Processing で視覚的な並列処理を行うためのフレームワークの開発と評価

A Parallel Processing Framework for Processing Language

岸野 竜司[*1]，笠波 将太郎[*1]，鈴木 優大[*1]，西口 敏司[*1]，橋本 渉[*1]，水谷 泰治[*1]
Ryuji KISHIMOTO[*1], Shotaro KASANAMI[*1], Yudai Suzuki[*1], Satoshi NISHIGUCHI[*1], Wataru HASHIMOTO[*1], Yasuharu MIZUTANI[*1]
[*1] 大阪工業大学　情報科学部
[*1]Faculty of Information Science and Technology，Osaka Institute of Technology
Email: e1b16024@st.oit.ac.jp

あらまし：本研究では Processing 言語を用いた図形アニメーションプログラムのための並列化フレームワークを提案する．本研究の目的は平易な教材を扱える並列プログラミングの学習環境を整え，並列プログラミング初学者の学習を支援することである．この目的を達成するために，Processing 言語で並列処理のための記述を隠蔽したクラスを作成した．本稿では作成したフレームワークおよび図形アニメーションを用いた並列処理の実験結果について報告する．
キーワード：並列処理，フレームワーク，図形アニメーション，Processing

1. はじめに

並列処理とは複数の PC や CPU，コアに処理を分散させることでプログラム全体の実行時間を短くする技法である．現在では一般的な PC においてもマルチコア CPU を搭載しており，並列プログラムをうまく記述すれば実行速度をコア数倍まで向上できる．そのため，今後は一般的な開発者も並列プログラミングを学ぶことが，高性能なアプリケーションを開発する上で重要になってくる．

しかし，並列プログラミングの学習は初学者の学習意欲を維持させにくい．一般に並列処理は数値計算を対象にすることが多く，数値計算に馴染みのない初学者には教材自体が難しい．また，これらは文字ベースのプログラムであるため面白みに欠ける．さらに，並列処理の実行環境に応じて MPI[1]，CUDA[2]といった，初学者にとっては煩雑なライブラリや言語を用いてプログラムを作成する必要がある．これらに対し，並列処理のための構文を追加して並列プログラミングの煩雑さを低減した教育向けのプログラミング言語の研究[4,5]や，Web ブラウザ上に表示させた 2 次元グリッドとアバタを操作させて並列処理を学習させる研究[6]が行われている．

本研究では，視覚的にわかりやすい図形アニメーションに着目し，図形アニメーションプログラムを並列プログラミングの教材にできる環境の構築を目的として，Processing 言語[3]上で簡単に並列処理を行うためのフレームワークを開発した．

2. 提案手法

2.1 Processing

Processing は Java をベースとしたグラフィック機能に特化したプログラミング言語および統合開発環境である．また，ビジュアルデザイン等を制作するためのプログラミング言語として開発され，プログラミング初心者が比較的簡単にプログラムを組み，視覚的に表現しやすいという特徴を持っている．

```
1 void setup(){
2   size(180,100);  //描画領域を作成
3 }
4 void draw(){
5   ellipse(30, 30, 50, 50); //円を描画
6   rect(60, 40, 50, 50);    //四角形を描画
7   line(120, 20, 170, 90);  //直線を描画
8 }
```

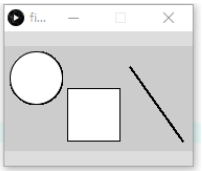

図 1．Processing における記述とその実行例

Processing のプログラム例を図 1 に示す．setup()は最初に一度だけ呼び出され，その中で size()を呼び出すことで描画領域のウィンドウサイズを設定する．次に draw()が繰り返し呼び出され，draw 内に記述した描画処理による図形アニメーションが行われる．この draw は 1 秒間に 60 回呼び出される．

2.2 フレームワーク化のアイデア

並列処理に関する煩雑な処理を親クラスに隠蔽し，このクラスを継承して，並列化したいプログラムを記述するだけで，並列に図形アニメーションを記述できるフレームワークを開発すれば並列プログラミング初学者が学習しやすいと考えた．隠蔽する処理として，(1)draw 内部の描画処理をマルチスレッドで並列化する処理，(2)各スレッドの画像を統合する処理，を隠蔽することにした．その理由は並列処理のための記述を減らすためである．(1)では並列プログラミングにおいて並列化するために必要な記述が多く，(2)では複数の画像を統合するための記述が多い．それらをそれぞれ 1 つのメソッドにまとめることができれば煩雑な事前知識を削減でき，初学者にとって簡単に扱えるようになると考えた．

3. 作成した並列処理フレームワーク

2.2 節で述べた親クラスを Parallel クラスとして実装した．Parallel クラスには並列処理を記述するためのメソッドがまとめられている．使用者は Parallel クラスを継承したクラスを作成し，Parallel クラスに含まる抽象メソッド run をオーバーライドする．次に，継承したクラスを並列処理したい数だけインス

