2ZC-05

マーチングキューブ法における exclusive_scan 関数の GPU 上での並列処理

Parallelization of exclusive_scan Function for Marching Cubes on GPU

吉川 一輝 †　　吉田 明正 †
Kazuki Yoshikawa　　Akimasa Yoshida

1　はじめに

3D コンピュータグラフィックスのために用いられるマーチングキューブ法 [1] は，ボクセルデータからポリゴンデータを生成するアルゴリズムであり，医療分野等で広く用いられている．マーチングキューブ法を用いた関連研究の一例として，複数の半透明の等値面のリアルタイム高速表示 [2] や組込みシステムにおける MRI 画像処理の並列処理 [3]，2D 医用画像の 3D 可視化のための GPGPU 並列実行 [4] 等が挙げられる．

マーチングキューブ法の処理過程において，ポリゴン面を構成するセルのスキャン及び個数の測定において，exclusive_scan 関数が用いられているが，この関数の実行には多くの処理時間を必要とする．一般的な実装では CUDA の thrust ライブラリである exclusive_scan 関数を用いることが多いが，さらなる高速化を実現するために，本稿では exclusive_scan 関数を独自の CUDA コードにより実装した．本手法によるマーチングキューブ法の CUDA プログラムを GPU 上で並列実行したところ，ポリゴンデータ生成にかかる処理時間が短縮されることが確認された．

2　マーチングキューブ法によるポリゴンデータ生成

本章では，マーチングキューブ法を用いたポリゴンデータ生成方法の一例として，メタボール法 [5] による濃度場のデータをマーチングキューブ法に適用する例を述べる．マーチングキューブ法の処理手順は，最初に各セル内の濃度場データからポリゴン面の有無を算出する．次に，ポリゴン面を含むセルを収集しデータを集約する．集約されたセルは対称性を考慮することで 14 種類のポリゴンパターンに分類される．

2.1　マーチングキューブ法の各処理時間

マーチングキューブ法のプログラムを 4.1 節のマシン上で実行した際の処理時間の内訳を図 1 に示す．全体の処理時間 26.848[ms] の内，99 %以上の処理時間が図 1 の処理 (2) に示すように，thrustscan 関数に割かれている．この関数では exclusive_scan を実行し，ポリゴン面の貼り付けを管理するセルのスキャンを行い，ポリゴン面を有するセルの抽出及びそのセルの総和を求める．

2.2　exclusive_scan アルゴリズム

本節では，exclusive_scan の計算方法とその役割について述べる．exclusive_scan で行われる計算を式 (1) に示す．

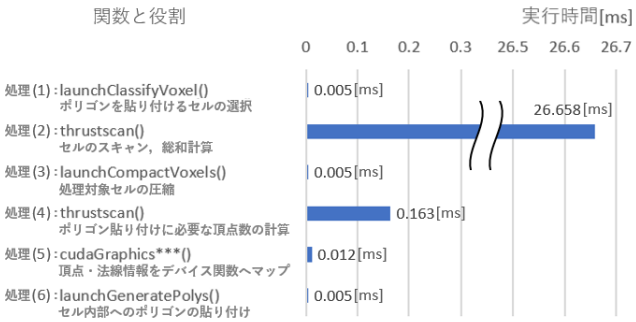

図 1　マーチングキューブ法の各実行時間．

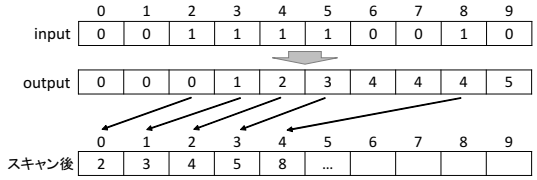

図 2　exclusive_scan とセルの集約．

$$output[i] = \sum_{j=0}^{i-1} input[j] \tag{1}$$

式 (1) で求めた結果は図 2 の output 配列として出力される．なお，input 配列において，1 はポリゴン面の元となる濃度面が存在しているセル，0 は存在していないセルを表す．出力された output 配列の末尾と input 配列の末尾を参照することでポリゴン面を有するセルの総和を求めることが可能となる．また，output 配列を参照することでポリゴン面を有するセルの抽出を行うことが可能になる．

