GPGPUにおけるデータ自動転送化コンパイラの提案

道浦 悌[†] 大野和彦[†]
佐々木敬泰[†] 近藤利夫[†]

1. はじめに

近年 GPU は CPU に比べ性能向上がめざましく，ムーアの法則をしのぐ性能向上を見せている．そのため，GPU に汎用的な演算を行わせる GPGPU への関心が高まっており，コアの最適な利用法などの研究が活発になっている[1]．また，CUDA[2] や OpenCL[3] などで GPU プログラミングがサポートされており GPGPU への敷居は低くなってきている．

しかし，CUDA や OpenCL などを用いてもハードウェアの構成上，ユーザが明示的にホストメモリ (CPU 側)・デバイスメモリ (GPU 側) 間のデータ転送を行う必要がある．このため，他のプログラミングに比べプログラマの負担が大きい．そこで，ホストメモリ・デバイスメモリ間のデータの自動転送化コンパイラを提案し，GPU プログラミングの負担軽減を目指す．

2. コンパイラの機能

簡単な変数宣言を行うだけで指定した変数がホストメモリ・デバイスメモリ間で自動転送されるコンパイラを提案する．ユーザは GPU カーネルで使用する変数に対し転送の形式を指定するだけでよい．

2.1 機能概要

GPGPU において GPU コードの実行単位であるカーネルを実行するためには，カーネルで使用するデータの転送が完了している必要がある．そのため，効率的にデータ転送を行い，データ転送完了待ちで GPU がカーネルを実行していない時間をなるべく短くしなければならない．しかし，多数のカーネルが様々なタイミングで実行されるため高効率なデータ転送を行うのは困難である．加えて，用いる GPU デバイスによってデバイスメモリの容量も異なるため，効率的なデータ転送は GPU デバイスごとにタイミングを考えなければならず，コードの移植性にも問題がある．

そこで，本コンパイラではデータ転送命令を自動的に生成することで，ユーザのコーディングを簡単にする．さらに，GPU デバイスに応じて転送を最適化することで，自動的に効率のよいコードを得ることができる．また，ホスト側の変数とデバイス側の変数を用意し，どちらで実行されるコードかによって変数を使い分ける必要がある．本コンパイラではホスト側，デバイス側の両コードで同じ変数をグローバル変数のように扱うことができる．これにより，ホスト側の変数とデバイス側の変数の管理の負担が低減される．

```
__download__ int temp;
__readback__ int res;
__share__ int array[256];
```

図 1 宣言方法

2.2 転送の形式

GPGPU におけるデータ転送は 2 種類ある．ホストメモリからデバイスメモリへの転送である download 転送と，デバイスメモリからホストメモリへの転送である readback 転送である．本コンパイラは転送の形式として download 転送の自動化を行う download 形式，readback 転送の自動化を行う readback 形式，どちらの性質も持つ share 形式の 3 つの形式を提供する．

宣言方法は download 形式，readback 形式，share 形式それぞれ変数宣言時に変数の前に`__download__`，`__readback__`，`__share__`とつけるだけでよい．図 1 に宣言方法の例を示す．

3. コンパイラの設計

3.1 変数管理

download 形式，readback 形式，share 形式で宣言された変数を変数表に登録し，管理を行う．また，各変数がどのカーネルで使用されているか解析を行う．この情報を元に各カーネル実行に必要な変数を管理し，データ転送のタイミングを決める．ユーザの記述した変数をホスト側で使用する変数とデバイス側で使用する変数に拡張し，それぞれでメモリを確保し，形式に応じてこの両変数間の転送を自動で行う．

