多倍長浮動小数点演算を並列化対象とする自動並列化コンパイラ

4U－1

児島 彰　藤野 清次
広島市立大学 情報科学部 情報工学科

1 はじめに

高次方程式の求解などでは、高い精度の演算処理が必要である。このような処理では、FortranやCなど既存の言語処理系に用意されている数値型よりも、はるかに高精度な演算が必要になる。計算機自体のハードウェアで用意された浮動小数点演算以上の高精度な演算を処理するためには、図1のように長い配列で数値データを表現した多倍長浮動小数点演算による処理を行う必要がある。ソフトウェアで実現された多倍長浮動小数点演算の処理は多くの時間がかかる。

2 これまでの多倍長浮動小数点演算の方式

これまでに、多倍長浮動小数点演算を利用するために、以下のような方式が提案され、実現されてきた。

- ライブラリ方式 [2]
  多倍長浮動小数点演算をFortranやCなど既存言語のライブラリ関数として実現する。ライブラリの呼び出しはCALL文などの形式で書く必要がある。実現が比較的簡単であるが、ユーザの負担が大きい。

- プリプロセッサ方式 [1]
  既存言語を拡張し、特別な多倍長浮動小数点数値のデータ型を新たに定義し、このデータ型の演算を通常の数値の演算と同じような式で書くことを可能にする。新しいデータ型に対する演算はプリプロセッサで処理し、多倍長浮動小数点演算ライブラリ関数を呼び出すプログラムに変換する。

- C++クラスライブラリ方式 [3][5]
  C++を使い、多倍長浮動小数点数のクラスを定義し、多倍長浮動小数点演算はクラスのメンバ関数として実現する。加減乗除などには関数のオバーロード機能を使い、このクラスの演算を通常の数値型の演算と同じように式で書くことを可能にする。

A(1)	A(2)	A(3)	A(4)	.....	A(N)
符号 1	指数 2	仮数 3	仮数 4	....	仮数 N

図 1: 多倍長浮動小数点数の数値表現

プリプロセッサ方式やC++クラスライブラリ方式を使うと、多倍長浮動小数点演算は、加減乗除など基本的な演算は、通常の数値データと同じように簡単な式による記述が可能になる。この結果、プログラムの可読性は高いものとなり、ユーザの負担を軽減できる。

3 多倍長浮動小数点演算の並列化

一般に多倍長浮動小数点演算には多くの時間がかかるので、高速化のため並列処理することが考えられる。多倍長精度の演算を含むプログラムでは、乗算など一つの演算を一つのプロセッサに割り当てる方法で並列化することができる。通常の4倍精度などの数値の場合では、演算時間が短く粒度が小さいので、この方法では通信コストの影響が大きく、十分な加速が得られない。しかし、多倍長浮動小数点演算の場合は、一つの演算時間が長く粒度が大きくなるので、並列化によって良い加速が得られる可能性が多くある。

多倍長浮動小数点演算のライブラリ関数内部にも並列化の可能性がある。桁数が大きい乗算には、FFTを用いる方法が有効であることが知られており、FFTの処理も並列化の対象にすることができる。桁数が大きいときには、並列化で高速化が期待できるが、桁数が小さいときは、通信がボトルネックになりやすい。

多倍長演算には並列化で速度向上する可能性があるが、プリプロセッサ方式で生成された関数呼び出しを含むコードや、C++のクラスのメンバ関数の呼び出しを含むコードは、コンパイラにとっては依存関係が複

A parallelizing compiler for multiple-precision number
KOJIMA Akira, FUJINO Seiji
Department of Computer Engineering, Faculty of Information Sciences, Hiroshima City University, 3-4-1, Ozuka-Higashi, Asaminami-ku, Hiroshima City,731-3194,Japan

雑で、これまでの自動並列化コンパイラでは並列化しにくいものである。現在のところでは、関数呼び出しやC++のクラスのメンバ関数を効率よく並列化することができる自動並列化コンパイラはほどんどない。

