既存の並列化手法を用いたGPGPUプログラミングの提案

大島聡史[†,††] 平澤将一[†,††] 本多弘樹[†,††]

GPUの性能向上に伴い，GPUの性能を様々な用途に活用するGPGPUが注目されている．GPGPUは特に並列プログラムにおいてCPUを超える高い性能が期待される一方，GPGPUプログラミング特有の手法を用いる必要があるためソフトウェアの作成が容易ではない．本稿ではGPGPUプログラミングを容易にする１つの手法として既存の並列化手法を利用することを提案する．また具体的な実装に向けて，近年利用され始めたGPGPU向けプログラミング言語CUDAを利用し，GPU上の処理をOpenMPやMPIで記述する方法を検討する．

Proposal of GPGPU Programming Using Existing Parallelizing Method

Satoshi OHSHIMA,[†,††] Shoichi HIRASAWA[†,††] and Hiroki HONDA[†,††]

GPGPU utilizing GPU's performance for general-purpose computation is attracting much attention. GPGPU is expected to effect higher performance than CPU. However, creating GPGPU programming is not easy because programming methods peculiar to GPGPU programming are needed. In this paper, we propose to use existing parallellizing method as one of a new method making GPGPU programming easier. Also we consider writing programs running on GPUs with OpenMP and MPI based on the new GPGPU programming language of CUDA.

1. はじめに

近年，高度な画像処理の要求に伴いGPU(Graphics Processing Unit)の性能が著しく向上している．GPUはCPU(Central Processing Unit)と比べて並列処理やベクトル処理に適したハードウェア構成であることから，GPUを汎用演算に利用するGPGPU(General-Purpose computation using GPUs)[1]への注目が高まっている．

従来のGPGPUプログラミングにおいては，グラフィックスAPIとシェーダ言語を用いたグラフィックスプログラミングの技術が必要とされてきた．こうしたプログラム作成手法はGPGPUプログラミングに特有のものであるため，他の分野のユーザにとってGPGPUの活用は容易ではない．現在ではCUDA[2]のようにGPGPUプログラミングの特殊性を隠蔽するプログラミング言語なども登場しているが，GPUを活用するにはGPUアーキテクチャの理解と新しい言語の習得が必要である．更に，GPUは並列処理に適したハードウェアであるが，既存の並列化手法を容易に利用することはできない．

そこで本稿では，既存の並列化手法を用いたGPGPUプログラミングを提案する．GPUは並列性の高いプログラムの実行に適しているため，既存の並列化手法を容易にGPUへ適用する手段があれば，GPUを並列プログラムの実行環境として有効活用可能となることが期待できる．本提案は複数のCPU(複数コアのCPU)や複数のGPUを搭載したPC環境，これらのPCを複数台用いた計算環境など多様な計算環境への適用も視野に入れている．しかしながら，本稿ではその第一歩として単一GPU上の処理を既存の並列化手法を用いて記述することを中心に話を進める．

2. GPUのアーキテクチャと既存のGPGPUプログラミング手法

GPUは本来，高速に画像描画を行うために発展したハードウェアである．現在では動画再生支援などの処理も統合される傾向にあるため，性能の差異は大きいものの，一般的なPCの多くに搭載されている．

† 電気通信大学 大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications
†† 独立行政法人科学技術振興機構，CREST
Japan Science and Technology Agency, CREST

図 1 に伝統的な GPU のハードウェア構成と画像描画のための主な処理の対応を示す．GPU が行う画像描画のための主な処理は，並列計算やベクトル計算による高速化に適している．そのため GPU 上の処理ユニットは並列化が進んでおり，ベクトル計算に適した内部構造となっている．また高度な画像表現には描画内容に応じた複雑な計算が必要なため，処理ユニットのプログラマブル化が進んでいる．画像描画処理全体の記述に DirectX や OpenGL といったグラフィックス API が用いられるのに対し，計算ユニットの処理の記述には HLSL，GLSL，Cg といった専用のシェーダ言語が用いられ，実行時に動的に読み込んで (切り替えて) 利用されている．

