2K-05

# 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

秦 将裕† 川端英之† 弘中哲夫†
広島市立大学大学院情報科学研究科†

## 1 はじめに

多倍長精度での精度保証付き数値計算を行うにあたり，MPFI[1] などの区間演算ライブラリが利用できる．しかしながら，精度保証付き数値計算は高速化が望まれており，メモリ使用量の削減 [2]，アルゴリズムの改善，並列処理 [3] など区間演算の高速化のための多くの研究がなされている．これに対し我々は，プログラム全体を俯瞰しつつ並列に実行可能な区間演算を抽出し動的にスケジューリングすることで区間演算プログラムを高速処理する手法を研究している．本稿では，開発中の動的スケジューリングシステムを用いた区間演算プログラムにおける動的スケジューリング手法の詳細とその性能評価について述べる．

## 2 動的スケジューリングシステムの概要

図 1 にシステムの概略図を示す．本システムは，本システム用に書き換えられた区間演算プログラムからデータフローグラフを生成し，その情報を用いて複数のスレッドが区間演算を並列処理し区間演算プログラムの結果を出力する．

本システム用に書き換えられた区間演算プログラムとは，MPFI の API と同じ入力形式で記述されたプログラムに並列実行するための関数を追加したプログラムで構成される．ここでの MPFI の API と同じ入力形式で記述されたプログラムは呼び出されるとすぐに計算処理を行わず並列処理の際に使用するデータフローグラフの生成のためのプログラムである．しかし，MPFI の API と同じ入力形式であるため，MPFI で記述された区間演算プログラムの簡単な書き換えで済む．並列実行するための関数は，生成されたデータフローグラフを実行するための関数であり，上記のデータフローグラフを生成するためのプログラムの後に追加するだけである．そしてこの関数は，ユーザがスレッドの生成や破棄，データ分割を意識せずとも記述された演算プログラムを並列に実行できる関数となっている．

本システムでは，データフローグラフを生成すると同時に実行可能な演算を検知し，情報をキューに追加する処理も行う．そして，並列処理時にスレッドがキューから実行可能な演算の情報を取得し計算を行う．計算後にはデータフローグラフから計算した演算ノードのエッジ先を確認し，エッジ先が実行可能になったかチェックする．計算と計算後の確認の作業を全ての演算を計算し終えるまで続けることで本システムは動的スケジューリングシステムを実現している．

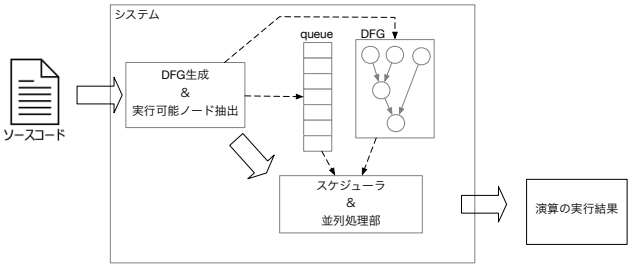


図 1: システム構成図

本システムは，データフローグラフの生成による並列処理のオーバーヘッドの増加が見込まれるが，キューの操作とデータフローグラフの活用によりスムーズなタスクの受け渡しにより高速処理を可能にする．そして，多倍長精度での区間演算は仮数部長が長ければ長いほど実行時間が増加するため，タスクの粒度が小さい場合でもオーバーヘッドの割合の減少すると考えれる．そのため，本システムを用いることで並列性のある区間演算プログラムでは高速化が期待できる．

## 3 設計

### 3.1 データフローグラフの生成

本システムでは，並列処理できる演算をデータフローグラフを基に判別し実行する．そのため，まずはじめに区間演算プログラムからデータフローグラフの生成する．生成には，gmpfi_add(a, b, c) のような関数をユーザが記述することでシステム上でデータフローグラフの生成される．上記の関数からは，ノード a はノード b，c のデータを加算するといった情報を取得できる．

本システムで使用するノードは 2 種類あり，演算処理を意味する演算ノードと演算のオペランドとして使用される数値ノードである．データフローグラフ生成時にノード b，c がどちらも数値ノードであるときには，a は実行可能なノードとしてシステムが演算ノードの識別子がをキューに追加する．キューは，実行可能なノードの情報を格納し並列処理の際にスレッドがデータフローグラフ全体を探索せずとも実行可能な演算ノードを取得するために使用される．

### 3.2 データフローグラフを用いた並列処理

図 2 にスケジューラと並列処理部の内部構成図について示す．スケジューラと並列処理部では，図 2 で示すように，ワーカと呼ばれるスレッドが前節で得られたデータフローグラフとキューを用いることで並列処理を行う．

