シェアードメモリを利用したGPGPU処理系の自動最適化機構

丸山剛寛[†] 神谷智晴[†]
松本真樹[†,††] 大野和彦[†]

1. はじめに

近年，CPUと比べ性能向上のめざましいGPUに汎用的な計算を行わせるGPGPUへの関心が高まっている．そこで，我々はデータ転送を自動化するフレームワークMESI-CUDA[1)]を開発している．本研究ではMESI-CUDA上に，シェアードメモリを利用する自動最適化機構を設計・実装した．

2. 背景

MESI-CUDAはホスト・デバイス間のデータ転送やメモリ確保・解放，ストリーム生成・破棄のコードを自動的に生成することで，ユーザーのCUDAプログラミング開発の負担を軽減する．図**1**と図**2**にそれぞれCUDAとMESI-CUDAのメモリモデルを示す．ホストとデバイスへの処理の振り分けやカーネル関数の記述はユーザ自身が従来のCUDAに準じる形でコーディングを行う．MESI-CUDAでは，データ転送やカーネル処理のスケジューリングを自動的に行う．そのために，図2のような仮想共有メモリ環境のプログラミングモデルを採用し，ホスト・デバイス両方よりアクセス可能な共有変数を提供する．

現状のMESI-CUDAはグローバルメモリのみを使用するCUDAコードを生成しており，手動でメモリ階層を最適化したCUDAプログラムと比較すると実行速度が遅いという問題がある．

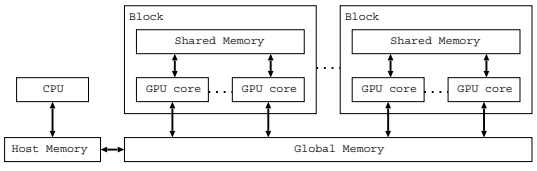

図**1** CUDAのメモリモデル

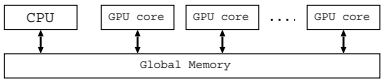

図**2** MESI-CUDAのメモリモデル

3. 自動最適化機構

3.1 概要

前述の問題を解決するためにシェアードメモリを使用するCUDAコードの自動生成を考える．シェアードメモリは，ブロック毎に存在し，ブロック内の各GPUコアで実行されるスレッドからアクセスすることができる．通常使用するグローバルメモリと比べ約百倍高速にアクセスできるため，メモリアクセスのレイテンシが小さい．よって，使用するデータをシェアードメモリに格納することでCUDAプログラムの高速化が可能となる[2)]．しかし，容量が64KBと非常に小さく，プログラム中で使用する全ての配列を格納することは困難であるため，効率的に用いるには使用頻度の高いデータを格納する必要がある．

本機構は，配列のアクセスされるインデックスの解析を行い，ブロック内の使用頻度が高い配列の区間を検出し，その配列についてシェアードメモリを使用するCUDAコードを自動生成する．

3.2 解析

今回解析の対象としたMESI-CUDAプログラムは，1次元のグリッド，ブロックで一重ループ中の1次元配列を扱うプログラムである．対象とするプログラムの簡単な例とそのアクセスの様子を図**3**〜図**6**に示す．これらのプログラムをk個のスレッドで実行した時，ブロック内のアクセス範囲は一致するが，図3の方がブロック内での使用頻度が高い．従来の範囲解析ではアクセスするかどうかは解析できる[3)]が，使用頻度を解析することはできない．

そこで，ブロック内のアクセス回数をブロック内のアクセス範囲の大きさで割ることで，配列の要素あたりの平均アクセス回数を求める．この値が1より大きい場合，ブロック内の各スレッドからのアクセス範囲

† 三重大学大学院工学研究科情報工学専攻
†† 現在，株式会社 医用工学研究所

```
for (i=0;i<4*k;i++)
 A[threadIdx.x]+=B[i];
```

