区間演算ライブラリを用いるプログラムのための依存グラフ分割と再利用による効率的なデータフロー並列処理

佐原 海哉† 川端英之‡ 谷川一哉‡ 弘中哲夫‡
広島市立大学情報科学部情報工学科† 広島市立大学大学院情報科学研究科‡

1 はじめに

精度保証付きの数値計算では，精度が高くなるにつれて数値計算時間が増加する．そのため，精度保証付きライブラリの高速化とともに，それを用いて記述されたプログラムの高速処理手法が求められている．

我々は，精度保障付き区間演算ライブラリ MPFI[1] を用いた数値計算プログラムを高速に実行できるようにするライブラリを開発している (GMPFI と呼ぶ [2])，GMPFI ライブラリではデータフローグラフとスケジューラを作成する．データフローグラフはプログラム中の全ての区間演算の依存関係を表す．スケジューラによりデータフローグラフ全体を確認せずに実行可能な演算ノードを取得し，並列処理により区間演算を用いた数値計算を高速に実行する．

データフローグラフに従って並列処理を行う GMPFI では，並列性のあるアプリケーションにおいて高速化が期待できる．しかし GMPFI のデータフローグラフの生成はアプリケーション全体の実行時間の大半を占めている．実際，論文 [2] において，仮数部長が 5000[bit] を超えないと並列処理の効果がないと報告されている．本研究の目的は，短い仮数部長の計算でも高速化できる手法の追求である．

2 改善手法

データフローグラフが大規模なものになるとデータフローグラフ作成時間が膨大となり性能が低下する．その問題点を解決するためにデータフローグラフ作成時に 1 つの大きなグラフを小さなグラフに分割する手法を提案する．

データフローグラフの作成は，ソースプログラムの制御構造に従って次々と依存関係グラフを積み上げることにより行う．その際，同じパターンの依存関係の部分があればその部分のグラフを繰り返し使える可能性がある．例えば，ループで反復的に計算される漸化式であれば，図 1 のように依存関係グラフを分割し，分割された依存関係グラフを繰り返し使うことができる．図 2 は全体の計算を 5 分割し，小さいデータフローグラフを 5 回再利用するプログラム記述例である. 図 2 のようなプログラムの記述により，全体の計算を N 分割すればデータフローグラフの積み上げ回数を $\frac{N}{5}$ 回に削減することができる. 具体的には, 図 1 より左のデータフローグラフから右のデータフローグラフのように依存先を整理することで再利用が可能となる.

図 1 のような部分グラフの再利用と分割において必要なものは以下の 4 つである.

1. データフローグラフを構成するデータ
2. 再利用に必要な MPFI 型の数値
3. 再利用先の変数のノード番号
4. データフローグラフの分割回数

1 は, データフローグラフ作成時点でデータフローグラフを構成するデータをコピーする．そしてデータフローグラフの演算終了時にコピーしていたデータをコピー元に代入する．2 はデータフローグラフの演算終了時に図 1 の右側の部分グラフのように変数を依存先に代入する．3 はデータフローグラフの再利用時，最後に計算を終えた変数のノード番号を 1 のように対応する根本の変数ノード番号として再定義する．4 の再利用回数はデータフローグラフの分割数と対応する. 分割した回数分再利用することで, 再利用を行わない場合と数値計算結果が一致する.

以上の 4 項目を図 2 のデータフローグラフの再利用に向けた整備の関数で行われる．GMPFI に対して，提案手法であるデータフローグラフの再利用処理を導入した並列処理システムを，本稿では DGMPFI と呼ぶ.

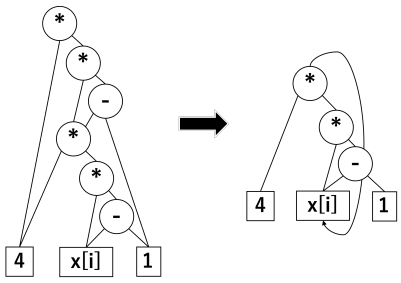

図 1: 依存関係グラフの再利用

```
int main(void){
        …
        for( int i=0 ; i < N/5; i++){
                データフローグラフの積み上げ(a, b, c);
        }
        for(j = 0; j < 5; j++){
                作成したデータフローグラフを並列実行( j );
                データフローグラフの再利用に向けた整備( j );
        }
        …
}
```

図 2: データフローグラフの分割と再利用を行うプログラム記述例

Efficient dataflow parallelization through iterative use of partial use of dependency graph for programs using interval arithmetic libraries

Kaiya Sahara†Hideyuki Kawabata‡Kazuya Tanigawa‡Hironaka Tetsuo‡
†Department of Computer and Network Engineering, Hiroshima City University
‡Graduate School of Information Sciences, Hiroshima City University

