2B-01

大規模計算環境の構築における簡便性の評価

中村 勝則†，小山 澄怜†，小川 真綾†，黒田 菜穂美†
武庫川女子大学 情報メディア学科†

1. はじめに

高精度計算，分散コンピューティング，データの可視化などを利用する大規模計算の実現においては，複数のプログラミング言語やツール，プログラムライブラリを利用することになる．本研究では，システム開発のための技術に疎い者が短時間でシステム全体を構築できることを示すことが重要な点である．システムの実装にあたっては，高精度計算を高速に実行する部分をC++で，分散処理と可視化にはPython言語と関連ライブラリを用いた．開発したシステムを用いて，マンデルブロ集合の高精度の可視化（10^{260}倍まで拡大する動画の生成）を行った．

2. 研究の動機と目的

近年，TCP/IP 通信を介した分散コンピューティングは各種のプログラミングプラットフォームで実現できるようになってきているが，情報工学の知識に疎遠な利用者にとってはそれらの機能を応用して独自の情報処理システムを組み上げることは未だに難しい．しかし，Python 言語処理系の整備と利用できるソフトウェアライブラリの充実により，情報工学を専門としない利用者にも，分散コンピューティングを利用した大規模な計算処理が実現可能になってきている．

筆者は現在，汎用人工知能の研究のための技術的な準備を進めており，機械学習，深層ニューラルネットワーク，推論の分散処理のためのシステム機能の構築方法を模索している．近い将来，それらを応用した教育活動や研究活動を展開することを考えているが，一般的にはHPC用の設備は高価であり，学校の PC 環境が有効に活用できるのではないかと考えて，今回発表する内容に取り組んだ．

今回の取り組みは，処理に時間のかかるカオス力学系のシミュレーション（マンデルブロ集合の生成）を短時間で大量に行うためのシステム機能を構築し，結果を回収するまでの作業量と時間を評価するものである．

3. 構築したシステム

3-1. マンデルブロ集合可視化ワークベンチ

これはPythonとC++を用いて構築した．（図1参照）

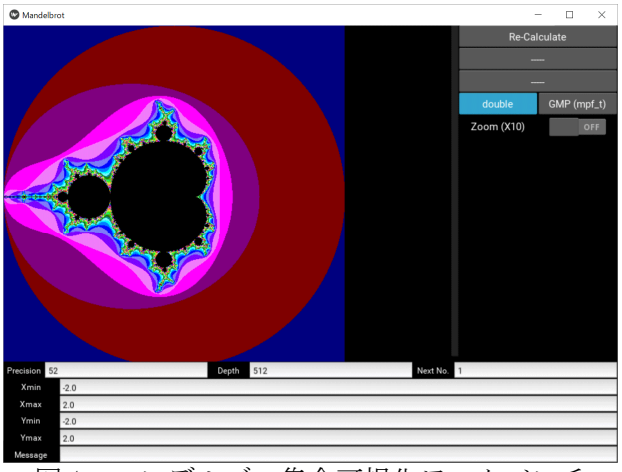
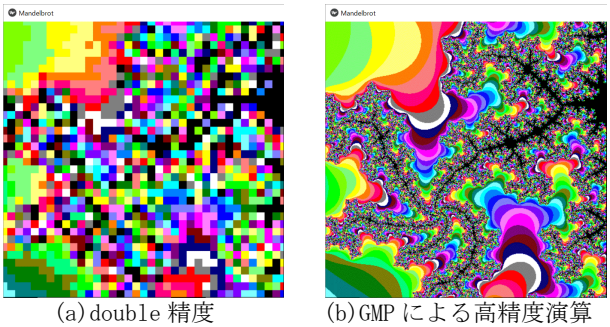


図1. マンデルブロ集合可視化ワークベンチ

マンデルブロ集合の生成には特に時間がかかるので，このためのプログラムは C++で記述した．Python で同様のアルゴリズムを実装した場合，C++と比較して，処理に要する時間が 200 倍以上となることからC++を採用した．