4 多倍長演算を対象にした並列化コンパイラ

現在、多倍長演算を並列化対象として直接認識する並列化コンパイラを開発中である。このコンパイラは、プリプロセッサ方式と同様に、既存言語を拡張し、多倍長浮動小数点演算を言語レベルでサポートする。そして、より上位の構文レベルで多倍長浮動小数点演算を把握することで、依存関係などの解析を容易にし、並列化の対象として捉え易くする。これにより多くの並列処理が可能となり処理速度が向上が期待できる。コンパイラの構造は以下のようになっている。

- 構文解析
  通常の構文解析とほぼ同等の処理を行う。多倍長演算も分解せずに通常の演算と同等に扱う。

- 依存解析
  他の自動並列化コンパイラと同様に、演算の順序などによる依存関係を解析する。

- 並列化
  依存関係を保持し、粒度が確保される演算を、並列化の対象に選び、並列化する。多倍長演算をそのまま残すので粒度は確保し易い。

- コード生成
  コード生成では多倍長演算ライブラリを含む、MPIライブラリによる並列化コードを生成する。

5 並列計算機での予備実験

並列計算機で並列化コンパイラによる並列化を想定して予備実験を行った。多倍長演算の例として、Chebyshevの積分公式の分点を求める方程式の係数を取り上げた。方程式が高次になる場合、この係数を得るには高精度な演算が必要である。分散型並列計算機IBM SP2で、MPIライブラリを使い、並列化コンパイラが出力するコードを想定した並列プログラムを作成した。多倍長演算ライブラリとして、平山氏によるFortran版のMPPACK[4]を利用した。取り上げた問題を処理するプログラムで、計算時間を支配するのは、

$$\sum_{j=1}^{k} \frac{1}{2j+1} a_{2(k-j)}$$

のような総和処理を行う部分である。この部分の計算を簡単なブロック分割にして、各プロセッサに割り当てて並列処理させた。次数が大きくなるとより高い精度が必要になる。一例の実験結果を表1に示す。この分割は、4倍精度では演算時間が短く粒度が小さいので、通信コストのために十分加速が得られない。しかし、桁数の大きい多倍長精度の演算では、一つの演算自体の時間が長くなり、粒度が大きな並列化になり、相対的に通信コストの割合は小さくなる。そのため、次数の高い場合の大きな桁数では、よりよい加速が得られている。

表1: 各次数における実行時間(4プロセッサ, 単位:秒)

次数	桁数	並列化前	並列化後(加速)
512	125	1.96	0.77(2.5倍)
1024	230	14.11	4.00(3.5倍)
2048	455	137.11	36.95(3.7倍)
4096	905	1582.12	400.41(4.0倍)

6 おわりに

高い精度の演算ための多倍長浮動小数点演算は多くの時間がかかるが、これまでに提供された多倍長演算パッケージで作成したプログラムは、自動並列化コンパイラを使って並列処理させるのは難しかった。この多倍長浮動小数点演算を並列化対象として直接認識する並列化コンパイラを実現することで、多くの並列処理が可能となり処理速度が向上が期待できる。現在、この方式の並列化コンパイラを開発中である。

参考文献

[1] Watt,W.T., Lozier,P.W. and Orser, P.J.: "A Portable Extended Precision Arithmetic Package and Library with Fortran Precompiler", ACM Trans. Math. Soft. 2 pp.209-231 (1976)

[2] Brent,R.P.: "A Fortran Multiple-Precision Arithmetic Package", ACM Trans. Math. Soft. Vol.4 No.1 pp.57-70 (Mar. 1978)

[3] 平山 弘: "C++言語による高精度計算パッケージの開発", 日本応用数理学会 Vol.5 No.3 pp.123-134 (Sep. 1995)

[4] 平山 弘: "高精度計算プログラム MPPACK(Fortran版)", 添付マニュアル (1997)

[5] 平山 弘: "多倍長演算パッケージ MPPACK の高速化", 情報処理学会研究報告 98-ARC-128 Vol.98 No.18 pp.1-6 (Mar. 1998)