Vol.2010-ARC-192 No.17
Vol.2010-HPC-128 No.17
2010/12/17

HPC用に欲しい数値演算ハードウェア機構

村　上　　弘[†1]

近未来の HPC 計算機に欲しいと思うハードウェア数値演算機構には例えば以下のようなものがある．1) 3 倍精度，4 倍精度など高精度の数値演算命令．2) 丸め指定付き数値演算命令．3) 数値を指定精度に丸める命令．4) 区間演算を行う命令．5) 小規模行列の積などの演算を高速に行う機構や命令．6) 積和演算での丸め誤差や演算の順序の違いによる結果の相違を防ぐための長いアキュムレータ．7) 多倍長演算の実現を支援する機構．

Hardware Mechanisms for Numerical Arithmetics Desirable for High-Performance Computations

Hiroshi Murakami[†1]

The following hardware mechanisms of numerical arithmetics are desirable for high-performance computing systems of the near future. 1) Triple or quadruple precision arithmetic operation instructions. 2) Arithmetic operation instructions with the specified rounding modes. 3) Instruction to round a number to the specified precision. 4) Instructions for the interval number arithmetics. 5) Mechanisms and operation instructions which enable fast computations of small size matrices such as multiplications. 6) The long accumulator to reduce rounding error effects during the vector inner product, or to prevent the change of results by the change of order of additions or subtractions. 7) Mechanisms to support the implementation of multiprecision arithmetics.

†1 首都大学東京 数理情報科学専攻
Department of Mathematics and Information Sciences, Tokyo Metropolitan University

1. はじめに

高性能計算機システムは 10 年間で約千倍のような性能向上がこれまで数十年に渡り続いてきた．演算の高速化と記憶容量の増大に伴い，扱える問題規模が拡大してきた．そのため，現在と比べてはるかに計算速度が低速で，規模の小さい問題しか扱えなかった時代にはある程度十分だと思われていた倍精度による演算は，近い将来には計算機の演算速度と扱う問題の規模や用いる計算の方法によっては，精度に余裕がないかまたは不足している可能性が考えられる．

データに施す演算回数を増すと数値誤差も増すが，そのほかにも方程式系の規模が大きくなると一般には条件が悪化するので，もしもこれまでは倍精度演算（IEEE754, 2 進 64bit）で十分であったとしても，同じ方法を使い続けると将来の大規模問題を解くには不十分となる可能性がある．もちろん，問題の大規模化が計算結果に与える影響の程度は問題自身の性質や採用した解法や計算の具体的な実装などにも依存する．

昔の富士通製の大型計算機には 3 倍精度演算があって，FORTRN 言語から使えたそうであるが，これはおそらくソフトウェアにより実現されていたのであろう．また，東京大学大型計算機センタに昔に設置されていた HITACH-8800 には，撤去前の最後の 1 年間程度の期間，4 倍精度のハードウェア演算機構が試験的に付加されていたことを当時の東京大学計算機センターニュースの記事で読んだ記憶がある．現在その資料は手元にはない（加減算と乗算の機能を持ち，除算はソフトウェアで実現したらしい）．

2. 高精度の演算機構

ある算法が必要とする演算精度の桁数が，解きたい問題の規模拡大や演算回数の増加に伴なって緩やかに増えるのであれば，倍精度では不十分となっても，当分の間は 3 倍精度あるいは 4 倍精度で十分となるようなことも多いかと思われる．しかも，必ずしも数値と演算のすべてを 4 倍精度にしなくても良いことが多い．

現在，4 倍精度演算はソフトウェアにより実現されている場合が多く，Fortran などの高級言語から利用可能な形で提供されている．また，より高精度の多倍長精度演算もソフトウェアにより実現されている．しかし，4 倍精度演算は倍精度の場合と異なり，ソフトウェアにより実現されているので演算が遅い．例えば現在の Intel Core アーキテクチャCPU と Intel Fortran の組み合わせでも演算速度が数十倍程度も遅く，GMP などのソフトウェアによる多倍長演算ではさらに遅くなる．もしも 4 倍精度演算がハードウェアで理想的に実

