Cerium Task Manager の Multi Platform 対応

平成25年度 卒業論文

琉球大学　工学部　情報工学科

105744J 小久保 翔平

指導教員 河野 真治

目 次

図 目 次

表 目 次

第1章　序論

1.1　研究背景と目的

コンピュータを高速化するため、CPU のビット幅の拡張、動作クロックの高速化、キャッシュの大容量化などが図られてきた。しかし、これらの方法による性能の向上はすでに限界に達している。そこで並列化による高速化が注目された。同一アーキテクチャのプロセッサを複数台使った並列コンピュータ (ホモジニアス) が一般的でしたが、GPU の普及と高速化、GPU の演算資源を画像処理以外の目的にも使用する GPGPU (GPU による汎目的計算) の登場によって異なるアーキテクチャのプロセッサを組み合わせた並列コンピュータ (ヘテロジニアス) が出現した。これにより Many Core CPU とは比較にならないほどの並列化数を実現できるようになった。ただし、GPU の各コアは CPU のコアほど高性能ではなく、演算は高速だが、制御部分は弱い。また、ヘテロジニアスなシステムはホモジニアスなシステムとは異なり、メモリ空間がまったく同一でない場合であることが多い。このような場合、メモリコピーによるオーバーヘッドが大きくなる。よって、GPU を用いて高い並列度を出すためには、特定の計算に特化した Task の生成やスケジューリング、Task をパイプライン実行することでデータ転送をオーバーラップする必要がある。本研究では、OpenCL, CUDA を用いて GPU 上での Task 並列実行と Data 並列実行の両方をサポートし、CPU と GPU の Task をほぼ同じに記述することができ、自動でパイプライン実行を行いデータ転送をオーバーラップするように当研究室で開発した並列プログラミングフレーム Cerium を改良した。WordCount, FFT を用いて実行時間を測定して、その結果から Cerium 上での GPU 実行機構を評価する。また、Task を CPU, GPU 上での同時実行も可能にした。しかし、Task をどの程度の割合で CPU/GPU に割り当てるかというスケジューリング等の問題がある。測定結果からスケジューリング等の問題の解決方法についても考察し、信頼性のある並列プログラミングフレームを目指す。

第2章　Parallel Computing Platform

2.1　OpenCL

OpenCL とは、Multi Core CPU と GPU のようなヘテロジニアスな環境を利用した並列計算を支援するフレームワークである。

OpenCL には主に 2 つの仕様がある。

- OpenCL C
- OpenCL Runtime API

OpenCL C は演算用プロセッサ (本研究では GPU) 上で動作する、C 言語を拡張したプログラミング言語である。一方で、OpenCL Runtime API は OpenCL C で記述したプログラムを GPU 上で実行させるため、制御用のプロセッサ (本研究では CPU) 上で利用する API である。

オペレーティングシステムなどが処理される、メイン CPU などのことを host、GPGPU を搭載したグラフィックボードなどのことを device と定義している。OpenCL では device に CPU を割り当てることも可能である。OpenCL Application は host 側のプログラムと device 側のプログラムが一体となって動作する。この device 側で動作するプログラムを OpenCL では、特別に kernel と呼ぶ。

2.1.1　Command Queue

OpenCL では、デバイスの操作に Command Queue を使用する。Command Queue は device に OpenCL の Operation を送るために仕組みである。Command Queue は clCreateCommandQueue という OpenCL API に所属するコンテキストと実行対象となる device を指定することで生成される。

Command Queue では kernel の実行、input buffer の読み込み、output buffer への書き込みといった Operation が in order で実行される。Command Queue を作成するとき CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE のプロパティを指定することで Operation を out of order で実行することが可能になる。Operation を out of order で実行する場合、データの依存関係を記述する必要がある。各 Operation には event_wait_list と event を指定することができ、これらを利用してデータの依存関係を記述することができる。しかし、この CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE のプロ

パティをサポートしている device は少なく、Mac OS X では OS レベルでサポートしていない。パイプライン実行を行うためには kernel の実行やデータ転送を out of order で実行する必要がある。CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE のプロパティが無効の場合、複数の Command Queue を生成し、Command Queue を複数投入することで Operation を out of order で実行することが可能になる。

