CUDA（Compute Unified Device Architecture）

馬場 建
Takeru BABA

1　はじめに

近年，Graphics Processing Unit（以下，GPU）の性能向上により，コンピュータ上で高度な 3D 表示が可能となった．GPU は，画像処理の演算を並列実行することにより，処理速度を実現する．しかし，高度な画像処理をする場面は限られている．General Purpose GPU（以下，GPGPU）は，CPU で行われる汎用演算を，GPU で扱える形式に書き換えて計算する技術である．しかし，画像処理以外で並列処理をするには，データ型やスレッド管理の問題によりプログラムや最適化が困難だった．

GPU のメーカーの NVIDIA が提供する CPU と GPGPU の演算を同時にプログラムできる環境が Compute Unified Device Architecture（以下，CUDA）である．CUDA により簡単で最適化された GPGPU プログラミングは，シミュレーションやデータセンター，機械学習など幅広い分野に導入されている．

2　CUDA

2.1　概要

CUDA は，CPU と NVIDIA 製の GPU のハードウェアを抽象化し，逐次実行プログラムと同じような書き方で，並列プログラミングできるプラットフォームである．CUDA は並列処理の低レイヤーなロジックを自由に記述できる．これは GPGPU プログラミングに必要な機能を，C 言語に追加するというアプローチで開発されたことに起因する [1]．自由度が高いという特徴から，個人の使用目的に応じて容易にプログラムの最適化が可能である．また，NVIDIA による最新技術の導入が積極的に行われており，最適化された高速フーリエ変換やディープラーニングなどのライブラリが公開されている．

2.2　CUDA のメモリ構成

CUDA により抽象化された，CPU と GPU のメモリ構成と通信経路を Fig. 1 に示す．

CPU と GPU は，それぞれホストとデバイスとして扱われる．データ転送の通信路には PCI-Express を使用し，ホストメモリとデバイス側のグローバルメモリ間を通信する．デバイス側の処理実行時によくアクセスされる変数は，シェアードメモリに保存される．

GPU がデバイスと呼ばれるのは，ホストである CPU からの実行命令がない限り，メモリ管理や処理実行ができない関係性からである．グローバルメモリは，デバイス内のすべてからアクセス可能な，大容量だが低速なメモリである．シェアードメモリは，各ブロック内からのみアクセス可能な，小容量だが高速なメモリである．二種類のメモリを上手く使い分け，高速処理が実現できる．

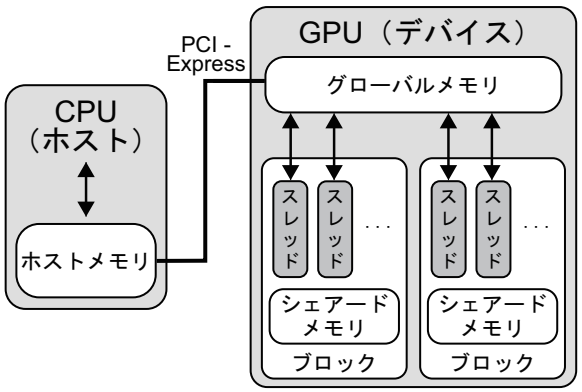

Fig.1　CPU と GPU のメモリ構成と通信経路

2.3　CUDA アプリケーションの流れ

CUDA のプログラムはコンパイル時に，デバイスコードとホストコードに分割する．デバイスコードはカーネルに変換され，ホストコードは一般のコンパイラでコンパイルされる．ホストは条件分岐が多い複雑な命令で，デバイスは大量の繰り返しに向いている．二つの要素が得意な役割ごとに，交互に実行処理して進んでいく [2]．

処理順序の詳細は次の通りである．但しここではホストを「H」，デバイスを「D」として表記する．

1. H は D の入出力とカーネルの保存領域を確保.
2. H は D で PCI-Express を介してデータ入力.
3. H は D のカーネルを起動.
4. D は指定ブロック数のスレッドを起動.
5. D はカーネルの処理結果を H に出力.
6. H は D のメモリを開放.

CUDA アプリケーションの実行単位を Fig. 2 に示す．ホストは並列処理を扱うときに，CPU のコアの数のみを考える．デバイスは並列処理を扱うときに，全スレッドを階層構造で考える．一つのグリッドは複数のブロックを持ち，一つのブロックは複数のスレッドを持つ．カーネル実行時は三つの階層のうち，処理に用いられるブロック数と，各ブロック内でのスレッド数を指定する [3]．

2.4　最適化

GPGPU の最適化には，全体的な利用率に着目した方法と，メモリ管理を効率化しアクセス時間の遅延を削除する方法が存在する．並列処理は最適化がされていなければ，CPU に比べて十分な高速性を発揮できるとは限らない．前者は，未使用の演算器が発生しないようなブロック数と