© 2010 Information Processing Society of Japan

Vol.2010-ARC-192 No.17
Vol.2010-HPC-128 No.17
2010/12/17

装されたならば，倍精度に対する速度低下は数倍の程度でおさまるであろう．

4 倍精度は，いまだに Fortran 言語の標準規格には含められておらず，またソフトウェアによる演算が遅いためであろう，Lapack など代表的な数値演算ライブラリにも 4 倍精度を用いるルーチンは現状では含まれていないのが普通である．4 倍精度の機能があってもソフトウェアによる 4 倍精度演算の実行はとても遅いので，ライブラリやアプリケーションは 4 倍精度演算を採用しないので，あまり利用されない機能は重要であるとはみなされないのでハードウェアによる実装で性能を追求する必要性を認められず，ソフトウェアにより実現した形でだけ提供される．すると機能があってもソフトウェアによる実現での実行はとても遅いので，ライブラリやアプリケーションからは使われない．のような悪循環の構造になっているのではないかと思われる．もしもハードウェアにより実現された倍精度と同等か半分程度の速度で動作する 4 倍精度演算機能があれば，そのようなマシン上でプログラムを作成したり，4 倍精度をうまく使うための研究にも実用への期待感が出てくることになる．

四則演算等の精度を実質的に高めた計算は，ソフトウェア的に特別な工夫をすることで可能であるが，問題点としてはプログラムコードが複雑化するほか，条件判断による分岐が増える，記憶へのアクセス回数が増え，さらに演算順序の依存関係があるので性能を出す上での難点がある．そのような特別な工夫は，高精度演算をハードウェアで行えない状況での必要上の工夫や知恵だといえる．

3. 丸め方式の指定付きの演算命令

レジスタ間の四則と開平の演算の命令フォーマット中には，演算精度（S,D,T,Q）の指定のほかに，演算結果の丸め方式の指定 RN(round-nearest)，RU(round-up), RD(round-down), RZ(round-to-zero) をも含める（または命令にプレフィックスを付ける）と，演算器の丸めモードを制御するフラグを設定するのと違って各演算をできるだけ独立平行に処理できるので良いのではないだろうか．

4. 数値の仮数部を指定した長さに丸める命令

数値の仮数部を指定した長さに丸める命令があれば，算法が用いている変数や配列の数値の精度の変化が算法の最終結果にどれだけ影響を与えるかを調べるなどの目的に使えるであろう．

5. 区間演算を行う命令

区間演算（四則演算とできれば開平も含めて）が普通の演算と同様の速度でハードウェア命令として実現できれば，区間演算を用いる算法とその応用の研究には有用であろう．ハードウェア命令かマイクロ命令を組み合わせた命令で実現すれば，条件分岐によるパイプラインの乱れを起こさずに演算を進められよう．区間の両端の数値には倍精度以外に 4 倍精度も可能とすると有用性が高まると思われる．ハードウェアだけによる命令の実現だけではなくて，プログラム言語からの区間数と区間演算の利用のサポートと，区間演算を用いた基本計算ライブラリの開発も強く望まれる．

6. 小規模行列の積などの演算を高速に行う機構

比較的小規模（16x16 とか 32x32 などの）の行列計算（行列積，行列転置，行列ロードストア）を行うための行列用レジスタと演算回路を（可能なら 4 組とか 8 組を）持たせることができれば，線形計算において有用であろうし，LINPACK ベンチマークの性能向上のためにも有用であろう．

7. 積和演算での長いアキュムレータ

大規模問題の解法の中核として用いられることが多い線形計算では，(直接解法でも反復法でも）内積計算が極めて重要である．内積演算では丸め誤差の累積の他に，相殺による桁落ちで精度が低下するが，良く知られている対策として加減算器内部のアキュムレータに保護桁（ガードディジット）を加えて内部精度を高めて影響を減らすか，演算器とレジスタ上の精度をメモリ上の精度よりも高くすることが，ある程度有効である．但し，数値をメモリに置くかレジスタに置くかの場所により表現可能精度が変わるので，その変化により計算結果が同一性ではなくなる．

積和の演算あるいは和の計算では，非常に長い仮数部を持つ特殊なレジスタを用意すると，途中でオーバーフローやアンダーフローが生じない限り，演算結果は正確で、加減算の演算順序によらない結果を求めることが可能であることが示されている[13)]．資源的には贅沢だが，加減算の順序を変更しても同じ結果を与える性質は，縮約（reduction）演算を並列計算で行う場合に，計算結果が演算の実行順序によらずに同じとなる利点を持つと考えられる．演算の順序を緻密に規定しない普通の並列計算のプログラムでは，実行のたびごとに演算の順序が変わり，結果も変わりうるので，プログラムのデバッグや計算結果の妥当性は