並列処理は，入力のプログラム中で実行用の関数が呼ばれた時に実行される．並列処理では，

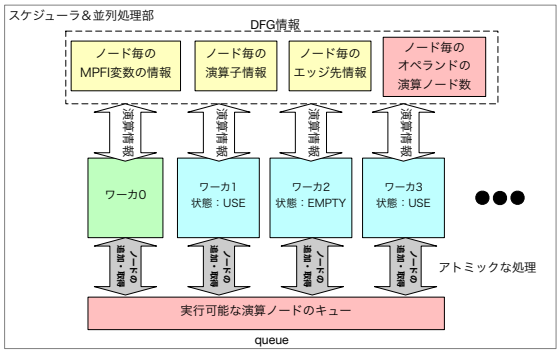


図 2: スケジューラと並列処理部の内部構成

```
1    #include “gmpfi.h”
2    int main(void){
3    gmpfi_t a, b, c, d, e, f, g;    //データ型変数の定義
4    …
5
6    gmpfi_mul(a, b, c);             //データフローグラフに乗算のノードを追加
7    gmpfi_sub(d, a, c);             //データフローグラフに減算のノードを追加
8    gmpfi_div(e, d, f);             //データフローグラフに除算のノードを追加
9    gmpfi_add(g, e ,a);             //データフローグラフに加算のノードを追加
10   …
11
12   gmpfi_cal(worker);              //作成したデータフローグラフをworker数で並列実行する
13   …
14   }
```

図 3: GMPFI に用意した API の使用例

1. キューを用いた実行可能な演算ノードの管理
2. データフローグラフを用いた演算情報と依存したノード情報の確認
3. 完了した演算の結果に依存しているノードの継続した計算

により行われる．3 では，ワーカが実行可能となった演算ノードを引き継ぎ計算することにより，全てのワーカが同時にアクセスする可能性のあるキューに対するノードの追加，取得処理を削減することができる．

## 4　実装

前節で設計した並列処理システムを GMPFI と呼ぶ．図 3 に GMPFI に用意した API の使用例を示す．gmpfi_t は，区間演算を並列に行うためのデータ型であり，gmpfi_init2 は gmpfi_t で扱うデータ型変数の仮数部長のメモリ領域を確保する関数である．gmpfi_add，gmpfi_sub，gmpfi_mul，gmpfi_div はそれぞれ名前に対応した演算子のデータフローグラフを生成する．gmpfi_cal は，ワーカスレッドを生成し，データフローグラフを基に並列実行する関数である．

## 5　評価

前節で実装した並列処理システムの性能評価を行うために，ヒルベルト行列を係数行列に持つサイズ 64 の連立一次方程式の LU 分解による求解プログラムを MPFI，GMPFI をそれぞれ用いて記述し，GMPFI では 1〜4 スレッドで計算速度を実測した．評価では，Intel Core i9-9900K の CPU でソフトウェアは MPFI 1.5.3 を用い，実行ファイルのコンパイル時の最適化オプションは-O3 とした．

図 4 に MPFI と GMPFI の仮数部長に対する実行時間を示す．図 4 より，仮数部長が 4000[bit] までは，GMPFI

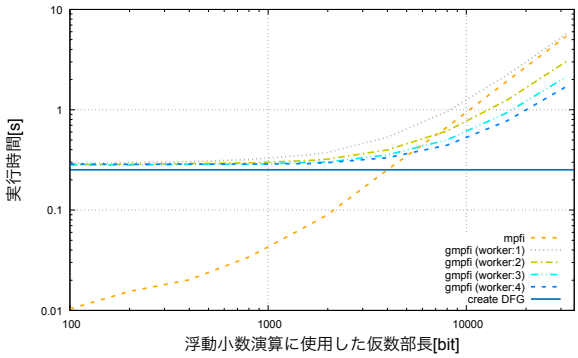


図 4: 連立一次方程式求解の実行時間

の実行時間が MPFI に比べ大幅に増加しておりデータフローグラフ生成の時間が大半を占めていることがわかる．実際に仮数部長が 4000[bit] までは，データフローグラフ抽出に要する時間の実行時間に占める割合が 60%以上であった．これは，計算処理よりも並列に実行するための情報をすることに多くの時間を費やしていることを意味する．

時間削減のために，データフローグラフを分割展開し実行することで計算量を抑える．または，一度生成したデータフローグラフを保存しておき使い回すなどの生成時間が実行時間に占める割合を減らすなどの改善が必要であると言える．一方で仮数部長が 5000[bit] を越えると区間演算次第に並列処理の効果が表れており，スレッド数が 4 つで仮数部長 32000[bit] の時に最大 3.2 倍の高速化が得られている．

## 6　まとめと今後の課題

本研究では，データフローグラフを基に区間演算プログラムを並列処理する動的スケジューリングシステムを実装した．

連立一次方程式の LU 分解による求解では，仮数部長によっては MPFI と比べスレッド数 4，32000[bit] で最大 3.2 倍の高速化を達成した．一方で，データフローグラフの生成に多くの実行時間を費やしており改善の余地があるといえる．

今度の課題として，データフローグラフ生成時間の影響を抑えるために，グラフを分割し展開することやグラフ情報を再利用できるようにすること，[2] や [3] などの MPFI 以外の区間演算ライブラリを用いた実装などが挙げられる．

## 参考文献

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

[2] F. Johansson,“Arb: Efficient Arbitrary-Precision Midpoint-Radius Interval Arithmetic,” in IEEE Transactions on Computers, vol. 66, no. 8, pp. 1281-1292, 1 Aug. 2017.

[3] 秦将裕, 川端英之, 弘中哲夫, “軽量な同期方法を用いた区間演算ライブラリの並列化”, 2021 年度 (第 72 回) 電気・情報関連学会中国支部連合大会 講演論文集, October 23, 2021.