3　GPU 上での exclusive_scan の CUDA 並列処理

本章では，exclusive_scan の独自プログラムを GPU 上で実現するために，CUDA による並列プログラムを作成した．カーネル関数内では，ブロック毎に exclusive_scan 処理を行った後，各ブロック内の総和を用いて全ブロックにまたがる exclusive_scan を求める．以下の節では，具体的なアルゴリズム及び GPU のブロック・スレッドの割り当てについて述べる．

3.1　exclusive_scan のブロック・スレッド割り当て

GPU 内の 1 ブロックにおける exclusive_scan の処理手順 [6] を図 3 に示す．各スレッドはステージ $d(0 \leqq d \leqq log_2 N)$ の値に基づいて配列要素の和を自身の要素に格納する．なお，データ数 N が $N = 1024$ の場合，$d = log_2 N = 10$ になる．

次に，GPU の全ブロックにまたがる exclusive_scan

†明治大学大学院 先端数理科学研究科 ネットワークデザイン専攻
Graduate School of Advanced Mathematical Sciences, Meiji University

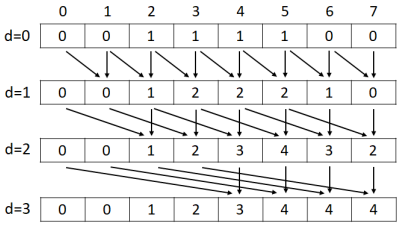

図 3　GPU の 1 ブロックにおける exclusive_scan 処理.

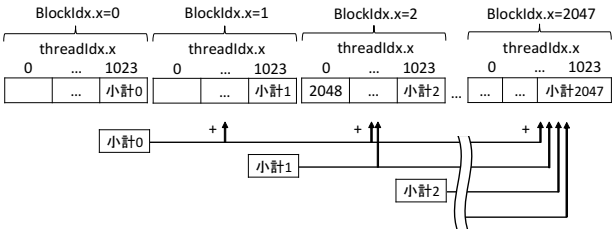

図 4　ブロック間の exclusive_scan 処理.

を図 4 のように求める．各ブロック内の最後の要素はブロック内要素の小計であり，この小計を後続のブロックの各要素に加算する．なお，加算した結果を格納する際，配列の添字を一つ後にずらすことで，exclusive_scan の全体処理が完了する．

マーチングキューブ法における exclusive_scan の input 配列は N_x, N_y, N_z 空間を用いる場合，$N_x * N_y * N_z$ 個の要素を持つ一次元配列となる．その際，GPU の 1 ブロックの計算を 1024 要素とし，各スレッドで並列処理を行う．また，$N_x * N_y * N_z/1024$ 個のブロックは GPU の MP(Maltiprocessor) に割り当てられて実行される．

3.2　シェアードメモリの活用

本手法ではさらなる高速化を実現するために，exclsusive_scan の各ステージの一時的なデータの格納の際に，各ブロック内に用意されている高速なシェアードメモリを利用してデータアクセス時間を短縮する．同様に，全ブロックにまたがった exclusive_scan の計算処理にもシェアードメモリを利用する．

4　GPU 上でのマーチングキューブ法の exclusive_scan 性能評価

本章では，GPU 上で行った exclusive_scan 処理の性能評価について述べる．

4.1　性能評価環境

性能評価に用いる PC は，GPU：NVIDIA GeForce GTX 1070，CPU：Intel Core i7-6700 3.4GHz *4 コア，メモリ：16GB，OS：Ubuntu16.04 LTS，CUDA の処理系：CUDA Toolkit 9.2 となっている．GeForce GTX 1070 は，15MP(Multiprocessor)*128=1920 CUDA コア，8GB のデバイスメモリ，48KB シェアードメモリの構成となっている．GPU 実行におけるスレッド数は 1024，ブロック数は配列の長さ 128^3=2048*1024 を考慮して 2048 に設定した．

4.2　GPU 上での exclusive_scan 処理の並列実行

本性能評価では，128*128*128 の空間を想定し，exclusive_scan の input 配列の長さを 128^3=2,097,152 とした．まず，マーチングキューブ法の実行における評価で

表 1　exclusive_scan の実行時間.

