高効率な資源利用を目指したGPUプログラム自動チューニング手法

竹嶌 良*，小野 和馬，津邑 公暁（名古屋工業大学）

Code Auto-tuning for efficiently utilizing GPU Resources

Ryo Takeshima, Kazuma Ono, Tomoaki Tsumura (Nagoya Institute of Technology)

1. はじめに

GPU（Graphics Processing Unit）は画像処理専用のプロセッサであり，データ並列性の高い処理に対し，少ない消費電力で高い性能を達成できる．この GPU の高い演算能力が注目され，GPU を汎用計算に応用する GPGPU（General Purpose Computing on GPU）が発展してきている．この GPU 向けの開発環境の中でも最も広く普及しているのが，NVIDIA が提供している並列計算アーキテクチャモデル CUDA（Compute Unified Device Architecture) の開発環境である．しかし CUDA において高い実行効率を得るためには，GPU に対する深い理解が必要となる．そこで本稿では GPU プログラムに対し自動的にチューニングを施す手法を提案する．

2. 研究背景

CUDA のプログラミングモデルでは，GPU が持つ多数のプロセッサを用いて，最小の実行単位である Thread を大量に，かつ並列に実行することで高い演算性能を実現する．CUDA では，これら大量の Thread は階層的に管理される．なお，Thread の集合を Block，Block の集合を Grid と呼び，Grid 内の Block の次元や Block 数，および Block 内の Thread 数を**実行構成**と呼ぶ．この実行構成は，GPU に実行させる処理を定義した関数（kernel 関数）の呼び出しとともに記述される．ただし，高い実行効率を得るためにはこの実行構成を適切に設定する必要があり，その設定には GPU に対する深い理解が求められる．また GPU は複数のメモリを搭載しており，それぞれのメモリは異なる特徴を持つ．そのため，プログラマは処理対象データの特徴に応じて，配置先のメモリを適切に選択する必要がある．このような背景から，プログラマをサポートするための研究として C-to-CUDA コンパイラ[(1)] がいくつか提案されている．

3. 自動チューニング機構の設計

CUDA には，GPU の実行効率の高さを表す指標として**占有率**が定義されている．この占有率は実行構成を基に算出されるが，kernel 関数や GPU のハードウェア構成によって，複雑に変化してしまう．そのため，最も占有率が高くなる実行構成を求めることは困難である．そこで本稿では，GPU プログラムに対し自動的にチューニングを施し，最も高い占有率を導く実行構成を設定する手法を提案する．本提案手法では，高占有率を導く実行構成を算出するために，プログラムの構文解析や，API の実行などにより必要なパラメータを自動的に取得する．その後，それらのパラメータを基に占有率を計算し，実行構成を決定する．そして決定した実行構成を用いるようプログラムを改変する．加えて本提案手法では，GPU の複雑なメモリ構成を隠蔽するために，各データの転送先メモリを適切に選択する．そのために，実行構成決定のための構文解析において，転送が必要なデータを併せて検出し，さらに転送先メモリを選択するための情報も取得する．そしてこの取得情報を基に，各データの適切な転送先メモリを決定する．最後に，データの転送ための API をプログラム中に挿入する．ただし，実行構成自体がプログラムのセマンティクスと密接に関わっている kernel 関数には，実行構成を変化させる本提案手法を適用することができないため，実行構成のチューニング対象とする関数を，プラグマを用いることでプログラマに指定させる．

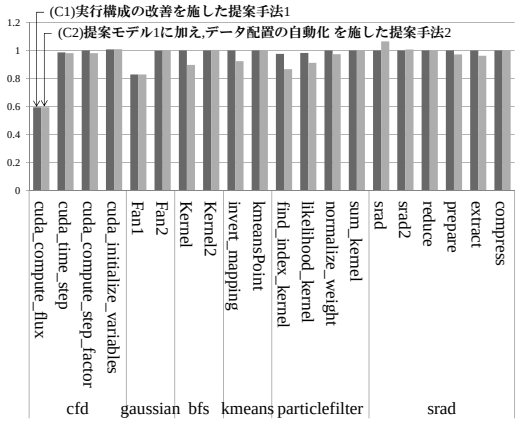

Fig.1　Evaluation Result

4. 評価

提案手法の有効性を検証するために，提案手法を適用したプログラムとハンドコーディングによりチューニングしたプログラムの実行時間を比較評価した．ベンチマークプログラムには，Rodinia Benchmark suite から，6 つのプログラムを使用した．評価結果を Fig.1 に示す．図の縦軸は，各ベンチマークプログラムにおける kernel 関数ごとの実行時間を表しており，ハンドチューニングしたプログラムの実行時間を 1 として正規化している．高占有率を導く実行構成の設定および，メモリの適切な利用により，ハンドチューニングしたプログラムと比較して，最大 40.5%，平均 5.11% の速度向上を確認した．

文　献

(1) Ventroux N.et.al.: "SESAM/Par4All: A Tool for Joint Exploration of MPSoC Architectures and Dynamic Dataflow Code Generation" *Proc. 12th RAPIDO* pp.9–16(2012)