第3部門　　地盤有限解析コードのGPU並列化に適したデータ構造の検討

神戸大学大学院工学研究科　学生員　○河野　博人
神戸大学大学院工学研究科　正会員　竹山　智英
神戸大学都市安全研究センター　正会員　橘　伸也
神戸大学都市安全研究センター　正会員　飯塚　敦

1. はじめに

地盤有限解析コードを用いた大規模解析計算を実施する際に, 入力データに基づいて処理を実行し, 結果を出力する際に長い時間を要することが課題であった. 計算時間短縮のために有用な手段として GPGPU 技術を用いた GPU 並列化が挙げられる. GPGPU とは, リアルタイム画像処理のための並列計算能力に特化した GPU を汎用計算処理の用途に用いる技術のことで、解析コード内の逐次処理命令を並列処理命令に置き換えることで処理速度の高速化を図る.

本研究ではプログラム内の質量マトリクスを計算する機能のみを抽出したテストコードを作成し、複数種の有限要素に関する質量マトリクスを計算させた場合のテストコードの挙動を調べると共に, 既存のコード内の静的配列を用いたデータ型を保守性・可読性・拡張性を確保した動的配列に置き換えた場合の GPU 並列化効果も検討した.

2. 研究概要

研究のためのテストコードを記述する際に C++と CUDA を使用した. CUDA(Compute Unified Device Architecture)は NVIDIA 社によって提供される GPGPU 環境開発環境である. また, CPU はホスト, GPU はデバイスと呼ばれ, それぞれのメモリはホストメモリ,デバイスメモリと呼ばれる. ホストとデバイスの関係性について述べる. 表1に示すように, ホストからはホストメモリとホストが確保したデバイスメモリにアクセス可能で, デバイスからはデバイスメモリにのみアクセスできる. デバイス上では cudaMalloc や cudaFree という専用の関数を使いホストからのデバイスメモリ確保・開放を行う. 確保したデバイスメモリを計算で使用する際にはカーネルと呼ばれるホストからデバイスを動作させる関数を用いる. 本研究ではホストやデバイスでマトリクスの計算を行うための各種関数のデータ型として静的配列と動的配列の二種類を扱った. 静的配列はコンパイル時に配列の大きさが具体的に決まるもので, 動的配列はコンパイル時には配列の大きさが具体的に決定されず, 関数が実際に使用されるときに配列の大きさが決まるものである. テストコードで扱った質量マトリクスは次数が(節点数×次元)の正方行列であるため, 要素ごとに大きさが異なる. テストコード及び元の地盤有限解析コードではクラスと呼ばれる雛形の中に関数が格納され, クラス同士は継承関係で結びついており, 実際の計算ではマトリクスの計算に必要な関数が連鎖的に呼び出される. 静的配列を用いるとコンパイル時に配列の大きさを具体的に決めるため, 実装する有限要素それぞれに対応した質量マトリクス計算用の関数を実装する必要がある一方で, 動的配列を用いると配列の大きさをコンパイル時に具体的に決めないため, 静的配列使用時に機能は同じであるが計算する配列の大きさが違うために別物とみなされていた複数の関数を統合することができる. 動的配列の性質を用いて統合可能な関数を統合し, 継承関係を構成する関数の総数を減らすことで, クラスの構造が単純化し, 保守性・可読性・拡張性が向上する. 計算時間を計測する際には要素数, 要素の種類, 使用プロセッサ, データ型に関する条件を変えて検証を行った.

表1　各プロセッサのメモリアクセス可否関係

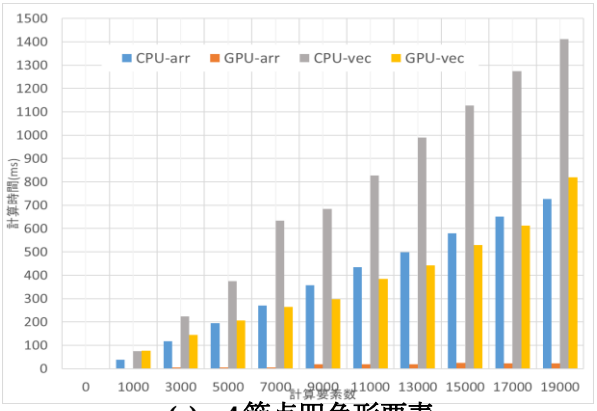
from / to	ホスト	デバイス
ホストメモリ	アクセス可能	アクセス不可
デバイスメモリ (ホストが確保)	アクセス可能	アクセス可能
デバイスメモリ (デバイスが確保)	アクセス不可	アクセス可能

