数理解析研究所講究録
199巻 1974年 21-27

FORTRANの最適化と数値計算

京大 大型計算機センター
星野 聰

FORTRAN の最適化と数値計算のアルゴリズムの関連について若干の計算機利用者にたずねたところでは、あまり関係がないように思われる。すなわち、最初は最適化されていないプロセッサでテストを行うことは、デバッギングを進める上で便利であるが、そのプログラムが予想通りの動きをすると考えられる段階にくると、そのまゝ最適化を行うプロセッサを用いて処理をすゝめるという人が多い。もちろん、各メーカのFORTRAN プロセッサの仕様をくわしくみれば FORTRAN の最適化と、数値計算は何らかの関連はりつかもしれないが、それが本質的なものかどうかについては個々に検討しなければならない。

最適化は、数値計算のアルゴリズムというよりは、プログラムの書き方と計算速度に影響している様である。たとえば最適化の一つは、プログラム中の共通因子は、もしの流れ

/

ば、まとめて一度に評価することである。これによって、プログラムを読みやすく出来るので、デバッギングの効率を上げると共に、あとからのプログラムの保守も容易となる。しかも、プログラムの実行速度は落さずにすむのであるからこのような最適化は有益である。

計算によっては、いくら処理の部分が能率のよい様に苦労して書いたとしても、これを計画してから所要の計算が終るまでに要する期間が長ければ無駄なこともある。とくに急いで結果がほしいときには、プログラム自身は駄長でも最短時間で答さえ得たいと思うものである。このような場合、最適化されたプロセッサはプログラムの能率を改善する上で特に有効である。もちろん、最適化をするプロセッサでは、コンパイルに長い時間を要するから、実行に要する時間との和が最小になるように考えなくてはならないから、問題によっては、最適化をしないプロセッサで処理をした方がよいこともありうる。

プログラムの実行速度をきめる一つの因子は、添字付変数の処理である。最適化をしないプロセッサでは、二次元以上の添字付変数も一次元に直してプログラミングされているのを見かけることがある。これは乗算を用いなくなるので、処理速度を上げられるのをねらったものであろう。しかし、

この仕様では、2次元ないし3次元の添字付変数を含まれる計算のプログラムを書くことは極めて繁雑であるし、あとからみても理解するのが困難である。この点、2次元添字変数は2次元の表示法でプログラミングすると、きわめて読みやすく、能率のよいデバッギングが出来る。また、最適化をするプロセッサをつかうと、処理も高速に行われる。むしろ、1次元表示にもっていると、(最適化の程度にもよると思われるが)、インデクスレジスタなどのレジスタの使用が充分に行われないので、かえって計算速度が低下することがある。

これについては、かって京都大学大型計算機センター広報に書いたことがある(4)。すなわち、プログラムのある部分

```
      DO  10  J = 2,25
      A( 2*I-3, 2*J-3 ) = A( 2*I-1, 2*J-1 )
      A( 2*I-3, 2*J-2 ) = A( 2*I-1, 2*J )
      A( 2*I-2, 2*J-3 ) = A( 2*I , 2*J-1 )
10    A( 2*I-2, 2*J-2 ) = A( 2*I , 2*J )
```

を実行させたところ次の結果をえた。

最適化しないプロセッサによる処理時間 = 194.6 秒

最適化するプロセッサによる処理時間 = 13.5 秒。

ところで，このプログラムを

```
      DO  10   J = 2, 25
      I0 = 2 * I
      J0 = 2 * J
      I1 = 2 * I - 1
      J1 = 2 * J - 1
      I2 = 2 * I - 2
      J2 = 2 * J - 2
      I3 = 2 * I - 3
      J3 = 2 * J - 3
      A (I3, J3) = A (I1, J1)
      A (I3, J2) = A (I1, J0)
      A (I2, J3) = A (I0, J1)
10    A (I2, J2) = A (I0, J0)
```

と書き直してみると，添字付変数のアドレス計算を逐一行う最適化しない形のプロセッサにとっては，手数が省けるので

計算時間を短縮する効果があると考えられるが、最適化を行うプロセッサにとっては、添字がDOの制御変数ではないので不利となる。実際の実験結果によると

最適化しないプロセッサによる処理時間 ＝ 148.3秒
最適化するプロセッサによる処理時間 ＝ 96.1秒

となった。このように最適化の効果はほとんどなくなってしまうが、これは添字付変数のアドレス計算に最適化が行なわれなくなったためである。各添字付変数のアドレスは、このDOループを一巡するごとに一定数だけ変化することは、わかるはずであるが、プロセッサの最適化の程度によっては最適化が行われないのである。最近は、プログラムの流れを解析し、変数の変化量も調べ、また使用可能なレジスタ類を活用するとともに、無駄なload/storeを減らすプロセッサも実用化され、実行速度は、かなり向上している。(1)

しかし、最適化するのがむずかしいところも残っている。たとえば、短かいサブルーチンを実行中に呼ぶときには、それを呼ぶための手続き部分に時間がとられてしまう。このようなlinkageのために計算機は一日のうち何時間占有されているのだろうかと、Knuth(1)は述べている。

これをさけるには　サブルーチンをopen化するか，又はhand codingによって無駄な部分があれば除去するかが考えられるが　一般性を失わない形で，最適化プロセッサにとり入れるのには問題がある。

ある種の，最適化プロセッサでは，プログラムのある範囲でラベルがなく　代入文が沢山つづくようなときに，コンパイル時間が異常に長くかかることがある。プロセッサの作成者は　この現象にあまり注意しないのかもしれないが　かなりの時間がこのために費されてしまう恐れがある。　これは，コンパイル時に，共通因子をさがす際に，対象となる因子の個数が多くて時間がかかるためと考えられる。これを避けるには　プログラムの流れとは関係のないラベルを，あちらこちらに立てて見るとよく，コンパイル時間を減らすことができる。これも共通因子をさがす段階でhashing方式を使って探索のスピード化を計れば解決されるのではないだろうか。

最適化により，また計算機自身が高速化することによって計算の手続きが効果化でなくても，何らかの答が，あまり長くない時間内に得られるようになったことは，その研究上からは良いことである。しかし，その結果，より能率のよく誤差の少ない数値計算法の研究がなおざりにされるようなことがあれば，こまったことである。

数値計算の種類によっては、そのアルゴリズムの計算時間に大きく影響し、それが長大な時間を要求するとき、やはりその問題についての数値解析的な検討が必要である。

参考文献

(1) D.E.Knuth, An empirical study of FORTRAN programs, Software-Practice & Experience, Vol.1,105-133,(1971).

(2) J.Larmouth, Serious FORTRAN, Software-Practice & Experience, Vol.3,87-107,(1973).

(3) D.C.Hoaglin, An analysis of the loop optimization scores in Knuth's 'Empirical study of FORTRAN programs', Software-Practice & Experience, Vol.3,161-169,(1973).

(4) 星野, プログラミングノート(9) 京都大学大型計算機センター広報, vol.4, No.7, 15-17.

(5) 星野, プログラミングノート(14) 同上, vol.4, No.8, 19-24.

(6) 星野, プログラミングノート(15) 同上, vol.4, No.9, 5-8.