406 GPGPU を用いたセルラオートマトン法による簡易弾性体解析

Structural analysis of elastic body using Cellular Automaton and GPGPU

○学 白井 朋彦(早大)

正 宮下 朋之(早大)

Tomohiko SHIRAI, Waseda University

Tomoyuki MIYASHITA, Waseda University

Key Words: GPGPU, CUDA, CellularAutomaton, elastic deformation

1. 緒言

GPU(Graphics Processing Unit, 以下 GPU と呼ぶ)の画像処理能力の性能向上には目覚ましいものがある。使用頻度の向上に伴い、昨今のコンピュータには高いグラフィックス表示能力が求められるようになってきている。これにより GPU は極めて高い並列処理能力を獲得したため、CPU はようやく 4 コア、8 コアという現在に GPU は既に 128 個やそれ以上の計算処理ユニットを持つに至っている。単精度とはいえ現在 GPU の浮動小数点演算数(FLOPS)では CPU を大きく上回るに至った。また、従来の GPU では固定されたグラフィックスパイプライン処理しか行うことができなかったが、近年の GPU ではグラフィックスパイプラインの一部をプログラミングすることが可能になってきている。これにより GPU は単なるハードウェアからデジタル信号処理を専門に扱う DSP(Digital Signal Processor)的な性格を持つプロセッサに変貌しつつある。このように、高度な並列計算能力を持ち、CPU に迫るプログラマビリティを備える GPU を、画像描画という本来の目的ではなく、より一般的な計算用途に使用するという「GPGPU(General Purpose GPU)」の考え方が広まっており、動画エンコードソフト等で実際に使われる場面も出てきている。(2)(3)

弾性変形解析にはさまざまな方法があるが、その 1 つにセルラオートマトン(Cellular Automaton, 以下 CA)を用いたものがある(1)。この方法は、二次元解析ながら、パラメトリックな形状表現を必要とせず、設計において限定された設計仕様・情報のみが明らかである段階での解析に有効であり(1)、また解析のための新規のモデル作成を必要としない。本研究ではこのセルラオートマトン法による弾性体解析法(1)に米 NVIDIA 社の提供する GPGPU 環境である CUDA(2)(3)を用いて GPGPU を適用することにより、CA を用いた弾性体解析の高速化を目指す。

Fig.1 Image of sectional Cells

2. セルラオートマトン法について

セルラオートマトン法を用いて二次元弾性変形を求める方法について簡単に述べる。

図 1 のような格子状のセルを用いて弾性体断面を模し、座標を定めた上で、つりあいの式、応力と歪関係式、歪と変位関係式より導かれる変位の状態遷移式(1)を利用し解析を行う(1)。なお、式(1)において u , v は x 方向 y 方向の変位を、 X , Y は内力を表し、 $A=E/(1-\nu^2)$, $B=E\nu/(1-\nu^2)$, $C=E/2(1+\nu)$ と置換してある。

$$\left. \begin{aligned} u_{i,j} &= \frac{A}{2(A+C)}(u_{i+1,j} + u_{i-1,j}) + \frac{C}{2(A+C)}(u_{i,j+1} + u_{i,j-1}) \\ &\quad + \frac{B+C}{8(A+C)}(v_{i+1,j+1} - v_{i-1,j+1} - v_{i+1,j-1} + v_{i-1,j-1}) + \frac{h^2}{2(A+C)}\bar{X} \\ v_{i,j} &= \frac{C}{2(A+C)}(v_{i+1,j} + v_{i-1,j}) + \frac{A}{2(A+C)}(v_{i,j+1} + v_{i,j-1}) \\ &\quad + \frac{B+C}{8(A+C)}(u_{i+1,j+1} - u_{i-1,j+1} - u_{i+1,j-1} + u_{i-1,j-1}) + \frac{h^2}{2(A+C)}\bar{Y} \end{aligned} \right\} (1)$$

3. CUDA の適用

3-1 プログラム流れ

本研究では C 言語で記述したもの(以下 CPU プログラム)と、CUDA を使用したもの(以下 GPU プログラム)について解析速度・精度を比較していく。CUDA は C 言語の拡張使用であるので大まかなプログラムの流れは同じものとなる。図 2 にプログラムの流れを示す。図 2 中※1 印は CPU プログラムと GPU プログラムで異なる方法をとる部分である(後述)。※2, ※3 は CA による解析の終了条件を調べる部分である。すなわち、セル毎のその回の計算前後の状態値の差 δ を計算し、あらかじめ定めた閾値 ε と比較することで、 $\delta > \varepsilon$ であれば解析続行、 $\delta \leq \varepsilon$ であれば解析終了、としている。

