ISS-P-136

GPGPU プログラムにおけるプログラミング容易化の検討

箱田　雄太†　　佐藤　寿倫††
† 福岡大学大学院工学研究科　　†† 福岡大学工学部

1. はじめに

GPU(Graphics Processing Unit)は CPU に比べて性能が向上している。その演算性能に注目して、GPU に汎用的な計算を行わせる GPGPU(General Purpose computation on GPU)への関心が高まっている。 また、CUDA[1]などの GPGPU プログラミング開発環境が提供されている。しかし、この開発環境は GPU アーキテクチャに合わせたプログラミングを必要とする。そのため、ユーザは GPU のアーキテクチャを意識しなければならずプログラミングは困難である。本研究では、GPGPU プログラムの難しさとそのプログラムを容易化する方法を検討する。

2. 実験

評価用プログラムは輪郭検出に用いるラプラシアンフィルタである。256×256、768×521、1200×1200、2448×2448、3648×2736 ピクセル(pixel)の 5 つのサイズの画像を用意する。

3 つの方法で実装し、それぞれの処理時間を計測する。プログラム 0(P0)は、CPU 向けに C で作成したものである。プログラム 1(P1)は、P0 を CUDA で記述し直したものである。プログラム 2(P2)は、コンスタントメモリを使う様に P1 を記述し直したものである。図 1 に結果を示す。

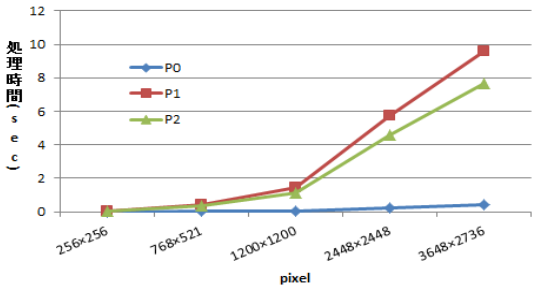

図 1. 実験結果

P0 よりも P1 の処理時間が増えた理由は、GPU コアから遠く低速なグローバルメモリを使用したためである。P2 が短縮できた理由は、キャッシュされる高速なコンスタントメモリを使用したためである。しかし、P2 は P0 と比べるとまだ低速である。その理由は、画像データがグローバルメモリに配置されていて、GPU のコアが演算をする際にグローバルメモリにアクセスしているためである。

3. 検討

単に CUDA 用にプログラムを記述するのみでは高い性能は得られない。GPGPU プログラミングにおける問題点は 4 つある。①CUDA 専用の言語を使用すること。解決策は一般的な言語、例えば C 言語で記述できれば良い。②明示的にデータ転送を行うプログラムを記述しなければならないこと。デバイス側(GPU)のコアからホスト側(CPU)のメモリを参照でき、両方のメモリを区別することなく使用できれば良い。③CUDA メモリモデルが複雑である。メモリ管理を考えないで記述できれば良い。④並列化が困難である。自動並列化できれば良い。以上を表1にまとめる。

表 1. 問題点と解決策

問題点	解決策	実現されている
CUDA専用の言語	C言語をそのまま使える記述方法	OpenACC[2]
データ転送	GPUからホストメモリを参照できる	Mapped Memory[3]
		Unified Virtual Addressing[3]
	両方のメモリを区別しない記述	Unified Memory[3]
		hUMA[4]
CUDAメモリモデル	メモリ管理を考えない記述	×
逐次処理	自動並列化	PGIアクセラレータ[5]
		Goose Domain-Specific Compiler[6]

表 1 で実現できてないものは「メモリ管理を考えないで記述できる」ことである。それを実現する 1 つの方法は、ホスト側のメモリとデバイス側のメモリを1階層の共有メモリとすることである。ハードウェア実装することは可能である。しかし、数千もあるコアが共有メモリに一斉にアクセスすることになり、処理性能は落ちる。また、たくさんのコアが処理するため消費電力も増える。そのため、現実的な実装ではない。

現状のハードウェア構成を現実的な実装のひとつの解であると仮定して、プログラミング容易化を検討する。プログラマがメモリ上の配置を全て管理するのでは無く、ハードウェアやソフトウェアからサポートを受けることが出来れば、負担が軽減する。例えば、プログラム中にヒントを挿入し、メモリ上の配置決定をコンパイラに任せる。再利用の頻度や参照するコアの分布などのデータの特徴をプログラマがコンパイラにヒントとして与えると、コンパイラは特徴を考慮した配置を考える。コンパイラの代わりに専用のハードウェアがデータの入れ替えを実施しても良い。

4. まとめ

P1 を P2 に改良するにはメモリモデルを意識することが必要で、プログラミングを容易化するには、プログラマがメモリ上の配置を全て管理するのでは無く、ハードウェアやソフトウェアからサポートを受けることが必要である。

参考文献

[1] https://developer.nvidia.com/cuda-zone (2015.01.29 アクセス).
[2] http://www.openacc-standard.org/ (2015.02.04 アクセス).
[3] NVIDIA CUDA C PROGRAMMING GUIDE v6.5 (2014).
[4] http://developer.amd.com/ (2015.02.05 アクセス).
[5] http://www.pgroup.com/resources/accel.htm (2015.01.28 アクセス).
[6] http://www.kfcr.jp/goose-e.html (2015.01.28 アクセス).