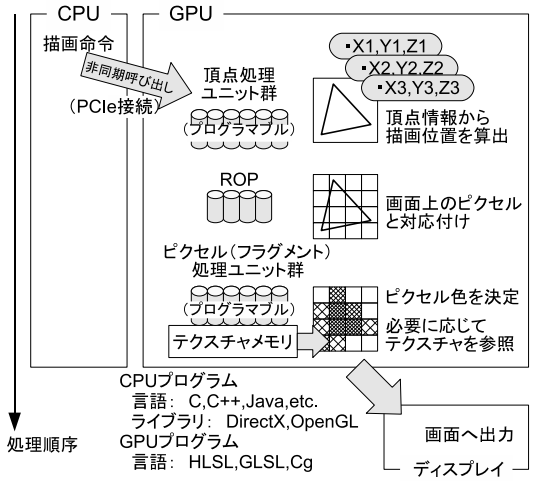

図 1 伝統的な GPU のハードウェア構成と画像描画処理の対応
Fig. 1 Hardware construction of traditional GPU and relationship with graphics processing

GPGPU は，GPU の特徴を利用して様々な処理 (汎用計算，General-Purpose computation) を行うものである．GPU のハードウェア性能を活かせる用途，すなわち並列計算やベクトル計算に適した対象問題に対しては，CPU を圧倒する演算性能を得ることができる．しかしながら，GPGPU アプリケーションの実装は容易ではない．例えば GPU を用いて数値計算を行うためには，数値計算アルゴリズムを画像描画のシステムに対応させて実装する必要がある．そのためには数値計算プログラミングとグラフィックスプログラミングの両方に精通している必要があり，高い性能を得るためには並列化手法や GPU アーキテクチャについての知識も重要である．CUDA を用いることができればグラフィックスプログラミングの知識は不要となるが，GPU アーキテクチャの知識が必要であり，また既存の並列計算やベクトル計算向けの手法を利用することも容易ではない．

3. 提 案 内 容

3.1 既存の並列化方式を利用した GPGPU プログラミングの提案

GPU は CPU と比べて高いハードウェアレベルの並列性を備えているため，GPU による演算性能の向上が期待されているのはもっぱら高い並列性を持つアプリケーションである．一方で並列化は SMP，PC クラスタ，Grid など既に様々な研究が進められているテーマであり，現在もマルチコア CPU の普及などによって注目されている．既存の並列化方式を用いて GPGPU プログラミングを行えるようにすることができれば，GPU を既存の並列環境により近いものとして扱えるようになると考えられる．GPU が身近で使いやすいものとなることで，多くのユーザが GPU の持つ高い計算性能を有効に活用できるようになることが期待できる．

そこで，GPGPU プログラミングにおいて既存の並列化方式を利用することを提案する．本章の残りの部分では，本提案に対する実装の例として GPU 向け OpenMP および GPU 向け MPI の検討を行う．更に，CUDA を用いた場合の実装についての検討を行う．ただし，今回は簡単のため単一 GPU 上の動作を OpenMP や MPI で記述するという最低限の実装に範囲を絞る．

現在，効率良く GPGPU プログラミングを行うための言語やライブラリの研究がいくつか行われている．中には BrookGPU[3] や RapidMind[4] のように，GPU のみではなく Cell Broadband Engine やマルチコア CPU などへの適用を視野に入れたものも存在する．これらは新しい言語やライブラリを作成しているという点で本研究とは異なるアプローチを採っているが，GPGPU プログラミングを容易にするという意味では本研究に近いものであると言える．

3.2 GPU 向け OpenMP および GPU 向け MPI の検討

最新 GPU の構造を概観し，OpenMP や MPI を GPU に適用するにはどうすればよいか，OpenMP および MPI の実行モデルと GPU ハードウェアをどのように関連付けるかについての検討を行う．

本稿執筆時点の最新世代 GPU である GeForce 8000(G8x) シリーズおよび Radeon HD 2000(R6xxx) シリーズではプログラマブル処理ユニットの並列化が進んでおり，GPU 全体で最大 100 以上の演算を並列実行可能となっている[2),5)]．ただし，GPU 上の全て