2.1.2 Memory Access

host 側は主にデータを input/output する Memory の確保を行う。GPU の Memory 空間 (図:2.1) や Cell の Memory 空間 (図:2.2) は Multi Core CPU(図:2.3) とは異なり、共有 Memory ではないため host と kernel 間でデータの共有ができない。アクセスするには Memory 空間ごとコピーしなければならない。

OpenCL では host 側で Memory Buffer を作成して Memory のコピーを行う。データの読み込みは clEnqueueReadBuffer、書き込みは clEnqueueWriteBuffer という API でそれぞれ行われる。前節で述べた通り、これらの Operation を Command Queue に Enqueue する。そして、event によってデータの依存解消が解消されると実行される。

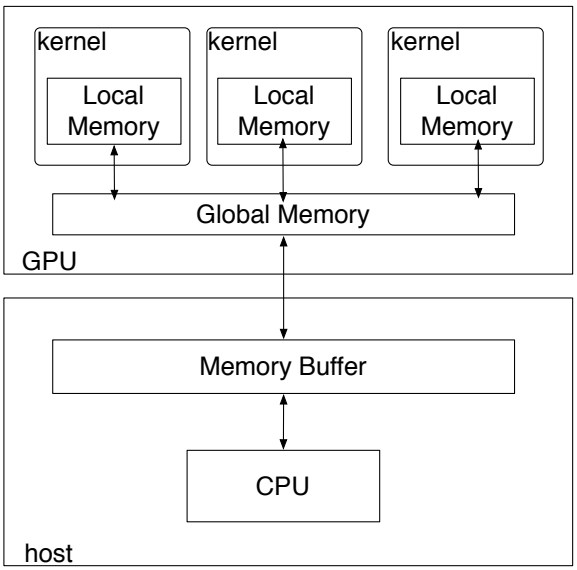

図 2.1: Gpu Architecture

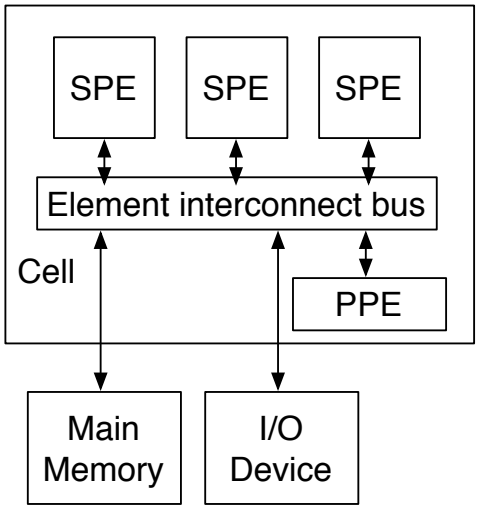

図 2.2: Cell Architecture

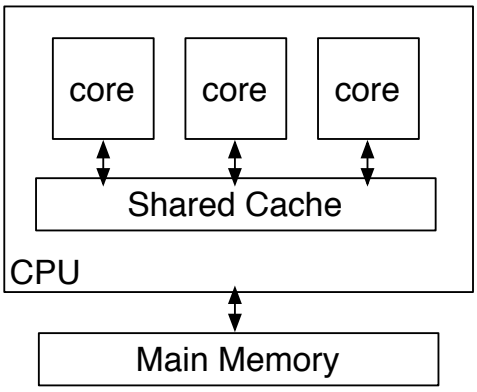

図 2.3: Cpu Architecture

2.1.3 Data Parallel Execution

多次元のデータ構造がある場合に高い並列度を保つには、それを分割して並列に実行する機能が必要である。これを OpenCL ではデータ並列と呼んでいる。OpenCL では次元数に対応する index があり、OpenCL は一つの記述から index の異なる複数の kernel を自動生成する。その添字を global_id と呼ぶ。このとき入力されたデータはワークアイテムという処理単位に分割される。