© 2010 Information Processing Society of Japan

Vol.2010-ARC-192 No.17
Vol.2010-HPC-128 No.17
2010/12/17

少数回のテストでは確認判断しにくい．

8. 議　　論

もしも 4 倍精度の演算が使われるようになって，誤差の影響が弱まり目立たなくなると，数値計算の誤差解析や数値誤差に強い計算法を開発する研究の需要や重要性が減るように考えられるかも知れないが，実際にはそういうことにはならないであろうと思われる．まず，数値誤差解析や誤差に強い計算法の開発は，有限精度で近似計算をする前提がある限りは不要にはならない．そうして解析の結果，倍精度演算による素直な実装で十分と保証された部分には，計算速度や半導体回路のエネルギ消費の観点からも 4 倍精度を使わずに倍精度を使えば良いし，倍精度では不十分な可能性があるが，4 倍精度では十分であることが示されたならば 4 倍精度を用いれば良い．4 倍精度でも不十分な可能性があれば，さらに高精度の演算を用いるべきであるなどとなる．あるいはとりあえずある精度で解いてみて，得られた結果の検証のためにより高い精度で計算をするというアプローチになるかと思われる．

数値計算は，過去の算法の丸め誤差解析や誤差の影響の低減化法を研究する分野に限定されているわけではないので，従来の算法を改良したり変形したり組み合わせたり，近似解法や新しい算法を発明したり発見すること，計算量の低減化以外にも記憶量，エネルギ消費，通信量などを低減化したり，算法の並列化や並列性の向上，同期頻度の低減化など，誤差解析以外にもやるべき課題はたくさんある．

精度の高い数値を扱うことで，長大な計算過程の中途で数値的破綻が生じる確率が減らせたり，結果の数値の信頼性が向上しうるだろう．数値の高精度化はデータの量を増やすが，もしも算法の安定のための余分な計算や解を改良するための反復計算を省略できるならば，計算全体としては記憶の参照や移動の量が減らせる可能性もあるほかに，数値の有効精度が高いことを積極的に用いて加速法などを利用すれば求解のための反復の収束を速くできる可能性もある．

ハードウェアの良い実装により，4 倍精度演算を従来の倍精度演算とそれほど違わない例えば半分から数分の 1 程度の速度で行えるようになったとすれば，その機能を用いて今よりもずっと丈夫な算法やアプリケーションを開発できる可能性があるだろう．また，これまでは精度的な面で実用にならないとされてきた高速算法などが実用となる可能性もあるだろう．もちろん新しいハードウェアの機構を生かしきるためのソフトウェアや算法の開発は，ハードウェアができた後の長期に渡る安定した開発作業や基礎研究の蓄積が必要となる．そのような投資がなされなければ，新たに加えられた機能はあまり使われずに終わる可能性もある．少なくとも新しい機能を生かすには，対応する事項を言語の標準規格に追加して，コンパイラ処理系がそれをサポートし，基本的な数値ライブラリが作られて提供されて，OS も支援する必要があろう．

参 考 文 献

1) 一松信, 戸川隼人 編:数値計算における誤差, 共立出版, 1975.
2) 高木直史:算術演算の *VLSI* アルゴリズム, コロナ社, 2005.
3) 大石進一:数値計算, 裳華房，1999.
4) 大石進一:精度保証付き数値計算, コロナ社，2000.
5) 大石進一:*Linux* 数値計算ツール, コロナ社，2000.
6) Wilkinson,J.H. : *Rounding Errors in Algebraic Processes*, Prentice-Hall, 1964.
（邦訳は）一松信, 四条忠雄 共訳:「丸め誤差解析」, 培風館,1974.
7) Kulisch,U. W. and Miranker,W. L.: The Arithmetic of the Digital Computer: A New Approach, *SIAM Rev.*, bf 28, No.1(1986).
8) Golub,G. and van Loan,C.: *Matrix Computations*, 3rd Ed., John Hopkins Univ.Press, 1996.
9) Chatelin,C.F. and Frayssé,V.: *Lectures on Finite Precision Computations*, SIAM, Philadelphia, 1996.
10) Higham,N. J.: *Accuracy and Stability of Numerical Algorithms*, SIAM, Philadelphia, 1996.
11) Hansen,P. C.: *Rank-Deficient and Discrete Ill-Posed Problems*, SIAM, Philadelphia, 1998.
12) Overton,M.L.: *Numerical Computing with IEEE Floating Point Arithmetic*, SIAM, Philadelphia, 2001.
13) Kulisch,U. W.: *Advanced Arithmetic for the Digital Computer*, Springer-Verlag, Wien, 2002.
14) Muller,J. M.: *Elementary Functions – Algorithms and Implementation*, 2nd Ed., Birkhäuser, Boston, 2006.
15) ヘネシー, パターソン: コンピュータアーキテクチャ, 第 4 版, 翔泳社, 2008.
16) Kornerup,P. and Matula,D.W.: *Finite Precision Number Systems and Arithmetic*, Cambridge Univ.Press, Cambridge, 2010.

© 2010 Information Processing Society of Japan