B-004

GPGPU における CUDA と OpenACC の計算性能比較と評価

Comparison and evaluation of CUDA and OpenACC computing performance in GPGPU

蓮見 祐貴† 増田 信之†
Yuki Hasumi Nobuyuki Masuda

1. はじめに

近年，ゲームや映像の高画質化に伴い，グラフィックス処理が膨大になってきている．その膨大なグラフィックス処理に GPU(Graphics Processing Unit)が多く用いられている．描画処理に使われる GPU の高い演算性能を利用して，汎用計算に応用した GPGPU(General Purpose computing on GPU)が盛んになっており，医療画像，数値流体力学，天体シミュレーション，暗号通貨の採掘，音声処理，ディープラーニングなど，さまざまな分野で用いられている．本研究では，NVIDIA 社が開発・提供するプログラミングモデルである CUDA(Compute Unified Device Architecture)とクレイ，CAPS，NVIDIA，PGI によって開発されたプログラミングモデルである OpenACC(open accelerators)を用いる．CUDA と OpenACC の 2 つのプログラミングモデルにおいて，それぞれで計算を行い，計算時間の比較および評価を行う．

2. CUDA(Compute Unified Device Architecture)

CUDA とは，NVIDIA 社が開発している，GPU 向けの汎用並列計算プラットフォームおよびプログラミングモデルである．CUDA の提供によって，GPU の内部構造を意識しなくてもよく，C 言語の知識があれば利用することができるため，汎用言語を用いる多くのプログラマが GPU コンピューティングを行うことができるようになった．また，CUDA は無償で提供されているため，広く普及した．

2.1 CUDA コンパイラ

CUDA でプログラミングをする際，一般的に“.cu”ファイルにホストコードとデバイスコードを記述する．“.cu”ファイルは nvcc コンパイラを用いてコンパイルすることで，実行ファイルの生成が行われる．CUDA では，拡張 CUDA 言語“.cu” で記述されたソースファイルは，コンパイルとリンクを行うために汎用 C++ホストコンパイラに引き渡され，通常の ANSI C++ソースファイルに変換される．

2.2 CUDA プログラミングモデル

CUDA のプログラムは CPU と PC のメモリのホスト，GPU とビデオカード上のメモリのデバイスに分けることができる．また，GPU で実行されるプログラムをカーネルプログラムと呼び，ソースコードの中では，__global__をつけて宣言する．カーネルプログラムはホスト側で起動する．大まかなプログラムの流れを図 1 に示す．

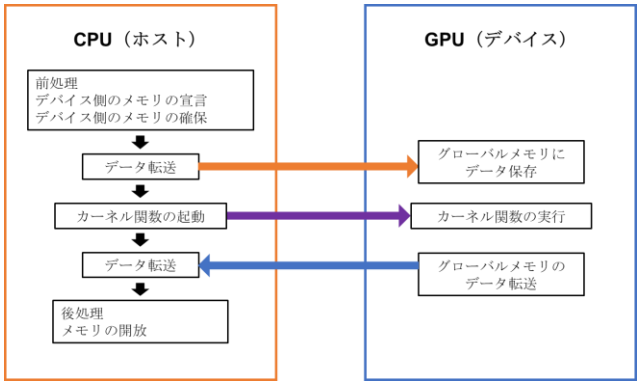

図 1 CUDA プログラムの流れ

逐次的に列挙すると次のようになる．

① デバイス側メモリの宣言（前処理）
② デバイス側メモリの確保（前処理）
③ ホスト側からデバイス側にデータ転送
④ カーネル関数の起動
⑤ デバイス側からホスト側にデータ転送
⑥ メモリの開放（後処理）

基本的にホスト側が主体となって命令が実行される[1].

3. OpenACC(Open ACCelerators)

OpenACC はクレイ，CAPS，NVIDIA，PGI によって開発された並列コンピューティングのための開発手法である．これまでの GPU を活用したプログラミングでは，OpenCL や CUDA などを用いる必要があったが，OpenACC を利用すると，ディレクティブを指定するだけで，通常のプログラミングをアクセラレータ用のプログラムへ変更できる．OpenACC の特徴として，以下のことが挙げられる．

- 新しいアクセラレータ用プログラミングインタフェース
- ディレクティブベース
- C/C++言語と FORTRAN に対応

3.1 ディレクティブベース

ディレクティブベースであると，プログラミングスタイルを大きく変更することなくアクセラレータ用のプログラムを開発できる．既存のプログラムをアクセラレータ用へ変更するのも簡単である．ほかのアクセラレータ用の環境では，ホストとアクセラレータ用のプログラムを開発しな

† 東京理科大学先進工学研究科電子システム工学専攻
Tokyo University of Science, Graduate School of Advance Engineering, Department of Applied Electronics

ければならないが，OpenACC では，単一コードとして記述できるため，開発もメンテナンスも容易である．

3.2 OpenACC コンパイラ

通常のホスト用プログラムに OpenACC ディレクティブを追加することで，OpenACC コンパイラは以下に示すホストコードとアクセラレータコードを自動生成する[2].

