ハイパフォーマンス
コンピューティング 58－7
（１９９５．１０．１８）

間接アドレッシング処理演算の高速化とその応用

濱口信行

（株）日立製作所　ソフトウェア開発本部

分子科学計算および気象計算プログラムの性能評価を実施すると2つの問題点があった．
(1)プログラムの性能が，間接アドレッシング処理演算のメモリアクセスパターンに非常に依存する．
(2)FORTRAN標準関数以外の数学関数を使用している為，プログラムのポータビリティがない．

そこで，間接アドレッシング処理演算を解析し、ハードウェア，ソフトウェア双方の改善を行ない，補誤差関数，Teten's公式，指数関数に適用し、分子科学計算，および気象計算プログラムの精度，性能，ポータビリティの向上を達成した．

Accelaration and application of index addressing operation

Nobuyuki Hamaguchi

Software Developement Center , Hitachi Ltd.

When I had evaluated the performance of molecular scientific simulation and meteorological simulation programs, their existed two drawbacks.
(1) Performance of programs depended extremely on memory access pattern of index addressing operation.
(2) Using non-standard FORTRAN mathematical functions, Simulation programs are lacking for portability.

I had analyzed index addressing operation and improved in respect of hardware and software, and applied complement error function, Teten's formula, exponential function, and attained the elevation of accuracy, performance, and portability of molecular scientific simulation and meteorological simulation programs.

1. はじめに

ベクトル型スーパコンピュータにおける分子科学及び気象計算分野の技術計算プログラムの性能評価で，メモリ素子の変更による影響を調査した．

見積りでは，低速メモリでのCPU時間/高速メモリでのCPU時間は1．10程度と考えていたが，分子科学計算で1．47，気象計算で1．33という値となり、その解析を行なうと，分子科学計算では補誤差関数，気象計算では放射計算処理部分で時間が要している事が分かった．この2つの計算処理には，間接アドレッシング処理において，そのアドレス参照パターンがある点に集中しているという共通点があった．すなわち，A(I)=B(L(I))，ループ長Nにおいて，L(I)の異なる値の数をMとすると，M<<Nの場合である．

また，その解析の過程で，補誤差関数，立方根がFORTRAN非標準数学関数の為，ポータビリティの確保の必要性が生じた．立方根に関しては、ベキ乗計算に置き換える事で特に困難ではないが，補誤差関数はその精度，性能を考慮しなければならず，なんらかの対策が必要になった．

間接アドレッシング処理はリストベクトル命令を使用して行なっているが，これを用いた高速化手法は多くあったが，その性能がアドレス参照パターンに大きく左右されるという問題があり，ハードウェア的にはリストベクトル命令の性能がアドレス参照パターンに出来るだけ依存されない改善，ソフトウェア的にはハードウェアの改善を前提として，その高速化手法の適用範囲の拡大を検討した．

分子科学計算では，補誤差関数計算[1] の改善と、，ポータビリティの確保，気象計算では放射計算部分はハードウェアによる改善を行ない，実行時間の25％程度を占めている水の相変化（主に水蒸気に関する計算）[2] に関連した処理部分でよく現われるTeten's公式[3] 処理の改善を実施した．

2. 基本DOループの性能測定結果

リストベクトル命令の性能のアドレス参照パターン依存性を調べるために次の3つのパターンを設定した．

CASE1 : L1(I)=I　　通常の連続アクセスパターンをリストにしたケース．

CASE2 : L2(I)=定数　　1つの要素のみを参照するケース

CASE3 : L3(I)=MOD(4*I-4,10000)+4　　ストライド4のメモリアクセスをリストにしたケース．

その，ハードウェア改善前（S-3600/180）と改善後（S-3800/180）での性能測定結果を表1に示した．

表1　基本DOループの性能測定結果

（ループ長 = 10000 ，配列DX,DY,DZは倍精度）　　単位 : MWORDs/sec or Mflops

項番	DO ループ	S-3600/180(改善前)	S-3800/180(改善後)
1	DZ(I)=DX(I)	600	1012
2	DZ(I)=DX(L1(I))	348	704
3	DZ(I)=DX(L2(I))	13	731
4	DZ(I)=DX(L3(I))	228	623
5	DZ(I)=DX(I)+DY(I)	480	869
6	DZ(I)=DX(L1(I))+DY(L1(I))	300	639
7	DZ(I)=DX(L2(I))+DY(L2(I))	8	702
8	DZ(I)=DX(L3(I))+DY(L3(I))	156	627

