A-018

# Javaによる精度保証付き並列計算
# Parallel Computing Guaranteed Accuracy using Java

古賀 雅伸 † 常岡 聡子 † 廣木 聡憲 †
Masanobu Koga Satoko Tsuneoka Akinori Hiroki

## 1. はじめに

近年、精度保証付き数値計算を CPU の丸めモードを利用し高速に実現する方法が提案された [1, 2]。しかし、この方法をマルチタスク OS で使用すると、計算中に他のプロセスによって CPU の丸めモードが変更されてしまう可能性がある。また、上下限の丸めの計算を行なうため、通常の約二倍以上の計算時間が必要となる。そこで、プロセッサ集合 [3] を用いることで CPU の丸めモードを一定に保ち、精度を確実に保証する方法が提案された [4]。

文献 [4] では、マルチプロセッサマシン上でマルチプロセスにより並列処理をすることにより計算を高速に行なう方法が提案されている。しかし、この方法を実現するためには、CPU を 3 個以上持つマルチプロセッサマシンが必要となるが、3 個以上の CPU をもつコンピュータは非常に高額であり、種類も少なく一般のユーザは使用することが出来ないためコストがかかる。また、1 台の計算機に搭載できる CPU の数には限界があるので、高速化には、限界が生じる。

そこで本研究では、低コストで精度保証を確実に高速に実現するために、Java 言語を使用し、RMI 分散オブジェクトを利用することによって複数のデュアルプロセッマシンを用いた並列処理の方法を提案する。

## 2. Javaによる精度保証付き数値計算

### 2.1 丸めの指定

実数 $r$ は、計算機では浮動小数点数に丸められ保存される。丸めには**上向きの丸め**、**下向きの丸め** などがある。浮動小数点数を用いた数値計算では、真の解そのものを計算することは一般には不可能となる。そこで、真の解の下限を与える浮動小数点数と上限を与える浮動小数点数を計算し、**浮動小数点数を両端とする区間の中に真の解を包み込もう**というのが精度保証付き数値計算の基本的なアイデアである。本研究では、Java で CPU の丸めのモードを指定出来るように丸めのモードを変更する C 言語のコードを JNI を用いて利用する FPU クラスを作成した。FPU クラスは丸めのモードを設定するメソッド **setRoundMode()** と丸めのモードを確認するメソッド **getRoundMode()** を持つ。

### 2.2 プロセッサ集合を用いた精度保証

プロセッサ集合 (PSet：Processor Set) とは、マルチプロセッサを有するコンピュータにおける CPU の集合であり OS のスケジューラがプロセス（タスク）を割り当てる単位である。多くの OS（Soaris,HP-UX,Digital-Unix,Linux) で利用可能であり、OS に指示を与えることで CPU を別の PSet に再配置可能である。プロセッサ集合同士は完全に独立しているので、特定のプロセスを固定することが可能となり、精度保証を他のプロセスの影響を受けないように実行することが可能となる。初期状態ではプロセッサ集合 PSetID:0 が存在し、そこに全ての CPU とプロセスが割り当てられている [3]。図 1 にプロセッサ集合を用いた精度保証の流れを示す。

Java でプロセッサ集合を利用出来るようにプロセッサ集合を操作する C 言語のコードを JNI を用いて利用する ProcessorSet クラスを作成した。ProcessorSet クラスは表 1 のようなメソッドを持つ。

表 1: ProcessorSet クラスのメソッド

| メソッド | 機能 |
|---|---|
| create | PSet を作成 |
| destroy | PSet を削除 |
| assign | CPU を PSet へ配置 |
| bind | プロセスを PSet へ固定 |
| getattr | PSet の属性を取得 |
| setattr | PSet の属性を変更 |
| ctl | PSet の情報を取得 |