このワークベンチは，ウィンドウに表示されるマンデルブロ集合の画像（複素座標系）の任意の点をマウスでクリックすると，その位置を更に 10 倍に拡大した画像を新たに生成して表示するものである．マンデルブロ集合を生成する計算には，double 型の精度で計算するプログラムと，それ以上の精度で計算するプログラムを別々に用意した．これは，10^{16}倍を超える倍率で画像を生成した場合は，double 型の精度の限界に起因する画像の乱れが発生するためである．（図2参照）高精度の演算にはオープンソースのライブラリである GMP を用いた．このワークベンチを用いて，拡大表示したい画像の場所を選定し，後で説明するプログラムを用いて拡大する過程の画像を大量に生成する．

An assessment of efficiency for development of HPC function
† Katsunori Nakamura, Sumire Koyama, Maya Ogawa, Naomi Kuroda
Mukogawa Women's University, Dept.of Informatics and Mediology



(a) double 精度　(b) GMP による高精度演算

図 2. 計算精度の違いによる画像の違い

3-2. 分散処理用スクリプト生成プログラム

今回は，マンデルブロ集合の画像を 10^{260} 倍まで拡大する動画（30FPS）を 5 分間の長さで作成した．合計フレーム数は 8,855 枚である．GMP ライブラリを用いて 1024 ビットの精度で 500×500 の画素構成のフレームを 1 枚生成するのに，筆者の大学の PC 教室のコンピュータで約 3 分かかるため，約 18 日分の計算時間がかかる．これを 14 台のコンピュータ（4 コアの CPU）で同時に 4 プロセスずつ並行して処理するためのバッチスクリプトを生成し，それを全 PC に TCP/IP 通信にて配布して実行したところ，約 8 時間で全ての処理が終了して 8,855 枚の画像が得られた．

必要なバッチスクリプトを生成するプログラムを Python で作成した．そのスクリプトを実際に実行している様子を図 3 に示す．

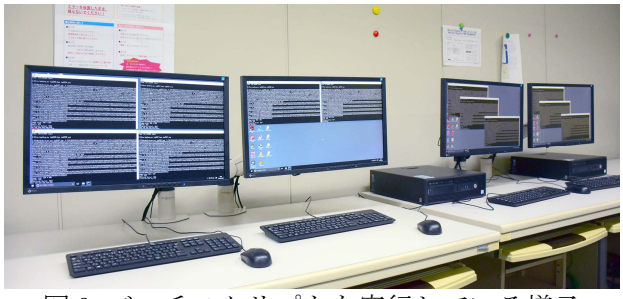


図 3. バッチスクリプトを実行している様子

3-3. 画像回収・ムービー生成プログラム

得られた 8,855 枚の画像を回収して連結するためのプログラムを Python で記述して実行した．このプログラムは外部プロセスとして FFmpeg プログラムを起動するものである．

完成した動画は YouTube にて配信しており，自由に閲覧できる．URL を本稿の「終わりに」のところで紹介する．

4. 評価と展望

今回の一連の開発は筆者の指導の下で学生たち（学部生）の手によって行われた．全てのプログラムを C++や Java で開発する場合と比較して，開発時間が大幅に短縮できたことを実感している．特に Python で利用できるソフトウェアライブラリは，容易な取り扱いができるように整備されているものが多く，異なるライブラリ間でデータを受け渡す際のデータ構造の変換の手間などが少ない．また C 言語で実装した外部プログラムとの連携も容易であり，複数の言語処理系で実装したプログラムを集約して，システム全体を統合することが容易であることがわかった．

今回得られた経験を基にして，PC 教室を HPC 環境として運用するための RJE（遠隔ジョブエントリ）システムを開発する予定である．そのようなシステムが完成した後は，ビッグデータ解析や機械学習，分散人工知能（分散協調型知的エージェント）に関する研究に活かしてゆくことを展望とする．

5. 終わりに

Python は外部プログラムと連携することが容易であることから「グルー言語」などと呼ばれることがある．今後も各種の言語処理系と連携する形でシステム開発をしてゆきたい．

今回作成した動画（マンデルブロ集合の拡大ムービー）を YouTube のコンテンツとして公開している．
作成したムービーの URL
https://youtu.be/h0Tj2wnsT1M

筆者が編纂した Python に関する技術資料をインターネットサイト
https://qiita.com/KatsunoriNakamura/items/b465b0cf05b1b7fd4975
で公開しており，自由にダウンロードして閲覧できる．（フリーソフト）