富山大学総合情報基盤センター広報　vol.7　(2010)　83-86頁.

CUDA, OpenCL は使えるのか？- GPGPU 開発環境・その周辺のいま

総合情報基盤センター　准教授　布村 紀男

１．はじめに

この頃の GPU(Graphics Processing Unit)はひとむかしと違い，元々の高いグラフィックス性能に加えて，ベクトル・行列演算の複雑な処理を行うことができる汎用演算器として進歩が目覚ましく，HPC(High Performance Computing)分野で大きな関心が集まっている．こうした GPU を一般目的のプログラムで活用する GPGPU(General Perpose)コンピューティングの話題は，一昨年あたりから情報処理学会誌で特集されたり[1]，コンピュータ情報雑誌でも取り上げられたり[2,3]，そして Web 上でも多くの情報を得ることができる．特に NDIVIA 社が提供している開発環境の CUDA は勢力的である．そんな中，生き馬の目を抜くコンピュータ業界においては，GPU に限らず，信号処理で使われる DSP，PS3 や家電製品に搭載されて Cell チップなどを CPU と組み合わせて利用するいわゆるハイブリッドなシステムへの流れが登場して来ている．それは，ベンダーに依存しないヘテロジニアスコンピューティングの標準化に向けての統一化として OpenCL が提案され，仕様も策定されるに至っている．

本稿では，こうした GPGPU を含むヘテロジニアスコンピューティングの話題を提供する．

２．CUDA について

GPGPU に関心ある読者の方は，すでにご存じのことかもしれませんが，はじめて CUDA という用語を聞いた方向けに，ちょっとだけ解説をしてみたい．なお，詳細については NVIDIA 社の CUDA 解説資料・マニュアル[6]をはじめ，解説記事[3]を参照するとよい．

2.1 CUDA とは

CUDA は（Compute Unified device Architecture no）の略で，「クーダ」と呼ばれている．NVIDIA 社が自社製の GPU で一般目的(GP)のプログラム作成するための開発環境を 2006 年 11 月に初めてリリースし，無償で提供している．現時点で Ver.2.3 が最新リリース版である．OS は 主要な Linux(SUSE, Fedora, RedHat, Ubuntu)をはじめ，Windows(XP, Vista)そして MacOS (10.5, 10.6)の各種に対応しているようである．CUDA2.3 からは Windows7 OS も利用できるようである．Linux と Windows は 32 ビット/64 ビットの両方をサポートしている．MacOS については，現在，32 ビットだけの提供のようである．

CUDA の SDK の付属されているサンプルプログラムはデモとしては非常に興味深く，これから GPGPU をやってみようとするモチベーションを高めてくれる．しかし，それらのプログラムソースコードを参考にして，いざ自分でプログラムを勉強しようとするとほぼ挫折するようである．(恥ずかしながら筆者もまさにその通りであった)

ではどうすればよいのか？やはり CUDA の特徴をとらえた，お手本になるコードから始めるのが王道なようである．これから CUDA を使って GPGPU をはじめようとしている方向けに待ち望まれた書籍[4]が出版されたので，これからプログラム開発する方に非常に有用であると思うので一読を薦めたい．

2.2 CUDA の特徴

CUDA の特徴を挙げれば，

・標準 C 言語と GPGPU 用の拡張機能を含む
・現在，GPGPU 環境としては一番普及している
・BLAS(CUBLAS)，Lapack(CULA)，FFT(CUFFT)といった代表的な計算ライブラリーが提供されている
・NVIDIA 社製の GPU がない場合でもエミュレーションモードが利用できる
・データ並列プログラミングが可能
（スレッド，スレッドブロック，グリッド）
・無償で利用できる
などがある．

Netbook の AtomCPU 向けの ION プラットフォームのチップセットにはNVIDIA社のGPUが搭載されており、また Apple 社製の MacBook や iMac には標準で GPU として NVIDIA 社の GPU の GeForce 9400M がチップセットに搭載されており CUDA を使える対応ハードが周辺に多くなって来ている. さらに、2009 年末に PGI 社からは CUDA に対応した Fortran コンパイラー[7]も提供されており、学術研究用途でのアプリケーションへも浸透していくことが考えられる.