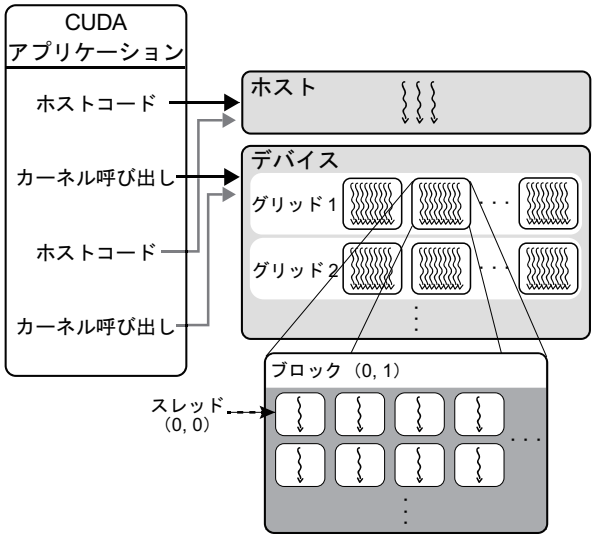

Fig.2　CUDA アプリケーションの実行単位

スレッド数にすることや，条件分岐で実行しない部分を減らす方法である．

後者は，ホストとデバイス間の通信での遅延や，デバイスメモリへのアクセス速度を解決する方法である．CUDA はストリーム関数が用意されており，データ転送とカーネル実行が同時に行うことが可能である．ストリーム関数によりデータ転送の待ち時間を減らすことで，PCI-Express の通信速度の遅さを補う．

またデバイスには，Fig. 1 のように多種のメモリが存在し，用途に合わせて使い分ける必要がある．高速で小容量のシェアードメモリは，ブロック内でよく使う変数を保持する．グリッド内で共通のグローバルメモリは，大容量だがアクセスが遅いため，保持する変数やデータ配置，計算順序を考慮する必要がある．データは明示的に保存先を指定しない限り，グローバルメモリに保存される．シェアードメモリを指定することで，プログラミング可能なキャッシュを扱うことができる．

3　CUDA の応用事例

3.1　自動運転

NVIDIA が提供する車載用 GPU の NVIDIA DRIVE は，フォルクスワーゲンやベンツ，トヨタなどの 145 社が採用している．特に，中国の百度は，自動運転車向けのソフトウェアをオープンソース化するアポロ計画を進めている．人工知能やディープラーニング，高精細地図生成において CUDA を活用している．2017 年のデモで公開された自動運転車は，人間のインストラクターの運転方法をディープラーニングで学習したものであり，複雑な環境において適切な運動行動が行えることを実証した．

2018 年に百度は Apolong と呼ばれる，特定の場所での完全自動運転のバスの実用化を発表し，北京市の AI 公園などで走行している．2019 年には日本での実証実験として，Apolong が十台持ち込まれる予定である．

3.2　ディープラーニング

機械学習のライブラリとして有名な TensorFlow や Chainer は，NVIDIA の GPU 使用時には，CUDA と拡張ライブラリの CUDA Deep Neural Network library（以下，cuDNN）を必要とする．cuDNN は，ディープニューラルネットワーク用のライブラリで，畳み込み演算などの深層学習に利用される処理を高速化する．cuDNN により，学習効率は CPU と比較して十倍以上に上げることが可能である．

SONY の研究者チームは 2018 年 11 月に，ResNet-50 という深層学習の方法に cuDNN を使用した．画像認識学習のデータセットを，75% の精度で正しく認識するために，224 秒で学習することに成功している．中国テンセント社の同年 7 月の記録の 396 秒と比較すると，約半分の時間で終了した．

4　今後の展望

CUDA の提供の開始により，一般人でもリソース負荷の高い開発が容易になった．CUDA は並列化が有効な様々な分野へ導入され，それぞれの分野へのライブラリの整備が年々と進んでいる．CUDA の使用が主流となる分野は，以下二種類の技術とともに広がると考えられる．

一つ目の技術は，Jetson と呼ばれる省電力で小型の組込みシステム向けの GPU である．従来，電力消費と体積の物理的問題から，GPU が使えなかった自律の小型ロボットの発展を支えると考えられる．ロボット上に CUDA による並列処理を組み込むことで，物体識別や音声認識のレスポンス性が向上し，安全で自然な動きを表現できる．特に，コミュニケーションロボットにおいては，高速な反応が生物のリアリティ性を向上させると考えられる．

二つ目の技術は，2018 年 10 月に公開された RAPIDS という CUDA の拡張ライブラリである．RAPIDS は世界初の並列処理によるデータ分析ツールである．従来は，市場や環境の複雑なパターンの検知及び予測をすることが困難だった．しかし，ライブラリによって容易に扱えるようになる．企業は経営戦略や新商品の売れ行き予測のために，大量のデータを高速に解析する必要性がある．また，IoT による大量のデータ処理の対策として，RAPIDS が貢献すると考えられる．

参考文献

1) Hisa Ando, GPU を支える技術, 技術評論（社）, 東京, 2017.
2) John Cheng, Max Grossman, Ty McKercher, CUDA C プロフェッショナルプログラミング, （社）インプレス, 東京, 2015.
3) NVIDIA CUDA Compute Unified Device Architecture プログラミング・ガイド Version 1.1 - NVIDIA, https://www.nvidia.co.jp/docs/IO/51174/NVIDIA_CUDA_Programming_Guide_1.1_JPN.pdf, 参照 Apr.21, 2019.