図 3　アクセス範囲が重なる例

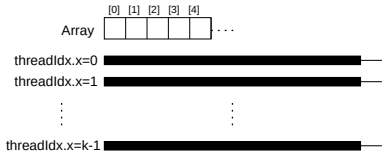

図 4　図 3 のアクセスの様子

```
for (i=0;i<4;i++)
 A[threadIdx.x]+=B[4*threadIdx.x+i];
```

図 5　アクセス範囲が重ならない例

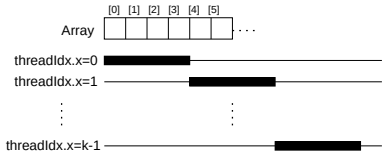

図 6　図 5 のアクセスの様子

が重複していると推測できる．図 3 ではブロック内アクセス回数は 4*k*k 回でアクセス範囲は 4*k であるので平均アクセス回数は k 回，図 **5** ではブロック内アクセス回数は 4*k 回でアクセス範囲は 4*k であるので平均アクセス回数は 1 回という解析結果が得られ，これは実際のコードの挙動と一致している．

3.3　コード生成

前述の解析方法で検出した配列に対し下記のコードを生成するように実装を行った．

- シェアードメモリの領域確保
- グローバルメモリからシェアードメモリへデータのコピー
- グローバルメモリをアクセスする元のコードからシェアードメモリをアクセスするコードに変換，それに伴うインデックスの変換
- シェアードメモリからグローバルメモリへデータのコピー

4. 評　　価

実装した自動最適化機構の有用性を示すために，本機構による最適化の有無による CUDA プログラムの実行時間の比較を行った．評価環境は Core i7 930 2.80GHz，メモリ 6GB，TeslaC2050 を搭載した計算機を使用した．評価には行列積と逆行列を求めるプログラムを用いた．正方行列の一辺の大きさが 256，512，1024，2048 の場合の実行時間を測定した．結果を表 **1** に示す．

表 **1**　実行時間 (秒)

	行列積		逆行列	
一辺の大きさ	非最適化	最適化	非最適化	最適化
256	0.028	0.028	1.520	0.969
512	1.437	1.346	6.434	3.042
1024	5.708	2.956	44.730	20.190
2048	23.632	7.551	120.751	53.194

表 1 からわかるように，一辺の大きさが 2048 の場合に両方のプログラムで実行時間が約 1/3 まで短縮されている．これは，本機構によって適切な配列のアクセス解析がされており，これによって前述したシェアードメモリを効果的に用いることが可能になりメモリアクセスのレイテンシが短縮されたためである．

5. おわりに

本研究では MESI-CUDA 上に，シェアードメモリを利用する自動最適化機構を設計・実装し，評価を行った．その結果，本機構を用いることで適切な配列のアクセス解析が行われ，シェアードメモリを利用する CUDA コードが自動生成できた．

今後の課題として，本研究では簡単な配列のアクセスにのみ対応しているが，より複雑な場合に対応していく必要がある．また，格納する配列を検出する方法を簡単な式で判別しているが，より精度の高い方法を導入する必要がある．

謝　　辞

本研究の一部は日本学術振興会科研費・基盤研究 (C)（課題番号 24500060）による．

参 考 文 献

1) 道浦悌, 大野和彦, 松本真樹, 佐々木敬泰, 近藤利夫: GPGPU におけるデータ転送を自動化する MESI-CUDA の提案, In 先進的計算基盤システムシンポジウム SACSIS2012, pp. 201–209 (2012).
2) Ryoo, S., Rodrigues, C. I., Baghsorkhi, S. S., Stone, S. S., Kirk, D. B. and mei W. Hwu, W.: Optimization Principlcs and Application Performance Evaluation of a Multithreaded GPU Using CUDA, *In Principles and Practices of Parallel Programming*.
3) Yang, Y., Xiang, P., Kong, J. and Zhou, H.: A GPGPU Compiler for Memory Optimization and Parallelism Management, *In Programming language design and implementation*.