B-003

Jetson Nano における GPU を用いた高性能計算システムの高速化
Accelerating High-Performance Computing Systems with GPU in Jetson Nano

清水 雄斗† 増田 信之†
Yuto Shimizu Nobuyuki Masuda

1 まえがき

近年，コンピュータの性能向上に伴い，大規模かつ高精度な数値計算の需要が高まっている．このような計算を行う際，近年ではグラフィック処理用の半導体チップである GPU を数値計算に応用する GPGPU(GeneralPurpose Computing on GPU) という技術が広く用いられている．しかし，GPU を搭載した高性能なコンピュータは，大型になりやすい・消費電力が大きいなどの課題を抱えているため，活用できる場面は安定した電力を供給できる屋内などの環境に限定される．一方で，数値計算の活用が期待されている場所や状況は様々であり，その中には，自動車，IoT 機器，ロボット，モバイル端末等の，ユーザーに近い場所 (エッジ領域) も含まれる．エッジ領域で高性能な計算を行う場合，限られたサイズの機器に組み込むことでも利用できるような省スペース性や，バッテリー駆動でも利用できるような省電力性などが必要とされるケースも多い．以上の理由により，計算システムの小型化・省電力化は重要な課題であるといえる．

本研究では，NVIDIA 社が提供する小型かつ低電力で動作するシングルボードコンピュータである Jetson Nano を用いて大規模な数値計算を行うプログラムを作成し，一般的な GPU 搭載コンピュータと計算性能を比較し、評価した。

2 Jetson Nano

Jetson Nano とは，NVIDIA 社が提供する開発ボードであり，大きさは 100mm × 80mm と小型ながら CUDA 対応の 128 コア GPU を搭載している．また，一般的な GPU 搭載コンピュータの消費電力が数百 W であるのに対し，Jetson Nano の消費電力は 5〜10 W 程度と非常に小さい．以上の小型かつ低消費電力という特徴より，Jetson Nano はエッジ領域における AI の推論システムに適していると言える．

3 CUDA(Compute Unified Device Architecture)

CUDA は，NVIDIA 社が開発・提供している，GPU 向けの汎用並列コンピューティングプラットフォームおよびプログラミングモデルである．言語としては C/C++言語を拡張したものであり，CUDA を利用することで，容易に並列アルゴリズムを実装することが出来る．

3.1 プログラミングモデル

CUDA プログラミングの流れを図 1 に示す．CUDA のプログラムは，ホスト (CPU) で動作させるプログラムとデバイス (GPU) で動作させるプログラム（カーネル）に分かれている．並列処理を行う際は，スレッド数を指定してカーネル関数を起動するだけでよいため，開発者は逐次的にコードを記述するだけで GPU プログラミングを行うことができる．

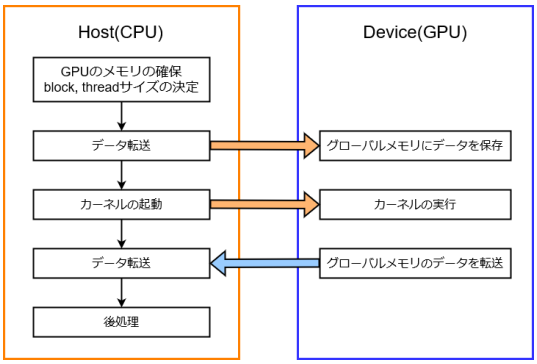

図 1 CUDA のプログラミングモデル

3.2 スレッドの構成

ホスト側からカーネル関数を実行すると，実行制御はデバイスに移動する．デバイスでは大量のスレッドが生成され，カーネル関数によって規定された処理が各スレッド上で実行される．CUDA のスレッドの構造を図 2 に示す．スレッドは複数のスレッドからなるブロックと複数のブロックからなるグリッドによって構成されている．

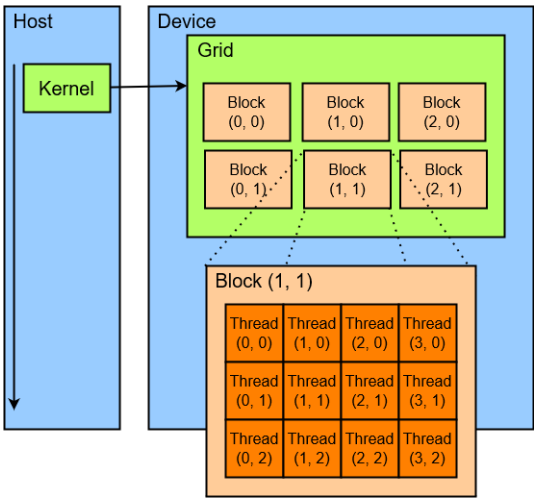

図 2 CUDA のスレッド構造

グリッド内のスレッドは全て同じグローバルメモリ空間を共有する．また，ブロックは互いに協調して動作できるスレッドのグループであり，ブロックに属するスレッド間の同期や，ブロック内で共有されるメモリ (シェアードメモリ) 等の機能を利用できる [1].

