第4部

第2章 C/C++拡張のSIMDプログラミング・モデルで環境構築から画像処理の実例まで

インテルGPU向け言語 C for Metalを試す

竹田 大将，近藤 鯛貴，佐藤 裕幸

● インテルGPUプログラミングに適した言語

GPUのアーキテクチャが分かったら実際にGPUにいろいろ計算させてみたくなります．ここではインテルGPUでGPGPUプログラミングをするための環境を構築し，実際にGPUでプログラムを動かしてみます．

GPGPUプログラミングのツールセットまたはプラットフォームと言えば，NVIDIAのGPUを使うならCUDA，AMDやインテルのGPUを使うならOpenCLなどが挙げられます．

今回はインテルのGPUを扱うのでOpenCLでもGPUコード・プログラミングはできます．しかし本稿では，よりインテルGPUのSIMDアーキテクチャに沿った細かいチューニングが可能なC for Metal言語（以降，CM言語）と，そのSDKを使ったGPGPUプログラミングを紹介します．

ハードウェアを意識した記述ができる

CM言語はインテルGPU向けの独自のプログラミング言語です．2018年にC for Mediaという名前で公開され，現在もオープンソースでコンパイラの開発が続けられています．新たな言語を覚えるのは大変ですが，CM言語を苦労しながらでも書くメリットがあります．

● キャッシュの動きも意識して制御できる

CM言語はC/C++の拡張言語であり，明示的なSIMDプログラミング・モデルで構成されます．従って，インテルGPUのアーキテクチャを意識しながら，よりハードウェアに近い直感的な命令でプログラミングができます．

▶変数とレジスタとの対応を制御しやすい

プログラム内で使用されるvector型やmatrix型といった変数は，直接レジスタ・ファイルに展開されます．つまり，レジスタ使用率が意識しやすいのはもちろんのこと，細かいレジスタ操作や，レジスタ節約のチューニングも思いのままです．

▶データの流れを制御しやすい

各メモリ，レジスタ間のデータの移動とキャッシュの明示的な制御が可能なので，時間のかかるデータ転送をどのタイミングで行うかや，どのキャッシュを通るのかといった細かいところまで開発者の思い通りにできます．

まとめると，よりベアメタルに近いプログラミングができるのでアーキテクチャを意識しながら使用すればインテルGPUの能力を余すことなく引き出す力を秘めた言語だと言えます．

開発環境の構築手順

CM言語のプログラムは，CM言語用のコンパイラを含むさまざまなツールやサンプルがセットになったC for Metal Software Development Kit（以降，CM_SDK）を使用することで，簡単に開発が始められます．

CM_SDKは，Linux kernel 5.3.0が動作するUbuntu 20.04システムで動作が確認されています．GPU上でアプリケーションを動かすには，インテルの比較的新しい世代（第9世代以上）のCPUが必要です．CPUエミュレーションも提供されているので，皆さんのPCに搭載されたGPUがサポート対象外であっても，ある程度は体験できます．

環境構築の方法は，2通りあります．それぞれの具体的な手順を次に示します．

● 構築方法1：Dockerを使う

本稿の内容を試しやすいように，CM_SDKが構築されたDocker環境とサンプル・コードを整えたリポジトリを作りました．これを使って解説していきます．

```
https://github.com/DefiosLab/cm_sdk_sandbox
```

リスト1のコマンドで，CM_SDKが使えるDocker環境が整います．