（注） S-3800/180とS-3600/180のベクトル クロック比は1:2 (2 ns,4 ns)

表1から，ハードウェア改善前と改善後を比較すると次の事が言えます．

(1)リストベクトルを使用しない場合と使用した場合の性能差が縮まった．特に演算が入った場合が顕著．

(2)リストベクトルを使用した場合，データ参照パターンの差による性能差がなくなった．特に項番3，項番7の場合が顕著．

この要因としては，演算速度とメモリアクセス速度の比率が変わっている事やインターリーブ数の増加もあるが，最も大きいのはバンクコンフリクトの削減である．

概略的に述べると，図2の様に，複数のメモリアクセス要求を出す前にアドレスコンペアを行ない，同一アドレスに対するメモリアクセス要求は1つにし，その要求応答完了後同一アドレスを要求するものにコピーするという方式です．

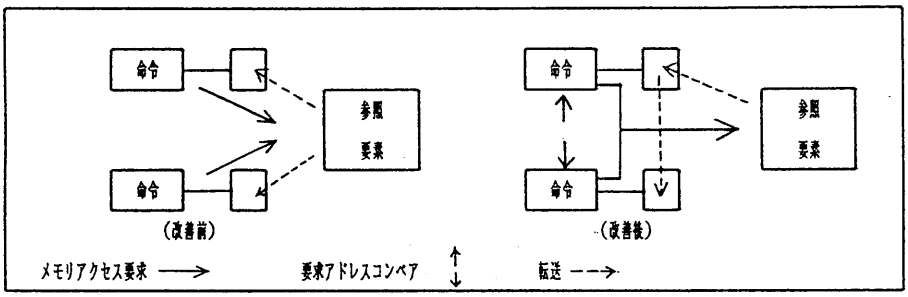

図2　メモリアクセス要求削減

この効果は改善後の項番3，7がそれぞれ項番2，4，項番6，8の性能を上回るところまで出ています．

以上によりリストベクトルの性能がデータ参照パターンに依存しなくなり，リストベクトルを用いた高速化を行なう場合，その命令数の増加にのみ留意すれば良い事になった．

3.ソフトウェアの改善

分子科学計算では引数0.1~6の補誤差関数，気象計算ではTeten's公式 q=6.11*10**(aT/(b+T))　a=7.5 b=237.2

Tの単位は℃ を変形して q=exp(19.0793-4098.026/(T-35.85)) : q 飽和水蒸気圧 (hPa)，T °K

すなわち、単精度指数関数計算が中心となり，今回はリストベクトルを使用したベクトル関数の改善について記述しました．

3.1 ベクトル関数の性能要因

ベクトル関数の性能は，近似式の項数と適用する近似式の数に依存します．

例えば，F1，F2，F3 : 異なる関数近似式で，それぞれの計算に要する命令数をN1，N2，N3の場合，次のDO ループでは

```
    DO 10 I=1,N
    IF(X(I).LE.-1.0D0) THEN
    Y(I)=F1(X(I))
    ELSE IF (X(I).LE.1.0D0) THEN
    Y(I)=F2(X(I))
    ELSE
    Y(I)=F3(X(I))
    END IF
 10 CONTINUE
```

ベクトル実行

ループ長Nのベクトル命令 N1+N2+N3 回を実行．

逐次実行

N×MIN(N1,N2,N3)~N×MAX(N1,N2,N3)

回の逐次命令を実行．

となります．この事から，ベクトル関数の性能を向上させるには，近似式の項数を減らす事よりもむしろ近似式の数を減らす方が効果があります．逆に逐次実行では区間分けをして，近似式の数を増やしてもそれぞれの近似式の項数を減らす方が効果があるため，ベクトル関数の場合，従来とは若干異なります．