† 三重大学大学院工学研究科情報工学専攻

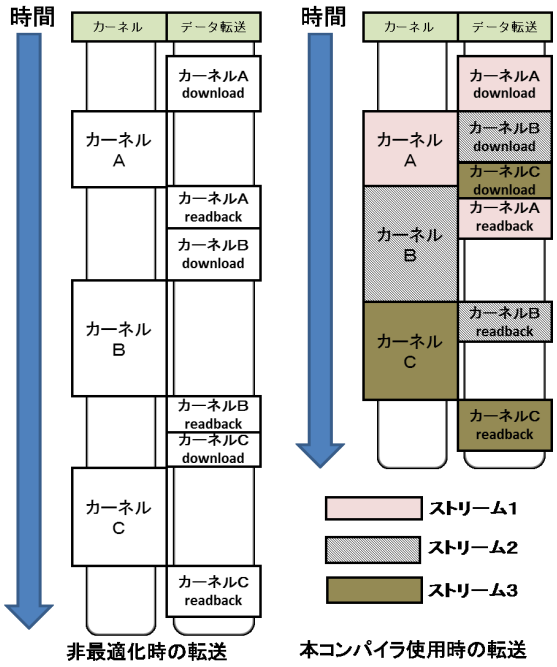

図 2　データ転送の流れ

3.2　データ自動転送化

3 つのカーネルを実行する場合を例に説明する．図 2 にデータ転送の流れを示す．

カーネルの実行状況が複雑な場合，ひとつのカーネルに対し単純に download 転送，カーネル実行，readback 転送を一連の動作としないとデータ転送の管理が煩雑となる．しかし，その場合，図 2 左に示すとおりカーネル未実行の期間が長く，オーバヘッドが発生する．従って，データ転送最適化を行わないとオーバヘッドが増大し，GPU で処理を行わせる利点が失われる可能性がある．しかし，データ転送最適化を行うためには，GPU のアーキテクチャの理解，GPU 固有のスペックなどに合わせた最適化が必要である．

本コンパイラでは，図 2 右に示すようにデータ転送とカーネルの実行を同時に処理するようにコードを生成することで，オーバヘッドを低減する．この処理のオーバラップは CUDA 既存の機能であるストリームを用いて実現する．ストリームとは関連性のあるデータとカーネルを結びつけるためのもので，各ストリーム上ではデータ転送，カーネル実行がそれぞれ登録順に実行されていく．また，異なるストリームに属している場合，カーネル実行とデータ転送が同時に実行可能であり，図 2 右に示すようなデータ転送のオーバラップが可能となる．本コンパイラは，このストリームを用いるために，各カーネルが使用するデータを静的解析し，データとカーネルの関係を明らかにする．関連のあるデータとカーネルを一つのストリームでまとめ，依存関係のないデータやカーネルは別ストリームとしてまとめる．それぞれのストリームごとにデータ転送命令を挿入することで自動転送を実現する．また，デバイスメモリの容量などを参照して最適化を行うことで，自動で GPU のスペックに合わせた最適化をプログラムの書き換えなしに行うことができる．

表 1　実行時間 (sec)

デバイス	想定コード	最適コード	非最適コード
TeslaC2050	13.50	13.46	18.51
TeslaC1060	35.81	35.79	39.10

4. 予 備 実 験

本コンパイラの有用性を示すために想定される出力コードを手動で作成し実行時間を計測した．マッチングによる暗号解読のプログラムについて，想定出力コード，手動で最適化したコード，非最適化コードの 3 種類を用意し，2 種類の GPU デバイス上で実行時間の比較を行った．数字 5 桁の暗号を 448 万個解読した場合の実行時間を表 1 に示す．

結果から本コンパイラを用いることにより，手動で最適化したコードとほとんど差がなく，非最適化コードよりも高速で実行可能であることがわかる．従って，本コンパイラを用いると低負担で GPU プログラミングが可能で，かつ高速に実行できる．

5. お わ り に

本論文では GPGPU におけるデータの自動転送化コンパイラの構造と想定される性能を示した．今後は本コンパイラの構成の妥当性の評価を行う．その後，明示的なデータ転送を行わなくてすむ CUDA コードから転送命令が挿入された CUDA コードを出力するコンパイラを実装する．また，最適コードとの比較を行い実行性能について評価する．さらなる展望として，OpenCL への対応，最適なメモリ階層への割り当ての自動化などが挙げられる．

参 考 文 献

1) YiYang,PingXiang,JingfeiKong,HuiyangZhou,A GPGPU Compiler for Memory Optimization and Parallelism Management
2) nVidia CUDA Zone, http://www.nvidiaco.jp/object/cuda_home_new_jp.html
3) OpneCL,http://www.khronos.org/opencl/