容易になる．今後は，このような方面の研究が必要である[8,9]．

参考文献

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

[２] 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)

[３] 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)

[４] 村田，小国，唐木: スーパーコンピュータ，丸善，(1985)

[５] 物理学会編: スーパーコンピュータ，培風館 (1985)

[６] 星野力: ＰＡＸコンピュータ，オーム社 (1985)

[７] 川合敏雄: 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)

[８] 福井，佐々木，鈴木，佐藤: 数値・数式融合計算のためのＦＯＲＴＲＡＮとＲＥＤＵＣＥの一結合方式，日本ソフトウェア科学会第３回大会論文集，D-6-2 (1986)

[９] 梅谷 他: 数値シミュレーション用プログラムＤＥＱＳＯＬ，情報処理学会誌，Vol.26, No.1, pp.168-180 (1985)