3.2 指数関数

指数関数の値を一次補間式で算出する場合，その相対誤差E(x)はa≦x≦b，(a<b) b-a=s 0<s≦0.5 x=a+ts 0≦t≦1 の場合

$E(x)=\frac{e^{a}(1-t)+e^{b}t-e^{x}}{e^{x}}=(1-t)e^{a-x}+te^{b-x}-1=(1-t)e^{-st}+te^{s-st}-1\geq 0$ となり

$t=\frac{e^{s}-s-1}{s(e^{s}-1)}$ で最大値を取ります．そしてその最大値は，a，bの値ではなく，区間幅b-a=sのみの関数となります．また $\frac{e^{s}-s-1}{s(e^{s}-1)}\fallingdotseq 0.5$，

$E(0.5)=\frac{1}{2}e^{-\frac{s}{2}}+\frac{1}{2}e^{\frac{s}{2}}-1\fallingdotseq s^{2}/8$の為，精度に応じて区間幅を設定する事が出来ます．

区間幅$s=2^{-n}$ (n=1,...,12)まで計算した結果は表3のようになります．

表3 区間幅と最大相対誤差

n	E'(t)=0となるt	E(t)の最大値	$s^2/8$
1	0.459	3.16×10^{-2}	3.12×10^{-2}
2	0.479	7.84×10^{-3}	7.81×10^{-3}
3	0.490	1.95×10^{-3}	1.95×10^{-3}
4	0.497	4.88×10^{-4}	4.88×10^{-4}
5	0.499	1.22×10^{-4}	1.22×10^{-4}
6	0.499	3.05×10^{-5}	3.05×10^{-5}
7	0.500	7.63×10^{-6}	7.63×10^{-6}
8	0.500	1.91×10^{-6}	1.91×10^{-6}
9	0.500	4.77×10^{-7}	4.77×10^{-7}
10	0.500	1.19×10^{-7}	1.19×10^{-7}
11	0.500	2.98×10^{-8}	2.98×10^{-8}
12	0.500	7.45×10^{-9}	7.45×10^{-9}

単精度指数関数では$n=12$，すなわち区間幅1/4096にとれば十分な精度が得られます．倍精度の場合，一次補間式では現実的に不可能で，

の部分をテーラー展開します．引数の区間への畳み込みは$x=y+n*\log_e 2$(2進)，$x=y+n*\log_e 16$(16進)で $e^{x}=e^{y}*2^{n}$，$e^{x}=e^{y}*16^{n}$

と指数部調整で計算が行なえます．

気象計算での単精度指数関数計算の精度と性能は表4，5の様になっています．

表4 単精度指数関数の精度と性能(10^6点)

(-180.218≦x≦174.673)

方式	平均相対誤差	最大相対誤差	S-3800/180 1000回実行
従来方式	0.154×10^{-6}	0.952×10^{-6}	6.935 秒
新方式	0.768×19^{-7}	0.476×10^{-6}	4.754 秒

表5 Teten's公式の精度と性能(10^6点)

(123.15≦T≦373.15)

方式	平均相対誤差	最大相対誤差	S-3800/180 1000回実行
従来方式	0.167×10^{-6}	0.952×10^{-6}	7.126 秒
新方式	0.166×10^{-6}	0.952×10^{-6}	5.320 秒

今後，倍精度化，IEEE化が考えられますので，引数範囲 0≦x≦1 (表6)， -170≦x≦170 (表7) 10^6点，M-880H での1回実行した時の時間を調査したところ，良い結果がでており，IEEE化のベクトル計算機(ベクトルパラレル計算機)でどの様になるかが今後の検討課題となっています．

表6 倍精度指数関数の精度と性能

方式	平均相対誤差	最大相対誤差	実行時間
従来方式	0.543×10^{-16}	0.222×10^{-15}	0.667 秒
新方式	0.469×10^{-16}	0.217×10^{-15}	0.488 秒