1. アクセラレータ側にメモリの割り付け
2. ホストとアクセラレータ間のデータ転送
3. カーネル関数の生成
4. スレッドの作成

これらのコードは以下のディレクティブで生成される．

1. kernels ディレクティブ
2. data ディレクティブ
3. loop ディレクティブ

3.3 OpenACC プログラミングモデル

OpenACC プログラミングモデルの動作概念図を図 2 に示す．ホスト側 CPU で実行する一部の処理をオフロードして，デバイス側で処理を行う場合の実行モデルである[3].

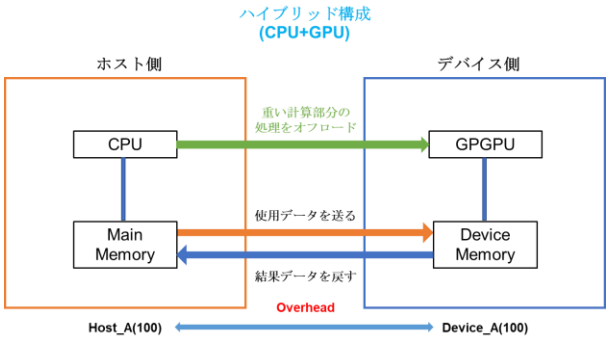

図 2　OpenACC プログラミングモデルの動作概念図

4. 実験

4.1 実験環境

本実験で用いたシミュレーション環境を表 1 に示す．計算時間の測定には，どちらも Nsight Tools を用いて，カーネル実行時間を測定した．

表 1　シミュレーション環境

OS	Ubuntu 20.04.2 LTS
CPU	Intel® Core™ i7-10700 2.90GHz
RAM	64GB(16GB × 4)
GPU	NVIDIA GeForce® RTX 3070 8GB
開発環境	NVIDIA HTC SDK
CUDA Compiler	nvcc cuda11.3
OpenACC Compiler	pgcc 21.5-0

4.2 測定

4.2.1 測定内容

本実験において以下の式(1)で表される$N \times N$の行列Bとベクトル$\boldsymbol{C}$との掛け算を求めるプログラムについて測定した．

$$\boldsymbol{A} = B \times \boldsymbol{C} \tag{1}$$

今回の測定では，$N = 1024$とし，ブロック数 8　スレッド数 128 での測定を行い，10 回測定した平均値を結果とした．

4.2.2 測定結果

測定結果を表 2 に示す．

表 2　計算時間の測定結果

	実行時間[μs]
CUDA	118.160
OpenACC	118.272

簡単な並列計算では，計算時間に差は見られなかった．

5. まとめと今後の課題

本稿では，CUDA と OpenACC の数値計算の比較する環境設定および簡単な並列計算での比較を行った．現状としては環境設定を終え，プログラムの実行および GPU の実行時間の計測が確認できた段階であり，行列とベクトル計算の測定結果のみに留まった．今後の課題としては，より計算量の多いプログラムや CGH のプログラムの実行によってさまざまな計算による比較を行う予定である．また，計算の最適化の検討し，CUDA と OpenACC の計算時間をどこまで詰められるのかを検討したい．また，それぞれのプログラムにおいて，データ型を変えた場合の時間の測定を検討している．

参考文献

[1] 伊藤 智義,『GPU プログラミング入門 ―CUDA5 による実装』, 株式会社講談社, 東京都, 2013 年.
[2] 北山 洋幸,『OpenACC 基本と実践 GPU プログラミングをさらに身近に』, 株式会社カットシステム, 東京都, 2018 年.
[3] Prometech Software, Inc. and GDEP Solutions, Inc. , “OpenACC Programing ディレクティブによるプログラミング 5 章 OpenACC ディレクティブの概要”, < https://hpcworld.jp/archive/SPG/Pgi/OpenACC/005.html>, 2021/6/17