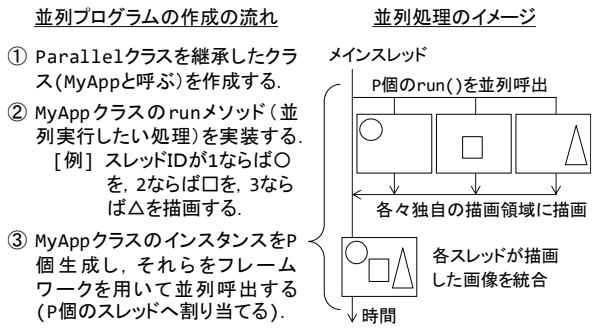

図 2. Parallel クラスを用いたプログラムの流れ

```
class MyApp extends Parallel{
  void run(){
    pg.beginDraw();  //描画開始
    if(myrank==0){   //スレッド番号で分岐
      pg.ellipse(30, 30, 20, 20);
    }else{
      pg.rect(60, 60, 20, 20);
    }
    pg.endDraw();    //描画終了
  }
}
MyApp[] app;
void setup(){
  size(100, 100);
  app = new MyApp[2];
  app[0] = new MyApp();
  app[1] = new MyApp();
}
void draw(){
  app[*]をフレームワークに
  渡して並列呼出
}
```

図 3. プログラムの概要と実行例

タンス生成し，それらのインスタンスをフレームワークが提供する並列呼出のためのメソッドに渡すことで，インスタンスの数だけ run メソッドが起動され，それらが並列に処理される.

Parallel クラスを利用した簡単なプログラムの流れを以下の図 2 に示す．このプログラムは 3 スレッドで並列処理される．各スレッドが○と□と△をそれぞれ独自の描画領域に描画する. Parallel クラスを継承した子クラス内に記述する．その後並列呼び出しを行うと各スレッドがそれぞれ描画し，それらを統合するというプログラムの流れになっている．左側かプログラムの記述の流れで右側が記述に対応したプログラムの動作の流れである.

4. 実験

今回作成したフレームワークを用いることで並列処理を簡単に実装できるか，また並列の効果を視覚的に体感できるかを簡単な図形を並列で描くプログラムとプランクトンの活動をシミュレーションするプログラムを作成し実験を行った.

4.1 フレームワークを用いた図形描画プログラム

2 つのスレッドで○と□を描画するプログラムを作成した. 図 3 にプログラムの概要と実行例を示す. プログラムでは Parallel クラスを継承した MyApp クラスを作成し, その中の run でスレッド番号(myrank)に応じて○または□を描画する．setup では MyApp のインスタンスを 2 つ生成しておき，draw でそれらを並列呼び出しすることで 2 つの run が並列実行される．各スレッドが描画した画像は並列呼出の終了時に統合され，1 つの画像として表示される.

4.2 プランクトンの活動シミュレーション

プランクトンの活動をシミュレートするプログラムを作成し，フレームワークによる並列化の効果を

```
Class MyApp extends Parallel{
      ⋮
  run(){
    スレッド1が前半250個のプランクトンの処理
    スレッド2が後半250個のプランクトンの処理
  }
}
      ⋮
draw(){
  フレームワークによる並列呼出
      ⋮
}
```

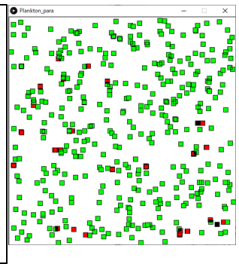

図 4. プランクトンプログラム

表 1. プランクトンプログラムの実行時間(ミリ秒)

スレッド数		1	2	4	8
実行時間	画像統合	1.5	8.4	20.3	45.7
	計算	15.6	8.1	5.8	6.3

調べるために実験を行った．このプログラムとは，動物プランクトンと植物プランクトンの生態活動を単純な規則に基づいてシミュレートするものである.

このプログラムではプランクトンを表すオブジェクトを配列に格納し，それを各スレッドが均等に分割してシミュレーションの処理を担当する. 例えば，総数 500 のプランクトンを 2 スレッドで処理する場合, スレッド 1 が 0～249 番, スレッド 2 が 250～499 番のプランクトンに対する処理を行う．それらのプログラムの大まかな流れと実行の様子を図 4 に示す.

このプログラムを 1，2，4，8 スレッドで実行したときの，アニメーションの 1 コマ分の処理に要した時間を表 1 に示す．表 1 より，並列化により計算時間は短くなっているが，画像統合に多くの時間を要したため，総実行時間は短縮できていない．これより，現段階のフレームワークでは，画像統合に対するオーバーヘッドが大きく，並列化による性能向上はまだ実現できていないといえる.

5. まとめ

本研究では初学者の学習支援のために並列処理を簡単に実装することができるフレームワークの開発を行った．今後の課題として，実際に初学者に使用してもらうための教材開発とその適用実験，および描画画像の統合処理の性能向上があげられる.

謝辞 本研究は JSPS 科研費 JP18K02916, JP17K00500 の助成を受けたものである.

参考文献

(1) https://www.mpi-forum.org/
(2) https://developer.nvidia.com/cuda-zone
(3) https://processing.org/
(4) 田中寛章,藤井健太,礒淵郁也,水谷泰治."複数のハードウェアでの共通操作に着目した教育用並列プログラミング言語の提案". 第 78 回情報処理学会全国大会, 5ZC-02, (2016-03).
(5) I. Finlayson, J. Mueller, S. Rajapakse, D. Easterling. "Introducing Tetra: An Educational Parallel Programming System". IEEE Int. Parallel and Distributed Processing Symposium Workshop (IPDPSW), pp.746-751, (2015-05).
(6) E. Buzek, M. Krulis. "An Entertaining Approach to Parallel Programming Education". IEEE Int. Parallel and Distributed Processing Symposium Workshop (IPDPSW), pp.340-346, (2018-05).