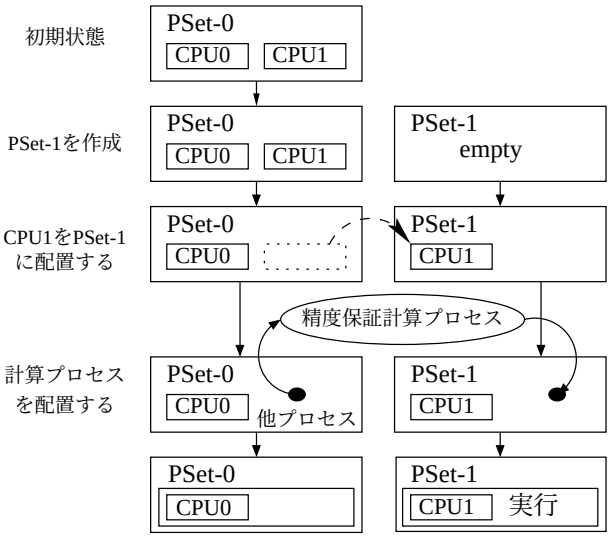


図 1: プロセッサ集合を用いた精度保証

### 2.3 精度保証付きベクトルの内積計算の例

2 つのベクトル $x$ と $y$ の内積 ($z = x \cdot y$) の精度保証付き計算を行なうプログラムを次に示す。このプログラムを実行すると真の解に対して上限と下限が得られる。また、行列計算に関しては我々のグループで開発されてい

†九州工業大学

る Java 数値計算パッケージ [5] を利用した。

```
public class Pset_inner {
 public static void main(String[] args) {
  Matrix.setFormatLength(25);
  Matrix XR=null, XC=null, YR=null, YC=null;
  try {
     FPU.setRoundMode(FPU.RoundDown );
     XR = Matrix.readMxFile("xr.mx"); // 半径
     FPU.setRoundMode(FPU.RoundUp);
     XC = Matrix.readMxFile("xc.mx"); // 中心
     FPU.setRoundMode(FPU.RoundDown);
     YR = Matrix.readMxFile("yr.mx"); // 半径
     FPU.setRoundMode(FPU.RoundUp);
     YC = Matrix.readMxFile("yc.mx"); // 中心
  } catch (IOException e) {
     e.printStackTrace();
  }
  // プロセッサ集合 1 作成
  ProcessorSet ps1 = new ProcessorSet();

  // プロセッサ集合 1 に CPU1 を配置
  ps1.assign(1);

  // プロセスをプロセッサ集合 1 へ固定
  ps1.bind(ProcessorSet.P_PID,
                          ProcessorSet.P_MYID);
  // 上向き丸め内積計算
  FPU.setRoundMode(FPU.RoundUp);
  Matrix ZU
    = (XC.multiply(YC)).add(XC.multiply(YR))
    .add(XR.multiply(YC)).add(XR.multiply(YR));

  // 下向き丸め内積計算
  FPU.setRoundMode(FPU.RoundDown);
  Matrix ZD
    = (XC.multiply(YC)).sub(XC.multiply(YR))
    .add(XR.multiply(YC)).add(XR.multiply(YR));

  // プロセッサ集合 1 を削除
  ps1.destroy();

  Matrix ERR = ZU.sub(ZD);  // 誤差を計算
 }
}
```

### 2.4 連立一次方程式の精度保証

ここで、連立一次方程式を精度保証付きで計算する方法を述べる。ただし、$A$ は $n \times n$ 行列、$x, b$ は $n$ ベクトルとする。連立方程式 $Ax = b$ の近似解が与えられたときに、その近くに真の解が存在するか否かの判定や真の解との誤差を求めるための原理となる定理を示す [1]。

定理

方程式 $Ax = b$ の近似解 $\tilde{x}$ と $A$ の逆行列の近似行列 $R$ が求められたとき行列 $G = RA - I$ が不等式

$$\|G\| < 1$$

をみたすときは、逆行列 $A^{-1}$ が存在し

$$\|A^{-1}\| \leq \frac{\|R\|}{1 - \|G\|}$$

および $x^*$ を真の解とすると

$$\|x^* - \tilde{x}\| \leq \frac{\|R(b - A\tilde{x})\|}{1 - \|G\|} \leq \frac{\|R\| \, \|(b - A\tilde{x})\|}{1 - \|G\|}$$