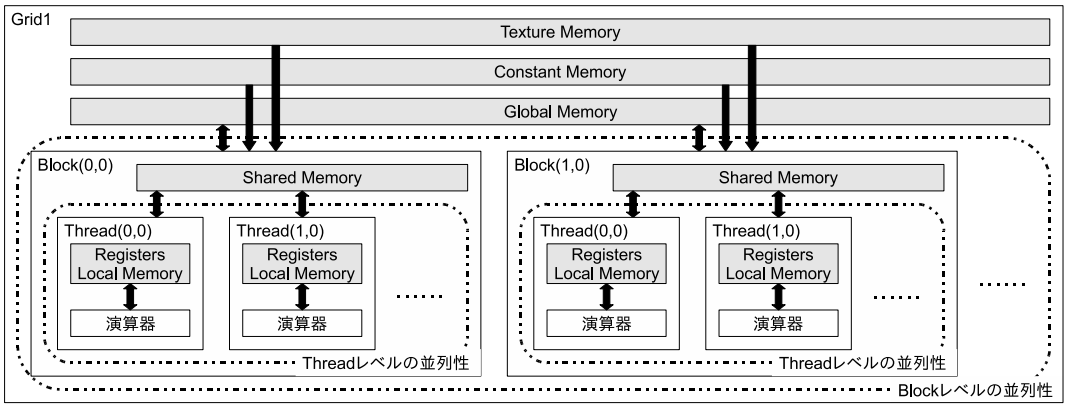

図 2 CUDA の並列実行モデルとメモリモデル
Fig. 2 Parallel execution model and memory model of CUDA

の演算器が同じ機能を持ち均等に配置されているわけではない．いくつかの演算器が集まって処理ユニットを構成しており，更に処理ユニットが集まって GPU を構成している．またメモリについても，GPU 全体から透過的にアクセス可能な (グローバルな) メモリのみを備えているわけではなく，処理ユニットごとにローカルなメモリも備えている．すなわち現在の GPU は，ハードウェアレベルでの高い並列性を備えているのみならず，搭載している演算器とメモリには階層性が備わっていると捉えることができる．

一方，OpenMP と MPI はそれぞれ共有メモリ型並列計算機および分散メモリ型並列計算機で多く用いられている並列化手法 (並列化ライブラリ) である．

OpenMP は逐次プログラムのソースコードにプラグマを挿入し，コンパイル時にプラグマを解釈して共有メモリを用いたスレッド並列プログラムに変換する．逐次プログラムの段階的な並列化にも適している．

MPI はメッセージ通信を利用し，複数ノード間におけるデータや制御のやり取りを行うことで並列プログラムを形成する．

OpenMP はループ変数の自動書き換えによるループの並列化といった細粒度の並列化に使われることが多く，MPI はノードごとに取得した rank 情報に基づいて処理を振り分けるとった疎粒度の並列化に使われることが多い．更に，SMP やマルチコア CPU を搭載したノードによって構成される PC クラスタなどにおいては，ノード内の並列化を OpenMP，ノード間の並列化を MPI によって行うハイブリッドな並列化も行われている．

以上から，GPU 向け OpenMP および GPU 向け MPI について次のような対応付けを考える．まず，処理ユニット内の処理を OpenMP の細粒度な並列処理に対応付け，ループの並列化などに利用する．更に，処理ユニット間のデータのやり取りを MPI の疎粒度な並列処理に対応付ける．これにより，GPU 内部の演算器やメモリが持つ階層性を活用した GPU 向け並列プログラムを容易に作成可能となることが期待できる．

3.3 CUDA を用いた GPU 向け OpenMP および GPU 向け MPI の検討

これまで GPU の機能を利用するにはグラフィックス API を介する必要があったため，全ての処理をグラフィックスプログラミングの方式にあわせて実装する必要があった．これに対して CUDA では多くの処理をより直感的に記述することができる．またこれまで GPU の内部情報についてはグラフィックスプログラミングに必要な程度の情報のみが公開されていたが，CUDA とともに多くの情報が提供されるようになった．これにより GPU プログラムのデバッグや最適化に有益な情報が入手しやすくなったと言える．そこで，CUDA を用いた GPU 向け OpenMP および GPU 向け MPI の実装を検討する．

OpenMP を用いた並列処理には各 CPU が自由にアクセスすることのできる共有メモリが，また MPI を用いた並列処理にはノード間の通信が不可欠である．そのため CUDA を用いて提案を実装するためには，これらに対応する機能が CUDA に備わっている，もしくは CUDA 上で模倣できる必要がある．図 2 に CUDA の並列実行モデルおよびメモリモデルを示す．これを元に，CUDA を用いた GPU 向け OpenMP および GPU 向け MPI の実装を検討する．

まずは，GPU 向け OpenMP の実装について考える．CUDA には共有メモリとして利用可能な複数種類のメモリが存在する．Global Memory, Constant Memory, Texture Memory は全ての Block および全ての Thread から共有メモリとして利用可能であり，Shared Memory は同一 Block 内の全ての Thread か

