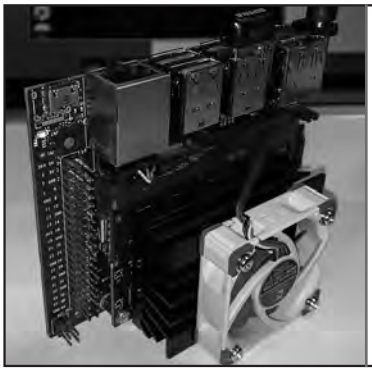
連載　githubでソースコード＆実行ファイル公開！



AI向け行列計算プログラムの作成

超並列演算器 NVIDIA GPU入門

〈5〉（最終回）最適化したGPUのパフォーマンス

桑野 雅彦 Masahiko kuwano

表1　Jetson Nano内蔵メモリを使った行列演算の処理時間を比較し，特徴を整理した

コア	メモリ	時間［秒］	特　徴
GPU	グローバル・メモリ（共有メモリ）	1.7	GPUプログラミングにおいて最も標準的なメモリであり，そこそこの性能を引き出せるが，扱いはやや面倒
	ユニファイド・メモリ	13.5	漫然とそのまま使うと性能面では不利になるが，広大なメモリ領域を自由に使えるような記述ができるのは大きな利点
	シェアード・メモリ	0.7	容量（48Kバイト）はそれほど多くなく，グローバル・メモリとの間の転送をカーネル・コード（GPU側で実行するプログラム）で行わなくてはならないなど，扱いもやや面倒ではあるが，速度面では圧倒的に有利
	ユニファイド・メモリ＋シェアード・メモリ	3.6	ユニファイド・メモリとシェアード・メモリを併用することで性能面の不利をかなりカバーできる．頻繁にアクセスするデータをシェアード・メモリに移しておく程度でもGPUの効果を期待できる
CPU（内蔵のARM Cortex-A75M）	メイン・メモリ	23.2	—

本連載では，NVIDIA製の100ドル・スタータキット Jetson Nanoを動かしながら，組み込みAIや科学計算に利用が広まっている「GPU」のハードウェアへの理解を深め，並列処理プログラミングの技術をマスタします．NVIDIA製GPUコア「CUDAコア」用のC言語「CUDA C」を使って，GPUのハードウェアを実感しながらプログラミングを進めます．

● 今回の内容

テーマは，AIアプリケーションで多用される「行列演算」の高速化です．大きな行列演算データを扱うためのメモリ・アクセスの最適化を解説します．

第4回では，GPUプログラムの開発の負担を減らす仮想メモリ・アクセス機能「ユニファイド・メモリ」とGPUコア専用の高速アクセス・メモリ「シェアード・メモリ」について概説し，観賞用シミュレータ（連載3回目で制作）で使えるサンプル・プログラムを作成しました．

今回は，メモリ・アクセスを最適化した結果，行列演算（サイズは64×64）の処理時間がどのくらい短縮したのか，Jetson Nanoを動かして測ってみます（**表1**）．　〈編集部〉

■ 行列演算はGPUと相性が良い

GPUプログラミングにおいて，「行列の乗算」は定番の処理です．AIが流行り，雑誌や書籍で画像認識や音声認識が注目を集めていますが，このAIのかぎは「行列演算」です．特に大事なのが乗算，すなわち「行列どうしのかけ算」です．行列演算はこの「かけ算」の量が非常に多いです．

Appendix（p.139）の式（26）の場合，次の1行だけでもかけ算の回数は9回です．

$$|2\times1+1\times4+3\times7 \quad 2\times2+1\times5+3\times8 \quad 2\times3+1\times6+3\times9|$$

式（26）の行列の大きさは3×3なので，計27回（＝3×3×3）のかけ算が必要です．

行列の大きさが4×4になれば4×16＝64回，5×5なら5×25＝125回と，かけ算の回数は行列サイズの3乗で増えていきます．一方，行列演算の各要素の演算は，積和計算の「独立性」が高いことが特徴です．例えば，Appendixの式（26）のような演算でも，GPUは3つの独立した積和演算を同時に実行できます．

$$2\times1+1\times4+3\times7 \cdots\cdots (1)$$
$$2\times2+1\times5+3\times8 \cdots\cdots (2)$$
$$2\times3+1\times6+3\times9 \cdots\cdots (3)$$