4K-4

OpenCLにおける 処理量自動調整によるデータ並列処理の高速化

澤田 晃平† 稲葉 崇文† 今井 満寿巳† 津邑 公暁† 松尾 啓志†
†名古屋工業大学

1 はじめに

スカラ処理性能の向上が困難になりつつあることから，今日ではマルチコアプロセッサが広く用いられている．中でもピーク性能，コスト対性能比，電力対性能比に優れるヘテロジニアスな実行環境が注目されている．これは一般的に，計算を実行する1つ以上のユニットである計算ユニットと，それらを制御するホストから構成される実行環境である．一方で，このような実行環境で並列処理を実現するための様々な開発環境が提供されているが，そのほとんどが製品に特化したものであり，プログラマは実行環境毎にプログラムを再実装しなければならないという問題がある．

そこで実行環境に依存しないソフトウェア開発環境として OpenCL[1] が提案されている．OpenCL により，幅広い実行環境に向けたアプリケーションのソースレベル互換性を維持でき，プログラムの再利用性と開発効率を改善する事が出来る．

しかし OpenCL は実行環境ハードウェアの抽象化オーバヘッドにより，特定の製品向け開発環境を使用した場合に比べて実行速度が劣る．そこで本研究では OpenCL の抽象化オーバヘッドを低減する手法を提案する．

2 OpenCL

2.1 OpenCL のプラットフォームモデル

OpenCL における仮想的なハードウェアは，1 つのホストと 1 つ以上の OpenCL デバイスで構成される．ホストが実行するプログラムをホストプログラム，OpenCL デバイスが実行するプログラムをカーネルプログラム (カーネル) と呼ぶ．ホストはコンテキストの作成と OpenCL デバイスへの処理の割当を受け持つ．これは API 関数の呼び出しによって実現する．コンテキストとはカーネルの実行環境を定義したものである．一方カーネルが処理内容そのものであり，OpenCL デバイスにより実行される．

2.2 OpenCL デバイス

OpenCL デバイスは 1 つ以上の Computation Unit (CU) を含み，CU は 1 つ以上の Processing Element (PE) を含む．PE は仮想的なスカラプロセッサである．

カーネルが実行される際には，まずホストプログラムによって n (n=1〜3) 次元の抽象的なインデクス空間が定義される．インデクス空間のそれぞれの点をワークアイテム (WI) と呼び，グローバル ID と呼ばれる n 組の整数値が割り当てられる．さらに，ワークアイテムをいくつかまとめたものをワークグループ (WG) と呼び，WG 内の WI 数を WG サイズと呼ぶ．

また，WI 間で同期を取るための WG バリアという機構が存在する．この機構により，WI は自身が属する WG 内の他のすべての WI がバリアに到達するまで，同期待ちをしなければならない．

2.3 実行モデル

OpenCL はタスク並列処理とデータ並列処理をサポートしているが，本稿では特にデータ並列処理に着目する．データ並列処理では，すべての WI は同一のコードを実行するが，自身のグローバル ID によって処理対象を変化させる事ができる．また，実行の際 WG を CU に，WI を PE に非同期に割り当てる．

3 データ並列処理の問題点と解決方法

3.1 問題点

データ並列処理には WI の切り替えオーバヘッドが存在する．計算ユニットが，ある WI の処理を終えて WI を切り替える際，コンテキストスイッチが発生し，その後 DMA 転送で実行結果をホストのメインメモリに書き戻すため，これら 2 つのためのオーバヘッドが発生する．

また，WI の切り替えの際，WG バリアによる同期も必要となる場合がある．データ並列処理は単一 WI の処理内容が単純で短時間で終了する場合が多いため，これらのオーバヘッドが相対的に処理時間に対して大きくなり，性能が悪化する要因になる．

3.2 提案手法

本研究ではデータ並列処理の処理量を調整するための手法を 2 つ提案する．

3.2.1 WI の統合

1 つの WI 内で複数のデータの要素を処理するために，WI を統合する．これにより，グローバル ID や変数などの宣言，および条件判定の回数を削減できるため，カーネル実行の高速化が期待できる． また，総 WI 数が削減されることで，WI の切り替えオーバヘッドも削減され，処理時間全体も短縮される．

A Speed-Up Technique for OpenCL by Automatically Merging Work-Items
†Kohei SAWADA †Takafumi INABA †Masumi IMAI †Tomoaki TSUMURA †Hiroshi MATSUO