ら共有メモリとして利用可能となっている．これらのうち，Constant Memory と Texture Memory は GPU から見ると読み込み専用のメモリであるため，今回は利用対象から除外する．残る Global Memory と Shared Memory については，Global Memory を利用すれば Thread レベルと Block レベル双方で，Shared Memory を利用すれば Block レベルで OpenMP の並列処理を置き換えられると考えられる．ただし，Shared Memory と Global Memory では Shared Memory の方が高速にアクセスできるため，並列化の粒度が小さい OpenMP では Thread レベルの並列処理と対応付けたほうが良い性能を得やすい可能性が高い．

続いて GPU 向け MPI の実装について考える．CUDA の並列実行モデルのうち，Thread レベルの並列処理は SIMD に近く，異種の演算を並列実行することができない．これは疎粒度の並列処理を行う MPI において致命的な問題であるため，Block レベルの並列処理を利用することにする．Block 間の通信については，Global Memory を利用して Block 間の通信を模倣する．なお，CUDA には Block 間の同期を取る機構が用意されていないため，Global Memory をうまく利用して同期処理を模倣する必要がある．

以上のように Thread レベルの並列化と Block レベルの並列化を適切に使い分けることで，CUDA を用いて OpenMP と MPI を模倣できる可能性を示すことができた．更に，Thread レベルの並列化を用いた OpenMP と Block レベルの並列化を用いた MPI を併用することで，既存の OpenMP と MPI を用いたハイブリッド並列プログラムを模倣することも可能となる．図 3 に我々が検討を進めている CUDA を用いたハイブリッド並列プログラムを既存の OpenMP と MPI によるハイブリッド並列プログラムと対応付けて示す．これにより GPU の性能をより効率良く発揮できることが期待できる．

CUDA は GeForce8000 シリーズのアーキテクチャに強く依存したライブラリである．CUDA は今後 NVIDIA のリリーする新しい GPU でも利用できるとされているが，今のところ他社の GPU で用いることはできないため，GPU プログラムをすべて CUDA で記述するというのは現実的ではない．これに対し，NVIDIA と GPU のシェアを二分している AMD も Close-to-the-Metal(CTM) の提供を表明している．GPU プログラムの記述しやすさや性能の発揮しやすさの面から，今後も GPU ベンダーが GPGPU 向けに開発環境や資料を提供し続ける可能性は高い．一方で，CUDA や CTM が言語仕様の変更なく継続して提供されるかは不明であり，また GPU の種類ごとに個別のプログラムを作成する必要があることは，GPGPU プログラミングを容易にするという観点からは大きな障害である．本提案は GPU の違いを吸収し共通に利用できる環境を提供するものであり，こうした問題を解決できる可能性があると言える．

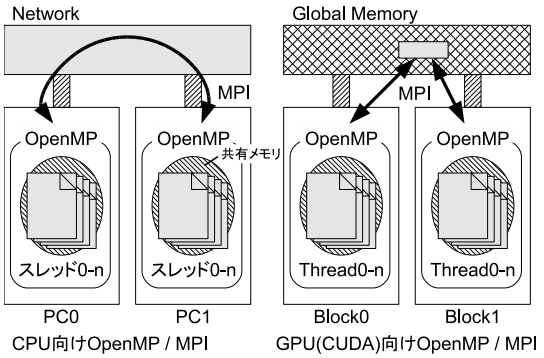

図 3 CUDA を用いた OpenMP と MPI
Fig. 3 OpenMP and MPI using CUDA

4. おわりに

本稿では GPGPU を容易に利用するための新しい手法として，既存の並列化手法を用いた GPGPU プログラミングの提案を行った．本提案は，今後ますますの性能向上が期待される GPU を容易に活用する新しい可能性を示すものである．

今後は本提案の内容を実装し，性能評価を行う．更に現時点では実行対象を単一の GPU のみに絞っているが，CPU と GPU を含めた PC 全体など様々な環境に対象を広げる．またこれらを通して GPGPU プログラミングに OpenMP や MPI といった既存の並列化手法を用いること自体についても評価を行うことで，より良い GPGPU プログラミング手法についての議論が深まることが期待できる．

参考文献

1) gpgpu.org: General-Purpose computation on GPUs(GPGPU), http://gpgpu.org/.
2) NVIDIA: CUDA Programming Guide 1.0 (CUDA NVIDIA Homepage), http://developer.nvidia.com/cuda/.
3) BrookGPU: BrookGPU, http://graphics.stanford.edu/projects/brookgpu/.
4) RapidMind Inc.: RapidMind, http://www.rapidmind.net/.
5) gpgpu.org: SIGGRAPH 2007 GPGPU COURSE, http://www.gpgpu.org/s2007/.