Hiroto KONO, Tomohide TAKEYAMA, Shinya TACHIBANA, Atsushi IIZUKA

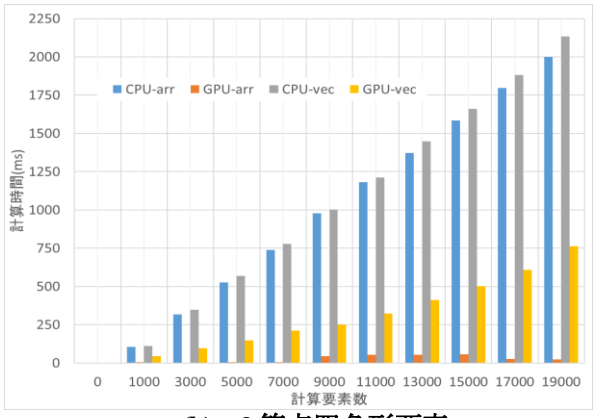
235t112t@stu.kobe-u.ac.jp

3. 検証結果

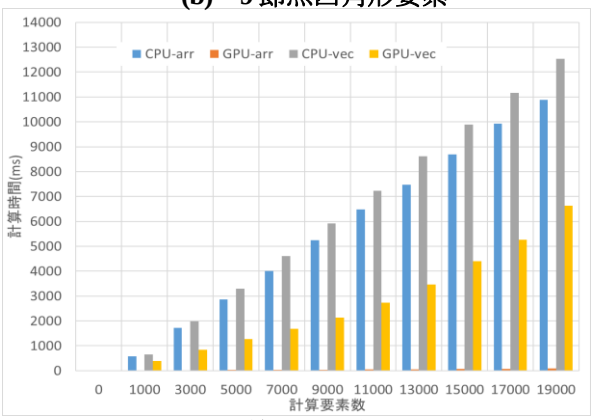
計算時間の検証条件について述べる. マトリクス計算対象の要素は 4 節点四角形要素, 9 節点四角形要素, 8 節点六面体要素, 27 節点六面体要素の四種類であり, 計算要素数は 1000 要素から 19000 要素まで 2000 要素ずつ, プロセッサと使用データ型の組み合わせは CPU-静的配列(arr), GPU-静的配列(arr), CPU-動的配列(vec), GPU-動的配列(vec)の 4 通りである. 検証結果を図 1 に示す. 検証結果に着目すると, どの要素の質量マトリクスを計算した場合でも CPU の計算時間より GPU の計算時間が短くなっていることが分かる. GPU で静的配列を用いた計算を行った GPU-arr と動的配列を用いた計算を行った GPU-vec を比較すると, GPU-arr は要素数が増えても計算時間が殆ど増加せず, GPU-vec は要素数の増加に伴う計算時間の増加が観察される. これは GPU によるデバイスメモリ確保時間による影響であると考えられる. また, 27 節点六面体要素の質量マトリクスを GPU-vec 条件下で計算した場合, 15000 要素以上の計算が実行されなかった. 計算が実行されない理由について考察する. メモリ領域はヒープとスタックの二種類があり, 動的配列はヒープに確保される. テストコードではヒープ領域を 6GB 確保した. 27 節点六面体要素 15000 要素分の質量マトリクスは各成分を double 型(倍精度浮動小数点型)と設定した場合 0.733GB となった. したがって計算が実行されない理由がメモリ領域不足ではないことがわかる. ほかの原因として考えられるのは CUDA 側の不具合である. 前述のとおり, テストコード及び元の地盤有限解析コードは複数の関数の連動により目的のマトリクスを計算しており, これらの関数はクラスの継承関係で結ばれている. この複雑な継承関係が, 計算が実行される際に不具合を引き起こしていると考えられる.



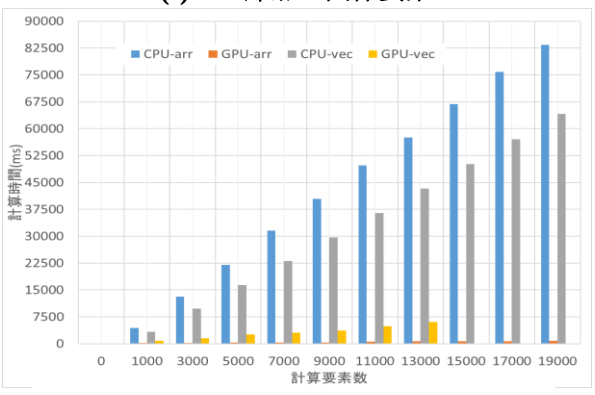
(a) 4節点四角形要素



(b) 9節点四角形要素



(c) 8節点六面体要素



(d) ２７節点六面体要素

図-1 要素数と計算時間の関係

4. おわりに

テストコードではマトリクスの計算機能を複数種の要素に対して実装することができたが要素数が増えると計算が実行できなくなることがあった. また, 配列の大きさがコンパイル時に決まる静的配列よりもコンパイル時に配列の大きさが決まらない動的配列は GPU 並列効果が小さくなる. 引き続き保守性・可読性・拡張性を確保し, GPU 並列化による計算時間短縮効果が見込めるプログラム設計に関して研究を続けていきたい.

参考文献

堂ケ原健：地盤応答解析コードの GPU 並列化と液状化解析　令和 3 年度神戸大学修士論文