† 東京理科大学 先進工学研究科

4　実験

本研究では、計算能力の比較を行うために，以下の式 (1) で表される $N \times N$ の密行列の積を求めるプログラムを実行し，その実行時間を測定した．

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

なお，$N = 2048$ とし，ブロックサイズは 256，グリッドサイズは 16384 で。

4.1　高速化手法

$N \times N$ の行列積の計算を高速化するための手法として，シェアードメモリを使用して実装を行った．シェアードメモリは，同じブロックに属するスレッド同士が共有できるメモリのことである．シェアードメモリは GPU 上に物理的に存在するメモリであるため，ローカルメモリやグローバルメモリと比べて帯域幅がはるかに広く，遅延がはるかに少ないという特徴がある [2][3]．行列計算の実装イメージを図 3 に示す．

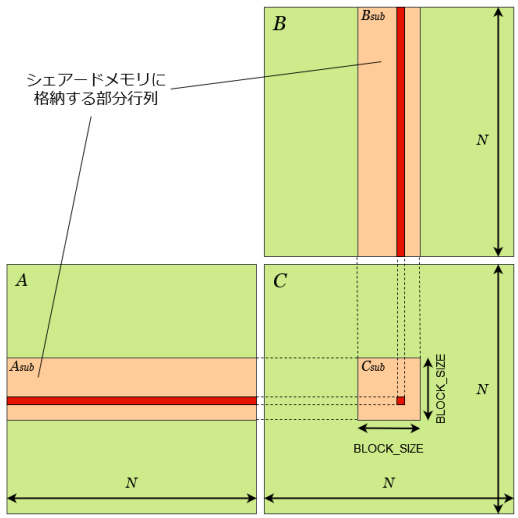

図 3　シェアードメモリを使用した行列積の計算

この方法では，各ブロックが C の部分行列 C_{sub} を計算し，ブロック内の各スレッドが C_{sub} の 1 つの要素を計算している．C_{sub} は，A の部分行列 A_{sub} と B の部分行列 B_{sub} の積になっており，計算を行う際に A_{sub} と B_{sub} をシェアードメモリにまとめて格納することでグローバルメモリへのアクセス回数を 1/BLOCK_SIZE に減らすことが可能であり，高速化が期待できる．

4.2　実験環境

本実験で使用した GPU 搭載 PC と Jetson Nano のシミュレーション環境をそれぞれ表 1，表 2 に示す．

表 1　シミュレーション環境 (GPU 搭載 PC)

OS	Ubuntu 18.04
CPU	Intel®Core™i7-8700 CPU 3.20GHz
RAM	24GB
GPU	NVIDIA GeForce®RTX3070
CUDA	11.1

表 2　シミュレーション環境 (Jetson Nano)

CPU	ARM Cortex-A57 (4 コア 1.43GHz)
RAM	4GB
GPU	NVIDIA Maxwell (128 コア)
CUDA	10.2

測定に用いたプログラムは，CUDA を用いて行列積を単純に実装したものに加え，節 4.1 で述べた手法でシェアードメモリを活用して高速化をはかったもの，GPU を使用せず CPU のみで計算を行うものの 3 種類を使用し，GPU 搭載 PC 上で実行した場合と Jetson Nano 上で実行した場合についてそれぞれ測定を行った．

4.3　実験結果

シミュレーション結果を表 3 に示した．

表 3　シミュレーション結果

	実行時間 [s]	
	GPU 搭載 PC	Jetson Nano
単純実装	0.1050	2.1839
シェアードメモリ使用	0.1105	1.0429
CPU のみ	118.7063	867.4249

Jetson Nano 上で GPU を用いた場合，PC 上で CPU のみ用いて計算する場合に対しても 50 倍程度高速であり，Jetson Nano が並列処理に適していることが確認できた．

GPU を使用した場合については，単純実装の場合，GPU 搭載 PC と Jetson Nano の実行時間には 20 倍以上の差があるが，シェアードメモリを用いて高速化することで，Jetson Nano 上での実行時間は 2 倍程度早くなり，GPU 搭載 PC との実行時間の差を約 10 倍まで縮めることができた．なお，GPU 搭載 PC 上ではシェアードメモリを使用した場合でも高速化しておらず，むしろ 10 % 程度速度が低下していることが分かった．

5　まとめと今後の課題

本項では，行列積の計算の実行時間の比較を行った結果，単純に実装を行った場合は GPU 搭載 PC の処理速度は Jetson Nano に対して 20 倍ほど早かったものの，高速化処理を行うことでその差を 10 倍程度にまで縮めることができることを確認した．

今後は，Jetson Nano をエッジ領域で活用することを想定し，画像処理，画像認識を行うプログラムについて，高速化の検討や性能比較を進めていく予定である．

参考文献

[1] John Cheng, Max Grossman, Ty McKercher,　『CUDA C プロフェッショナルプログラミング』，株式会社インプレス，p36，2018 年
[2] John Cheng, Max Grossman, Ty McKercher,　同上，p163
[3] Jason Sanders, Edward Kandrot,　『CUDA BY EXAMPLE 汎用 GPU プログラミング入門』，株式会社インプレス，p36，2015 年