OpenCL はワークアイテムに対して、それぞれを識別する ID(global_id) を割り当てる。kernel は get_global_id という API によって ID を取得し、取得した ID に対応するデータに対して処理を行うことでデータ並列を実現する。この ID によって取得してきたワークアイテムをグローバルワークアイテムという。また、ワークアイテムは 3 次元までにデータを渡すことができる。

データ並列による kernel 実行の場合、clEnqueueNDRangeKernel API を使用するが、この関数の引数としてワークアイテムのサイズと次元数を指定することでデータ並列で実行できる。

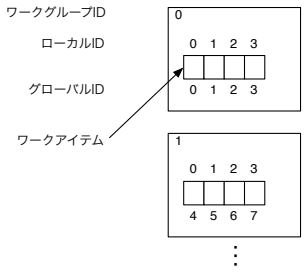

図 2.4: WorkItem ID

参考文献

[1] 河野真治. 検証を自身で表現できるハードウェア、ソフトウェア記述言語 continuation based c と、その cell への応用. 電子情報通信学会 VLSI 設計技術研究会, March 2008.

[2] 神里晃, 宮國渡, 杉山千秋, 河野真治. C から cell アーキテクチャを利用した cbc への変換. 電子情報通信学会 VLSI 設計技術研究会, March 2008.

[3] 神里晃, 河野真治. Continuation based c による ps3 cell のシミュレーション. 情報処理学会システムソフトウェアとオペレーティング・システム研究会, May 2006.

[4] 神里晃. Cell を用いたゲームフレームワークの提案. Master's thesis, 琉球大学理工学研究科情報工学専攻, Feb 2008.

[5] 宮國渡. Cell 用の fine-grain task manager の実装. Master's thesis, 琉球大学理工学研究科情報工学専攻, Feb 2009.

[6] 宮國渡, 河野真治, 神里晃, 杉山千秋. Cell 用の fine-grain task manager の実装. 情報処理学会 システムソフトウェアとオペレーティング・システム研究会, April 2008.

[7] 杉山千秋. Scenegraph と statepattern を用いたゲームフレームワークの設計と実装. 琉球大学工学部情報工学科 平成 19 年度卒業論文, 2008.

[8] 赤嶺一樹, 河野真治. Meta engine を用いた federated linda の実験. 日本ソフトウェア科学会第 27 会大会 (2010 年度), Sep 2010.

[9] 多賀野海人. Cell task manager cerium における task を用いたパイプラインの改良. Master's thesis, 琉球大学理工学研究科情報工学専攻, Feb 2011.

[10] 金城裕, 河野真治. Fine grain task manager cerium のチューニング. 日本ソフトウェア科学会第 27 会大会, Sep 2010.

[11] 金城裕, 河野真治. ゲームフレームワーク cerium taskmanager の改良. 情報処理学会システムソフトウェアとオペレーティング・システム研究会, Apr 2011.

[12] 金城裕, 河野真治. Cerium における datasegment api の設計. 日本ソフトウェア科学会第 28 会大会, Sep 2011.

[13] G. Jaeschke. On strong pseudoprimes to several bases. Vol. 61, pp. 915–926, 1993.

[14] 高山征大. 「cell regza」が搭載する並列化技術「molatominum」. Dec 2009.

[15] Yellow Dog Linux for PowerPC Computers. http://us.fixstars.com/products/ydl/.

[16] SourceForge.JP: Cerium Rendering Engine. https//sourceforge.jp/projects/cerium/.

[17] Sony Corporation. Cell BroadbandEngine ™ アーキテクチャ, 2006.

[18] International Business Machines Corporation, Sony Computer Entertainment Incorporated, Toshiba Corporation. *C/C++ Language Extensions for Cell Broadband Engine Architecture Version 2.6*, 2008.

[19] blender.org. http://blender.org/.

[20] opengl. http://www.opengl.org/.

[21] 赤嶺一樹, 河野真治. Meta Engine を用いた Federated Linda の実験. 日本ソフトウェア科学会第 27 会大会 (2010 年度), Sep 2010.