評価手法	thrust 実行	提案手法	実行時間短縮率
マーチングキューブ法	26.64[ms]	26.46[ms]	0.7[%]
単独実行	7.78[ms]	6.63[ms]	14.8[%]

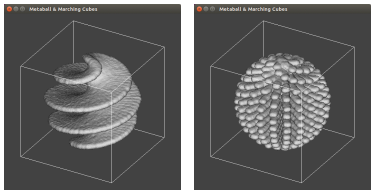

図 5　ポリゴンモデル出力結果.

は，空間に原点を中心とする YZ 平面上の円盤領域内に 100,000 個の粒子をランダムに配置し，等間隔で粒子群を分割した後，Z 軸座標値に比例した速度で Z 軸周りを等速回転させる処理 [5] が行われ，マーチングキューブ法を用いてポリゴンデータ生成をリアルタイムに行う．この出力を図 5 に示す．この処理における面を有するセルの抽出と総和の計算を行う際に用いられる exclusive_scan 関数の実行時間を測定した．実行結果は表 1 に示す．thrust ライブラリにおける実行時間は 26.64[ms]，提案手法による実行時間は 29.46[ms] であり，0.7% の実行時間の短縮となった．本プログラムでは exclusive_scan のみならず，同一 GPU 上で OpenGL を用いた描画処理を行っており，exclusive_scan が GPU を専有利用できていないため，処理時間の短縮効果が小さいと思われる．

次に，exclusive_scan 単体の性能を正確に測定するために，単体で性能評価を行った．その実行結果は表 1 の通りであり，thrust ライブラリで実行される exclusive_scan の実行時間は 7.78[ms] であり，一方提案手法による実行時間は 6.63[ms] であったため，14.8%の実行時間が短縮された．これにより，提案手法による exclusive_scan 関数は高い実効性能を達成できることが確認された．

5　おわりに

本稿では，3DCG の生成に用いられるマーチングキューブ法のための exclusive_scan の並列処理手法を提案した．本手法では GPU のブロック間及びブロック内のスレッド間の並列性を利用し，かつシェアードメモリを活用して実行時間の短縮を実現している．NVIDIA GeForce GTX 1070 上で行った性能評価の結果から，exclusive_scan の単体性能評価において thrust ライブラリより 14.8%の速度向上が得られており，提案手法の有効性が確認された．

参考文献

[1] William E. Lorensen,Harvey E. Cline. Marching cubes:A high resolution 3D surface construction algorithm. ACM SIGGRAPH,1987.

[2] Y.M.Xie,G.Y.Wang,T.T.Wong,P.A.Heng. Parallel visualization of multiple translucent isosurfaces. APCCAS,2008.

[3] Manuel Beniani,Mariagiovanna Sami,Danilo Pietro Pau. MRI Parallel Processing for Embedded Visualization. ICCC-Berlin,2013.

[4] Virasin Archirapatkave,Hendra Sumilo,Simon Chong Wee See. GPGPU Acceleration Algorithm for Medical Image Reconstruction. ISPA,2011.

[5] 乾 正知. GPU 並列図形処理入門. 技術評論社,2014.

[6] Mark Harris. Parallel Prefix Sum (Scan) with CUDA NVIDIA,https://www.mimuw.edu.pl/˜ps209291/kgkp/slides/scan.pdf,2007.