3-2 プログラムの相違

CPU プログラムと GPU プログラムで CA の計算方法に異なる部分がある。以下に記述する。

CPU では一度に一つの処理を行うので、CPU プログラムではセル計算では端のセルから順に一つずつ計算していく「逐次計算処理」を行う。対して、並列処理の得意な GPU を用いる GPU プログラムでは複数のセル計算を同時に行う「並列計算処理」を行う。このイメージを図 3 に示す。ここで、GPU プログラムでは断面形状を模したセルを縦 2×横 2 のブロック毎に分割することで、ブロックの数だけ並列処理をしつつも計算の際の参照の重複を防いでいる。(図 2 中※1)

また、GPU プログラムではCPU からGPU, GPU からCPU ともに直接メモリを参照することはできないために、CPU-GPU 間のメモリコピーと回収を適宜行っている。これはCPU プログラムにはない処理であり、GPU プログラムにとっては時間的デメリットとなる。

Fig.2 Program stream

Fig.3 Difference between calculation methods

4. 解析
4-1 解析条件

CPU プログラムと GPU プログラムで解析を行い、解析速度の検証を行った。以下に解析条件を述べる。
図4に示すような片持柱に一様分布引張過重を作用させたモデルを使用する。

Fig.4 Analysis object

$P=5.0 \times 10^5$ [N/m]である。一辺の長さ x を 100~1000 セルのあいだで変化させ、CPU, GPU それぞれのプログラムの計算速度を比較する。また、今回 2 台の PC を用いて解析を行った。それぞれの PC, GPU の緒言を表 1~4 に示す。

Table1 PC1 (DIMENSION9100)	
OS	WindowsXP Professional
CPU	Pentium® D CPU 3.00GHz
Memory	1.00GB
ビデオカード	MSI N9500GT-MD512 D2 (PCIExp 512MB)
開発ツール	NVIDIA CUDA 2.0 Microsoft Visual C++ 2005 Express Edition

Table2 MSI N9500GT-MD512/D2	
GPU	NVIDIA GeForce™ 9500 GT
グラフィッククロック	550MHz
プロセッサクロック	1400MHz
ビデオメモリ	GDDR2 512MB
メモリクロック	800MHz
メモリーインタフェース幅	128bit
メモリ帯域	16.0GB/sec.
テクスチャフィル速度	8.8 Billion Texels/sec
バスインタフェース	PCI-E 2.0 16X

Table3 PC2 (PRECISION T7400)	
OS	WindowsXP Professionalx64
CPU	Xeon® X5472 3GHz
Memory	16.00GB
ビデオカード	NVIDIA Tesla™ C1060
開発ツール	NVIDIA CUDA 2.3 Microsoft Visual C++ 2008 Express Edition

Table4 NVIDIA TeslaC1060

GPU	NVIDIA Tesla™ C1060
プロセッサクロック	1.3GHz
ビデオメモリ	4GB
メモリクロック	800MHz
メモリインタフェース幅	512bit
メモリ帯域	102GB/sec.
バスインタフェース	PCI-Ex ×2

4-2 解析結果

解析結果を Mathematica を用いて図とし図 5, 図 6 に示す. また, それぞれの PC での解析時間を比較し図 7 に示す. 本プログラムは単精度浮動小数点 float により記述しているが, 図 5, 図 6 に表れているようにこの精度では形状について両者はまったく同じ結果を得ることができた. 図 6 を見るといづれの PC でも GPU プログラムが速いことがわかる. 傾向を見ると, セルが多く, プログラムに占める計算量が多くなるほど GPU プログラムが高速であることが分かる. また, 一般的な GPU である 9500GT を用いた GPU プログラムがワークステーション用 CPU である Xeon3.0GHz の CPU プログラムをも速度で大きく上回っていることがわかる.

Fig.5 Result(CPU)

Fig.6 Result(GPU)

Fig. 7 Time comparison

5. 結言

本研究では, セラオートマトン法による弾性体解析法に CUDA を用いて GPGPU を適用することにより, 解析速度を大きく向上させることができることを確認した. これより, 高価なコンピュータを用意せずとも GPGPU の効果を生かすことでより高速な演算解析が可能となることがわかる. また, GPU は倍精度小数点以上をフルサポートしていないという現状はあるが, 解析速度が高まることと新規のモデルを作成せずとも解析できることは将来的にリアルタイムでのシミュレーションにつながることを期待される.

参考文献

- (1) 宮田悟志, 登坂宣好, セラオートマトン法における弾性体解析法, 日本計算工学会論文集 vol.2004, 233-243(2004)
- (2) http://www.nvidia.com/object/cuda_home_new.html
- (3) http://www.nvidia.co.jp/object/cuda_home_new_jp.html