連立方程式に対して、$A$ の近似逆行列 $R$ を求め、近似解 $x = Rb$ を計算する。このとき、連立方程式の真の解の存在の十分条件を検証し、もし、存在する場合には真の解 $x^*$ と $x$ との誤差 $error = \|x - x^*\|_\infty$ を計算するプログラムを次に示す。

```
// PSet-1 を作成し、プロセスを固定

FPU.setRoundMode(FPU.RoundDown);
Gd = (R.multiply(A).sub(E)).abs();
rd = (A.multiply(x).sub(b)).abs();

FPU.setRoundMode(FPU.RoundUp);
Gu = (R.multiply(A).sub(E)).abs();
ru = (A.multiply(x).sub(b)).abs();

// 成分毎に値が大きな方を取得
ru_max = rd.maxElementWise(ru);
Gu_max = Gd.maxElementWise(Gu);

// 最大値ノルム
double R_norm = R.norm("Inf");
double G_norm = Gu_max.norm("Inf");
double r_norm = ru_max.norm("Inf");

FPU.setRoundMode(FPU.RoundDown);
double D = 1 - G_norm;

if (D > 0) {
  FPU.setRoundMode(FPU.RoundUp);
  double A_inv = R_norm/D;
  double error = A_inv * r_norm;
  System.out.println("誤差"+error);
} else {
  System.out.println("false");
}

// PSet-1 を削除
```

このようにすることで連立一次方程式を精度保証付きで数値計算することが出来る。ここで、利用した norm メソッドは [5] で開発された無限ノルムを返すメソッドである。

## 3. RMI & スレッドによる並列化

精度保証付きで数値計算を行なう場合、上限下限の計算が必要となるため、計算時間が通常の 2 倍以上かかる。そこで、JavaRMI とスレッドを利用して精度保証を行なう部分を並列処理し、処理速度の向上を図る。スレッドと RMI を利用した精度保証の様子を図 2 に示す。

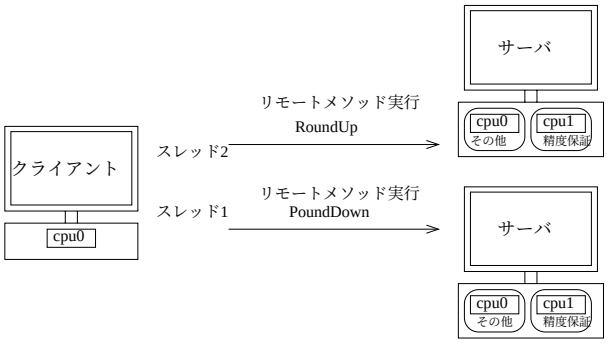


図 2: スレッド・RMI を用いた精度保証

まず、クライアントで新たにスレッドを 2 つ生成する。それぞれのスレッド内で上限を計算するリモートメソッド RoundUp()、下限を計算するリモートメソッド RoundDown() を呼び出し、2 台のサーバで上限と下限の計算を並列に処理する。このように、ネットワークにつながれたデュアルプロセッサマシンが 2 台あれば、精度保証にかかる時間を短縮することが出来る。

### 3.1 連立一次方程式の並列解法

連立一次方程式の精度保証付き数値計算を RMI とスレッドを用いた方法で並列処理し、提案法の有効性を確認する。

連立一次方程式方の次数 $n$ の値を変化させ、逐次処理・並列処理それぞれの方法を用いて処理にかかる時間の測定を行なった。本実験に使用した計算機の環境を表 2 に示す。pc1 をクライアントとし、pc2、pc3 をサーバとした。実験結果を図 3 と図 4 に示す。図 4 は片対数グラフとなっている。

表 2: 計算機の環境

| | CPU[MHz] | メモリ | OS |
|---|---|---|---|
| pc1 | Celeron 700 | 256MB | RedHatLinux 7.2 |
| pc2 | PenIII 650×2 | 512MB | RedHatLinux 7.2 |
| pc3 | PenIII 1000×2 | 512MB | RedHatLinux 7.2 |