表7 倍精度指数関数の精度と性能

方式	平均相対誤差	最大相対誤差	実行時間
従来方式	0.261×10^{-16}	0.222×10^{-15}	0.666 秒
新方式	0.230×10^{-16}	0.220×10^{-15}	0.561 秒

3.3 補誤差関数

補誤差関数はFORTRANの標準関数ではありませんが，数学関数としてベクトル関数ルーチンを提供しています．補誤差関数 $f(x)=erfc(x)=\frac{2}{\sqrt{\pi}}\int_x^\infty e^{-t^2}dt$ は，x=-∞ で2，x=0で1，x=∞で0である事から，x>0でxの値が大きくなるほど近似式の項数は多くなり，他の関数（指数関数等）と同じ様に区間分けをして実行しようとすると，近似式の数が多くなり実行するベクトル命令の数が非常に多くなります．その為，定数テーブル $\{a_{ij}\}$ を持ち（必要ならば $a_{ij}=0$ としておく）

{ X(I) }から{ 引数の範囲 J } の対応配列を生成して $Y(I)=a_{j0}+a_{j1}*X(I)+\cdots+a_{jm}*X(I)^m\cdots$ (1)(X(I)の入る範囲 J)

とリストベクトル命令を使用して1つの近似式で計算します．ハードウェアの改善により性能はかなり上がりましたが，一般の数学関数の計算のパターン

$Y(I)=b_0+b_1*X(I)+\cdots+b_m*X(I)^m\cdots$ (2)と比較した場合，必要なロード命令の数が (1):(2)=m+2:1となり，まだソフトウェア的に改善の余地があります．またFORTRANの非標準関数であるため，ポータビリティがありません．

そこで補誤差関数の微分項の形を見ますと，$f'(x)=-\frac{2}{\sqrt{\pi}}e^{-x^2}$，$f''(x)=-\frac{2}{\sqrt{\pi}}e^{-x^2}(-2x)$，$f'''(x)=-\frac{2}{\sqrt{\pi}}e^{-x^2}(4x^2-2)$

$f^{(4)}(x)=-\frac{2}{\sqrt{\pi}}e^{-x^2}(-8x^3+12x)$，$f^{(5)}(x)=-\frac{2}{\sqrt{\pi}}e^{-x^2}(16x^4-48x^2+12)$，$f^{(6)}(x)=-\frac{2}{\sqrt{\pi}}e^{-x^2}(-32x^5+160x^3-120x)$

と $-\frac{2}{\sqrt{\pi}}e^{-x^2}$ x多項式の形になりますので，補誤差関数のテーブルと e^{-x^2} のテーブルを用意すれば，5~6次の多項式の計算と2度のテーブルサーチで計算出来ます．

また補誤差関数のテーブルと e^{-x^2} のテーブルをテキスト形式で保存すれば，プログラムのポータビリティは確保出来ます．

分子科学計算では，補誤差関数の引数の範囲が0.1~6.0であった為，区間幅を1/4096とすると，4次微分項までとると，x=6で相対誤差≒0.17×10^{-14} となり十分な精度が得られます．引数範囲を拡げる為に，5次微分項までとると，区間幅=1/4096，x=13で相対誤差≒0.89×10^{-16} となります．

また，IEEE形式の場合を考慮すると，引数はx≒26.6まで検討を要するが（$EXP(-26.6^2)\fallingdotseq10^{-308}$），5次微分項までとると区間幅=1/4096 x=26.6で相対誤差≒1.24×10^{-16} と十分な精度が得られます．

その精度と性能は，表7，表8の様になり，テーブル値を保存する事により，精度，性能を落す事なく，逆に向上させてポータビリティを確保する事が出来ました．

表7　補誤差関数4次微分項まで取った場合の精度と性能

（引数範囲 $0\leq x\leq6$ ： 10^6点）

方式	平均相対誤差	最大相対誤差	S-3800/180 1000回実行
従来方式	0.547×10^{-15}	0.342×10^{-14}	54.088 秒
新方式	0.547×10^{-15}	0.162×10^{-14}	9.048 秒