2.3 CUDA を試してみる

実際に CUDA を利用するための導入についてここでは記述する. CUDA 開発環境のインストール作業方法は, CUDA のインストールガイドたは書籍[2]の付録には詳細にき CUDAZone[6]からまず、各 OS に対応したバージョンを選び,

- CUDA ドライバ
- CUDA ツールキット
- CUDA SDK

を入手する. 筆者は Linux 環境(Fedora9, 10),CentOS5.4 に導入している. さらに MacOS では MackBook OS は 10.5 に CUDA 2.2 を導入している. Windows 環境の導入は試みていないがそれほど難しくないと思われる. MacOS10.6(Snow Leopard)に CUDA2.3 をインストールするには、どうも次のようにしないとうまく入らないので紹介しておく[13].

インストールの次の順番で行う必要ある.

1. CUDA ツールキット 2.3a
2. CUDA ドライバ 2.3.1a
3. CUDA SDK 2.3a

また, SDK を make する場合には次のように引数を指定する必要がある.

```
$ cd /Developer/GPU ¥Computing/C/
$ sudo  make i386=1 x86_64=0
```

3. OpenCL について

3.1 OpenCL とは

OpenCL (Open Computing Language)はヘテロジニアス並列コンピューティングのための統一的なフレームワークの位置づけのようである. 現在, OpenCL は, クロスプラットホームであるグラフィックスAPIのOpenGLを管理しているクロノスグループによって標準化されている[8,9]. OpenCL の標準化メンバーとしてインテル, アップル, AMD, そして NVIDIA 社, さらにその他の多くのコンピュータ業界のプロセッサーベンダーやハードウェアベンダーなどが含まれている.

プログラミング言語仕様はOpenCL C言語である. OpenCL の大きな特徴は,何といっても演算デバイスを構成しているハードウェアがなんであるかはっきりと指定してしないことである.

解説記事[3]によれば, OpenCL は次の 4 つのモデルの理解が重要であると説明されている.

- プラットフォームモデル
- メモリモデル
- 実行モデル
- プログラムモデル

以下では, これらのモデルの概略を記述する.

3.2 OpenCL のプラットフォームモデル

まず,OpenCL が想定するプラットホームは, ホストと OpenCL デバイスが基本である.ここで, ホストとは 制御用プロセッサを意味している.ホストは実行, データ転送を行う. OpenCL デバイスは演算用プロセッサのことである. OpenCL デバイスは, 演算ユニット(compute units) とプロセシングエレメント(processing elements)から構成される.

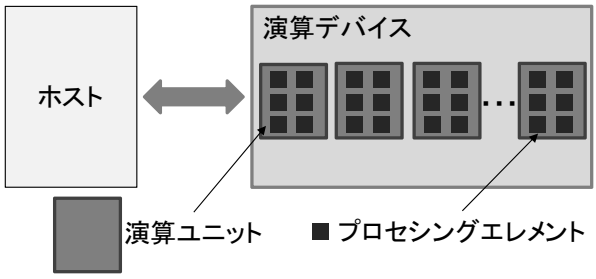

図 1 OpenCL プラットフォーム

3.3 OpenCL の実行モデル

実行モデルは以下で構成される.

- インデックス空間
- ワークグループとワークアイテム
- コンテキストとコマンドキュー

・カーネル

OpenCL ではインデックス空間が定義される. ワークグループとワークワークアイテムでの実行はワークグループ ID 番号で指定される. ワークアイテムはワークグループ内でローカル ID が割り当てられる. プロセッシングエレメントで実行されるワークアイテムは演算ユニットの単位でワークグループとして定義される. さらに全てのワークアイテムはインデックス空間内の全ワークアイテムの通し番号であるグローバル ID もしくはワークグループ ID とローカル ID の組み合わせで識別可能である. ワークグループは演算ユニットで実行され, ワークアイテムはプロセッシングエレメントで実行される.