ただしWI内で複数の要素を扱うため，インデクス計算のためのコストが発生してしまう．また，従来手法に比べ1つのWIの実行時間が長くなるため，同期の際の待ち時間が増加する可能性があるといった問題もある．

3.2.2 WGサイズの変更

WGに対してもWI同様切り替えによるオーバヘッドが存在するため，処理量を調整する必要がある．そこで，WGサイズを変更する事でこのオーバヘッドを低減する．原則としてWGサイズは可能な限り大きく設定するが，同期が必要な場合，待ち時間の増大を防ぐため，小さく設定する．

4 実装

提案手法を実現するためにプリプロセッサを実装した．通常OpenCLでは実行環境のベンダによって提供されたホスト用コンパイラとカーネル用コンパイラを使ってコンパイルするが，これに先がけて，プリプロセッサによりプログラムを変換する．本章ではプリプロセッサの具体的な処理内容について説明する．

4.1 カーネルのフロー解析

WIの統合に伴い発生するインデクス計算のコストにより，却って実行速度の低下を招く場合がある．これを回避するため，変換前にカーネルのデータの依存関係や処理内容を解析することで，高速化が得られるかどうかを判定する．なお，高速化が得られないと判定した場合は変換しない．

4.2 ホストプログラムの解析と変換

プリプロセッサはホストプログラムを解析する事で，OpenCLデバイスへ処理を割り当てるAPI呼び出しからWI数，インデクス空間の次元数などの情報を取得し，これらの値から統合数を算出する．ここで，次元数と統合数はカーネルを変換する際にも必要な要素なので記憶しておき，その後，統合数に基づいてWI数とWGサイズを書き換え，処理量を調整する．

4.3 カーネルの変換

1つのWI内で統合数と同数のデータの要素を扱えるようにカーネルを変換する．変換にはいくつかの変換規則を用いる．単一ワークアイテム内で, ある要素に対してコードを実行し，その後インデクスを書き換え，別の要素に対して先ほどと同じコードを実行するといった変換方式では，インデクスの計算コストが原因で速度低下を招きかねない．そこで，1つの命令に対して複数要素処理するようにコードを生成する．これにより，変数や条件判定文などの命令が再利用できる．ただしその場合，書き換えるインデクスによって値が変わる変数は再利用できないため，そのような変数は複製する必要がある．

5 評価

5.1 評価環境

前章で述べたプリプロセッサを実装し，評価を行った．実行環境としてはSPEを6基搭載したCell/B.E[2]を用い，開発環境にはOpenCL Development Kit for Linux on Power[3]を用いた．またベンチマークプログラムには配列を加算するaddVector，行列積演算のMatrixMul，台形公式による積分演算をするintegral，LU分解を行うLUを用いた．

5.2 評価結果

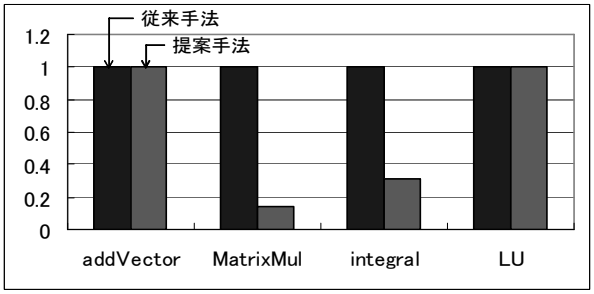

図1: 各プログラムにおける評価結果

評価結果を図1に示す．MatrixMulでは3.8倍，integralでは3.1倍の速度向上が得られたが，addVectorとLUでは速度向上が得られなかった．

MatrixMulおよびintegralはデータの要素間で依存関係が存在しない完全なデータ並列性を持つプログラムであり，提案手法によるカーネル実行の高速化とオーバヘッドの削減の効果が得られた．一方，速度向上が得られなかったプログラムについては，まずaddVectorは演算が非常に計量であるため，またLUはデータの要素間で強い依存関係があるために，インデクスの計算コストが大きいと判断され，プリプロセッサによる変換が行われなかったためである．

6 まとめと今後の課題

データ並列処理に対して処理量を調整することで最大で3.8倍の性能向上が得られた．今後の課題は処理量調整以外の高速化手法を実装する事や，よりハードウェアに特化した変換をする事が挙げられる．

参考文献

[1] Khronos OpenCL Working Group: *The OpenCL Specifiction*, 1.0 edition (2009).

[2] Sony Computer Entertainment: *Cell Broadband Engine Architecture*, 1.01 edition (2006).

[3] IBM: *OpenCL Development Kit for Linux on Power*, 0.3 edition (2011).