表8　補誤差関数5次微分項まで取った場合の精度

（引数範囲 $-13.2\leq x\leq13.2$ ： 10^6点）

方式	平均相対誤差	最大相対誤差
従来方式	0.386×10^{-15}	0.339×10^{-14}
新方式	0.386×10^{-15}	0.433×10^{-15}

4. 分子科学計算，気象計算プログラムによる実測結果

4.1 分子科学計算プログラムの実測結果

3つの異なる規模のプログラムを使用して，S-3800/180でのメモリ素子の影響，チューニングの効果を測定した．

表9 分子科学計算プログラムによるメモリ素子の影響

プログラム規模	プログラム	メモリ素子A		メモリ素子B		性能比(1/2)
		CPU時間	VPU時間(1)	CPU時間	VPU時間(2)	
小	オリジナル	13.06 秒	11.94 秒	11.31 秒	10.13 秒	1.18
	チューニング	8.84 秒	7.46 秒	8.77 秒	7.32 秒	1.02
大	オリジナル	446.10 秒	406.95 秒	428.23 秒	355.25 秒	1.15
	チューニング	261.31 秒	221.66 秒	263.71 秒	221.38 秒	1.00

バンクコンフリクトの影響はVPU時間の差に出ますが，チューニングによりその影響が出なくなっています．

表10 分子科学計算プログラムによるチューニング効果(メモリ素子B)

プログラム規模	オリジナル		チューニング		性能比(1/2)
	CPU時間(1)	VPU時間	CPU時間(2)	VPU時間	
小	11.31 秒	10.13 秒	8.77 秒	7.32 秒	1.29
中	55.14 秒	46.83 秒	39.98 秒	31.41 秒	1.38
大	11.31 秒	428.23 秒	355.25 秒	263.71 秒	1.62

プログラム規模が大きくなるほど性能向上比が増加している事が分かり，このチューニングの有効性が出ています．

4.2 気象計算プログラムの実測結果

気象計算プログラムを使用して，Teten's 公式の計算部分をテーラー展開のチューニングと一次補間式による効果を比較した．また，他の一般的チューニングとの比較の為，水蒸気関連部分とほぼ同じ計算量を持つ部分に施したチューニング効果を測定した．(S-3800/480)

表11 水蒸気計算処理に対するチューニング効果

プログラム	経過時間	性能向上比
オリジナル	162 秒	1
テーラー展開	138 秒	1.17
一次補間式	127 秒	1.28

表12 各種チューニングによる短縮時間

チューニング処理内容	短縮時間
行列積の十段十列化	25 秒
アンローリング	10 秒
整数除算/剰余の高速化	7 秒

現在の数学関数ルーチンの指数関数はかなり高速化されていますが，リストベクトルを使用し，一次補間式で計算すると，その短縮時間は35秒(性能向上比1.28)となり他の一般的なチューニング方法よりも，効果が出ています．

5. おわりに

技術計算分野のプログラムでよく出現する間接アドレッシング処理を検討し，その改善をハードウェア，ソフトウェア両面から，実施し，基本演算，数学関数を使用したプログラムの計算の精度，性能の向上を達成できた．

またFORTRAN非標準数学関数である補誤差関数をあらかじめ計算しておいた数値を入力する事により，プログラムのポータビリティを精度，性能を落すことなく逆に，向上させて，確保することができた．

今後は，より汎用性を広げて多くの計算機アーキテクチャーに対しての精度，性能の向上を検討，実施していく予定です．

謝辞 ： 性能評価に当たり，分子科学計算プログラムにおいて，御指導，御助言していただいた御茶ノ水女子大 長嶋助教授，ハードウェア処理検討で御協力いただいた(株) 日立製作所 汎用事業部 林原氏に感謝申しあげます．

参考文献 ：

(1) 電子計算機のための数値計算法 III　　一松信，字野利雄，山内二郎 共編

(2) 気象力学　　小倉義光 著

(3) 天気予報の技術　　新田尚，立平良三，市橋英輔 共編