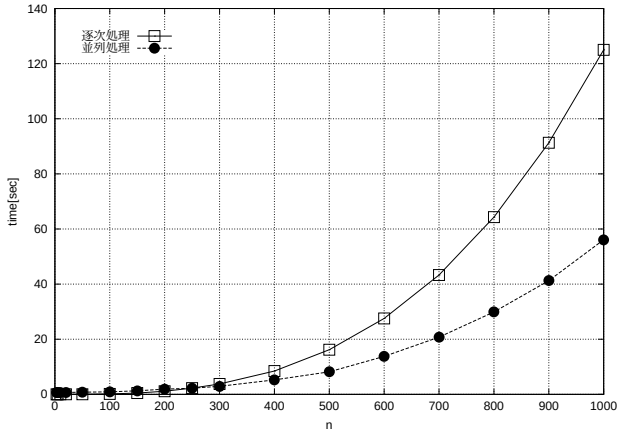


図 3: 逐次処理と並列処理の比較

図 3 より、$n$ の値が大きくなれば、RMI を用いて並列化することで精度保証にかかる時間が 1/2 になったとわかる。$n$ の値が小さなときは通信によるオーバーヘッドの割合が大きなため RMI を利用して並列処理をしても効果はなかった。効果があらわれるのは $n \leq 300$ のときであった。

## 4. まとめ

本研究では、低コストで精度保証付き計算を確実に高速に実現するために、JavaRMI 分散オブジェクトを利用することによって複数のデュアルプロセッマシンを用いた並列処理の方法を提案した。行列の乗算、連立一次方程式の精度保証付き数値計算において提案法の有効性を確認した。図 5 に示すようにネットワークにつながれた複数台のデュアル CPU マシンを用いれば、精度保証付き計算をさらに高速に行なうことが可能である。

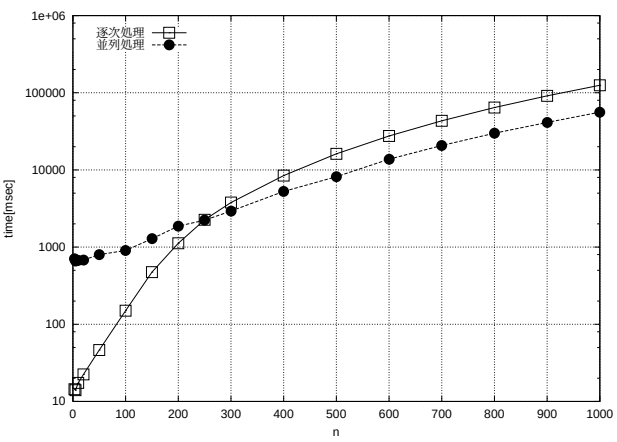


図 4: 逐次処理と並列処理の比較 (対数グラフ)

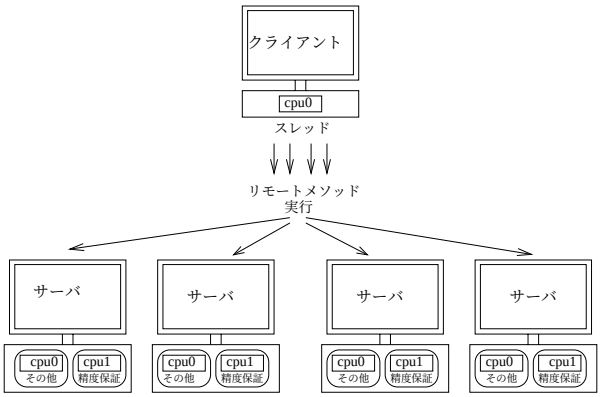


図 5: さらに高速化を図る方法

## 参考文献

[1] 大石進一. **精度保証付き数値計算**. コロナ社, 2000.

[2] 大石進一. *Linux* **数値計算ツール**. コロナ社, 2000.

[3] ヒューレット・パッカード社. hp-ux 11i プロセッサ・セット (processor sets:pset), 2003. http://www.jpn.hp.com/products/software/oe/hpux/partition/resource/psets/index.html.

[4] 竹井 公則. プロセッサ集合を用いた精度保証付き並列計算に関する研究, 2001. 平成 13 年度卒業論文.

[5] 松木 毅. Java 数値計算パッケージの開発と制御系設計への応用, 2002. 平成 14 年度卒業論文.

[6] 中山 茂. *Java* **分散オブジェクト入門**. 技報堂, 2000.