コマンドには, 同期, メモリ操作, 計算の実行などの命令が用意されている. コマンドキューはインオーダー実行型とアウトオブオーダー実行型の２つがある.

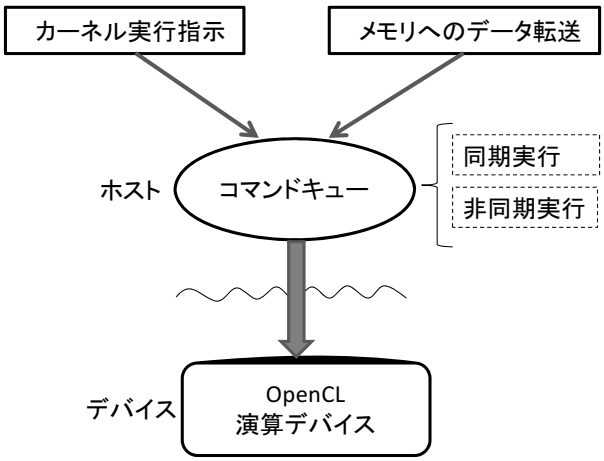

図 3 OpenCL コマンドキュー

3.4 OpenCL のメモリモデル

デバイスメモリ階層では以下の４つで定義されている.

・プライベートメモリ

ワークアイテム専用のメモリ領域

・ローカルメモリ

ワークグループ内のワークアイテム間で共有できるメモリ領域

・グローバルメモリ

全ワークアイテムから読み書き可能なメモリ領域

・コンスタントメモリ

全ワークアイテムから読み込みのみ可能なメモリ領域

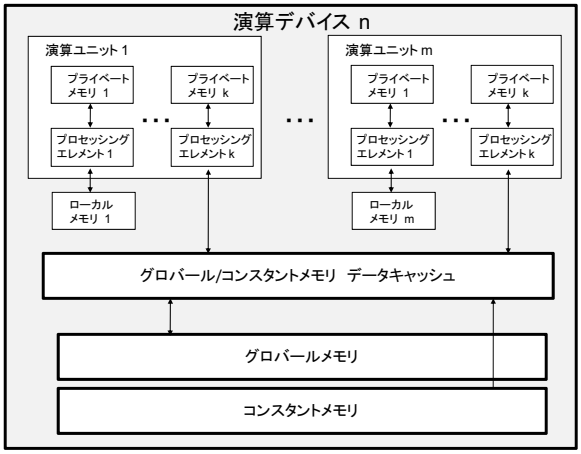

図 4 OpenCL のメモリモデル

3.5 OpenCL のプログラムモデル

CUDA では GPU での演算処理のデータ並列に重きをおいているようである. 一方, OpenCL では, データ並列プログラムモデルとタスク並列プログラムモデルの２種類の並列プログラムモデルの API が提供されている.

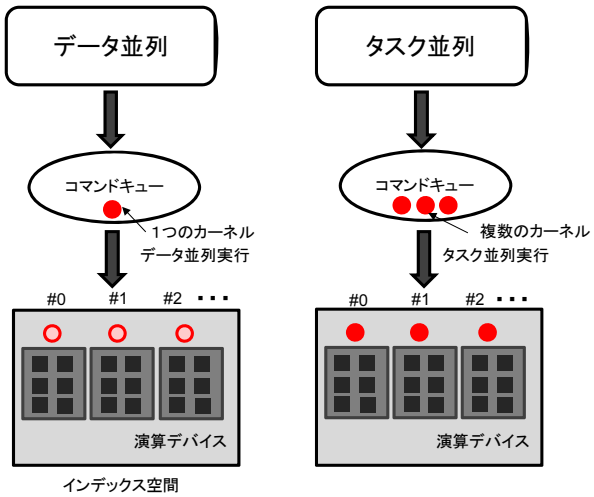

図 5 OpenCL のプログラムモデル

フレームワークはプラットフォーム層として OpenCL 実装を認識するプラットフォーム情報とデバイス情報を取得する API の２つを提供してい

る．OpenCL ランタイム API では，ホストプログラムがデバイスとやりとりするための API を提供している．OpenCL コンパイラはデバイスで動作するカーネルを作成する役割を持つようである．

