所属実験室	計算機システム	指導教員	佐藤 寿倫
学籍番号	TL061329	氏　名	山本 鉄兵
論文題目	CUDA プログラミングの難点に関する調査		

1. はじめに

GPGPU(General Purpose Computation on Graphics Processing Unit)と呼ばれている GPU で汎用演算を行うという試みが近年盛んになっている。GPGPU 向けの統合開発環境として注目されているのが、NVIDIA 社が無料で提供している CUDA である。本研究では CUDA によるプログラムの難点を、CPU のみでおこなう普通の逐次プログラミングと比較して考察を行う。

2. CUDA プログラミングの難点

2.1 CUDA のメモリモデル

CUDA では独自のメモリモデルが導入されており、CPU と異なる点は種類が多いこと、使用範囲の制限が多いことが挙げられる。

CUDA プログラムを組む時は CPU と GPU でそれぞれプログラムが処理されるため、普通の逐次プログラムとはメモリへのアクセスが異なっていることに注意せねばならない。シェアードメモリは 16 個のバンクによって構成されており、同じバンクに複数のアクセスが起こると、順番に処理されるため作業量が増加し、実行時間が長くなってしまう。これをバンクコンフリクトと呼び、これを回避することがプログラムを組む上での課題となる。

2.2 スレッド

GPGPU では、CPU と比べて圧倒的な数を誇るスレッドで計算することで GPU の性能を最大限引き出すことを実現している。CUDA ではスレッド内に階層構造の概念を導入し、スレッドの管理を効率的に行っている。CPU のプログラムの中でカーネル関数を実行する際に、グリッド内のブロック数や、ブロック内のスレッド数を指定することができる。

2.3 ワープ

GPU は CPU のよう分岐予測、アウトオブオーダ処理機構などの命令処理能力を持っていない。そこで GPU ではワープ（Warp）という 32 個のスレッドの単位で分岐の判定を行う。このときワープ内のスレッドで分岐の方向が異なる場合、ワープの作業量が増加し、実行速度の低下につながる。これをワープダイバージェントと呼ぶ。CUDA プログラミングではできるだけ複雑な分岐命令の数を減らすことがプログラム実行時間の短縮につながる。

3. まとめ

CUDA プログラミングは CPU と GPU を同時に使い、多数の演算コアにより高速な演算処理を実現している。そのため高速化のためには、並列プログラミングの知識が重要であり、ワープダイバージェントや、バンクコンフリクトなどを最適化により削減するための技術が問われることが初心者にとっては難点である。

参考文献

[1] 青木尊之、額田彰, ”はじめての CUDA プログラミング”, 工学社, 2009.

[2] NVIDIA , ”CUDA Zone”, http://www.nvidia.co.jp/object/cuda_home_jp.html

[3] 千曲エンジニアリング, ”CUDA を使う”, http://tech.ckme.co.jp/cuda.shtml, 2009.

[4] Hisa Ando, ”超並列プロセサ ― GeForce アーキテクチャと CUDA プログラミング”, http://journal.mycom.co.jp/special/2008/cuda/menu.html, 2008.

[5] “CUDA Information”, http://gpu.fixstars.com/index.php/, 2009.