3 評価

3.1 並列度2のデータフローグラフ

評価に用いるアプリケーションとして定常的に並列度2のデータフローグラフを作成した．図3は定常的に2並列処理が可能なデータフローグラフである．以下の漸化式の計算における反復1回分の処理に対応するデータフログラフは図3に示す通りである．

$$a_n = c + a_{n-1} \times b, \quad e_n = c + e_{n-1} \times d$$

評価には, 以下の数値で多倍長演算を行う．

$$a_0 = 2.01,\ e_0 = 1.99,$$

$$b = 1.01,\ c = 1.05,\ d = 0.99,\ n = 32768$$

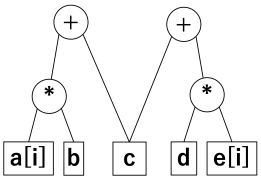

図 3: 2並列データフローグラフ

3.2 DGMPFIとGMPFI，MPFIの実行時間比較

GMPFIとDGMPFIのワーカ数は4,DGMPFIのデータフローグラフの分割数は16に固定している．図4にMPFI，GMPFI，DGMPFIの実行時間の比較を表す．図4の横軸は仮数部長 [bit]，縦軸は実行時間 [s] である．図4中のDFG_GMPFI，DFG_DGMPFIは，GMPFI，DGMPFIの実行中のデータフローグラフ生成時間を表す．図4より, よりデータフローグラフの生成時間がDGMPFIはGMPFIに比べて約 $\frac{1}{16}$ に削減できている．また，GMPFIの実行時間は9216[bit]以上でMPFIより速くなっているが，DGMPFIでは3072[bit]以上でMPFIより速くなっている．定常的に並列度が2のデータフローグラフの実行時間においては，DGMPFIはGMPFIに比べ $\frac{1}{3}$ の仮数部長でMPFIより速くなっている．

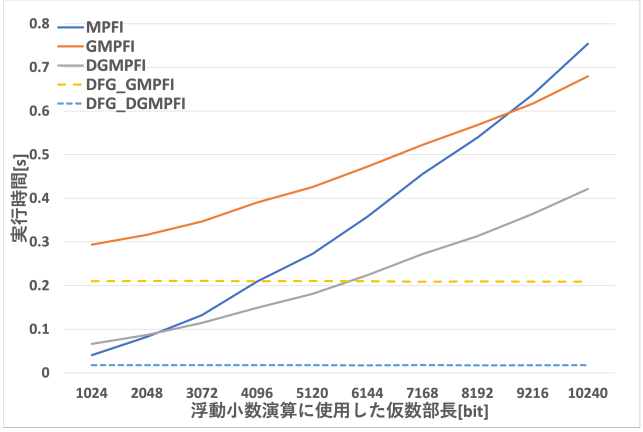

図 4: 2並列データフローグラフの演算の実行時間

3.3 DGMPFIのワーカ別の実行時間

ワーカ数を1，2，3，4の4つ，データフローグラフの分割数は16に固定して実行時間を計測した．図5にDGMPFIのワーカ数ごとの実行時間比較を表す．図5の横軸は仮数部長 [bit]，縦軸は実行時間 [s] である．図5より，ワーカ数2，3，4の実行時間に差がほとんどない．これは今回使用したアプリケーションの並列度が定常的に2であるため，ワーカ数を2以上にしても実行速度の向上は見込めないことが原因である．

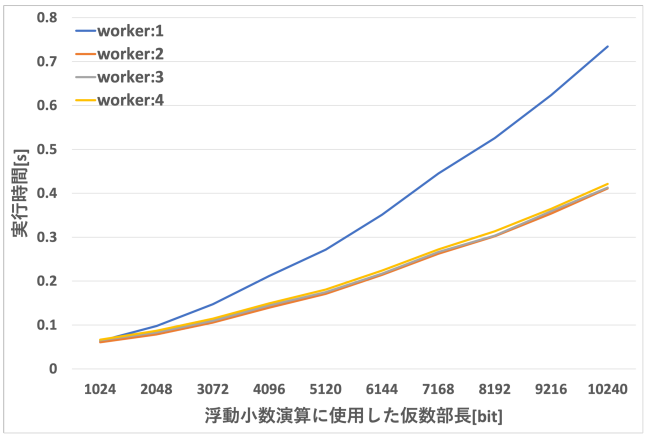

図 5: 並列列度2のデータフローグラフのワーカ数別のDGMPFIの実行時間

4 まとめと今後の課題

本研究では区間演算に用いるデータフローグラフの分割と再利用を行なった．これにより分割数に応じた割合分データフローグラフの作成コストを抑えることができた．その結果，データフローグラフの分割と再利用を導入する前と比較し，より短い仮数部長でMPFIより速くできた．

今後の課題の1つは，より短い仮数部長でMPFIライブラリよりDGMPFIの実行速度を速くすることである．

参考文献

[1] MPFI: `https://perso.ens-lyon.fr/nathalie.revol/software.html`.

[2] 秦将裕, 川端英之, 弘中哲夫, “ループ境界を越えた動的スケジューリングによる区間演算プログラムの高速化”, 2022年度情報処理学会第84回全国大会 講演論文集, March 3, 2022.