本稿では OpenCL の具体的なプログラミングについては，筆者の勉強不足が最大の理由で解説することはできないが，興味を持たれた皆さんは，是非，参考文献[3,6]をご覧いただきたい．サンプルコードを眺める限り，OpenCL のランタイム API は CUDA のランタイムに比べて非常に複雑であるように思える．

４．CUDA から OpenCL への移行？

CUDA から OpenCL へ移行する場合には、最低限，用語の対応を知っておく必要がある．表 1 に代表的なものを記す．実行単位のスレッドはワークアイテム，スレッドブロックはワークグループに対応する．メモリでは、CUDA のシェアードメモリ，ローカルメモリは，OpenCL のローカルメモリ，プライベートメモリという名称になっている．

表 1 CUDA と OpenCL の用語対応

CUDA の用語	OpenCL の用語
スレッド	ワークアイテム
スレッドブロック	ワークグループ
グローバルメモリ	グローバルメモリ
コンスタントメモリ	コンスタントメモリ
シェアードメモリ	ローカルメモリ
ローカルメモリ	プライベートメモリ

５．使えるのか？

表題の「CUDA，OpenCL は使えるのか？」に対する筆者なりの意見を述べるならば，CUDA にいたっては，情報が多く，ライブラリーも豊富なのでちょっと頑張れば，何とか使える状態にあると言えよう．CUDA 対応アプリケーションも増えていると聞くので，開発者以外の一般ユーザーもその恩恵を受けることができる．一方， OpenCL は ver1.0 が仕様策定・公開されてから 1 年経過している．NVIDIA，AMD 各 GPU ベンダー等からもコンパイラー，ランタイム API を含む SDK が提供されているので，試せる環境は整っているように思われる．しかしながら，OpenCL のランタイム API は CUDA のそれよりも複雑になっているため，習得には相当の努力と忍耐力が必要になると思われる．本気で OpenCL を使おうとするには，それなりの覚悟が必要そうである．

６．おわりに

インターネットをはじめとするネットワーク分野では，TCP/IP がデファクトスタンダードとしてその地位を獲得するに至った．果たして，同じように OpenCL がヘテロジニアスコンピューティングのデファクトスタンダードとして なり得るかは今のところ予測はできない．まだ，「ヘテロジニアスコンピューティング」の標準化という目標に向けて始まったばかりなので OpenCL の動向を見守っていきたい．

参考文献

[1] 情報処理学会誌「情報処理」 Vol.50 No.2
[2] 日経 Linux「Linux で ION の性能を引き出す」2009 10 月号 p.56.
[3] ASCII.technologys 2009 12 月号:「GPGPU による並列処理」p.24.
[4] 青木尊之・額田 彰: 「はじめての CUDA プログラミング」工学社　2009 年 11 月
[5] 株式会社フィックスターズ:「OpenCL 入門 –マルチコア CPU・GPU のための並列プログラミング」インプレスジャパン　2010 年 1 月
[6] CUDA zone – CUDA 開発者向けリソース
http://www.nvidia.com/object/cuda_home_jp.html
[7] PGI CUDA Fortran
http://www.pgroup.com/resources/cudafortran.htm
[8] OpenCL Khronos group
http://www.khronos.org/opencl/
[9] OpenCL 仕様
http://www.khronos.org/registry/cl
[10] OpenCL 技術フォーラム
http://www.khronos.org/message_boards/
[11] NVIDAI のための OpenCL
http://www.nvidia.co.jp/object/cuda_opencl_jp.html
[12]OpenCL 入門
http://atl.amd.com/technology/streamcomputing/intro_opencl.html
[13] NVIDIA Forums
http://forums.nvidia.com/index.php?showtopic=106394