■光学工房

OpenMPによる三行革命 ～三行でできるプログラムの並列化～

近年，コンピューターの中央演算ユニット（CPU）の演算性能は，動作クロックの向上やクロックあたりの処理能力の向上よりも，コア数の増加による性能向上に重きが置かれている．そのため，一部のノートパソコン用CPUを除き，ほぼすべてのパソコン用CPUは2～6コアのマルチコアCPUとなっており，これらのパソコンでシングルスレッドのプログラムを実行させることは，演算能力を無駄に遊ばせていることに等しい．複数のコアを同時に働かせるためには，マルチタスクに対応したオペレーティングシステム（OS）上で，シングルスレッドのプログラムを複数実行するか，プログラムの内部でOSの提供するスレッドとよばれる機能を使い，並列処理のためのプログラムを作成する必要がある．スレッドを用いた並列処理では，逐次駆動から並列駆動へ移行するにあたってプログラムモデルの大きなパラダイムシフトが存在するため，逐次駆動型プログラミングにたけた者でも，いきなり並列駆動型プログラミングを行うのは困難である．そのような中で，既存のプログラムに少しの手間をかけることで並列計算を可能とするOpenMPとよばれる実装が話題を集めている．本稿では，このOpenMPを用いた並列プログラミングの基礎について述べる．OpenMP環境の設定方法と使用方法，および簡単な並列化とその際に気を付けるべき事項について記す．

1. OpenMPの導入

OpenMPとは，特定のプログラミング言語を指す言葉ではなく，ある共通化された並列化手法の実装の名前である．そのため，対応するコンパイラーであれば，ほぼ同様の記述で並列化が可能である．現在，GNUやインテル，PGI，マイクロソフトなどの主要なコンパイラーがOpenMPに対応している．本稿では，研究室内で最も多く利用されていると思われるMicrosoft Visual C++(MSVC)2008を対象に記述する．MSVC 2008のProfessional Edition以上でOpenMPに正式に対応しているが，Express Editionでも，別途ソフトウェア開発キット[1]をインストールすることでOpenMPが利用可能になる．OpenMP用ライブラリーのインストールが終了したら，次はプロジェクトの設定を行う．プロジェクトの新規作成，あるいは既存のプロジェクトの読み込みを行い，プロジェクトのプロパティー欄の「構成プロパティー」—「C / C++」—「言語」—「OpenMPサポート」を「はい」にする．こうすることでコンパイル時のコマンドラインに「/openmp」が追加され，OpenMPが有効化される．また，MSVCで「Windows フォームアプリケーション」を作成した際は，「構成プロパティー」—「全般」—「共通言語ランタイムサポート」を「共通言語ランタイム サポート（/clr)」にする必要がある．

2. OpenMPによる並列プログラミング

OpenMPの一番の特徴は，通常のシングルスレッドの逐次駆動型プログラムに2，3行追加するだけで並列駆動させることが可能なことである．リスト1にOpenMPを用いた並列処理のサンプルプログラムを示す．このプログラムは，0～1までのランダムな実数データ配列を，0～255の整数値へ正規化している．このような処理は実数データの画像化によく用いられる．リスト1において太字で示す行がOpenMPによる並列化のために追加された行であり，これらの行をコメントアウトしても，プログラムは逐次駆動型プログラムとして動作する．OpenMPで最もよく用いられるのが，13，17行目の「#pragma omp parallel for」節である．これは，「この節の次に現れるforループを並列化して実行せよ」という指示である．例えば2コアのコンピューター上でプログラムが実行されると，0～LoopNum-1までのforループが，0～LoopNum/2-1とLoopNum/2～LoopNum-1までの2つのforループに分解され，それぞれ別のスレッドに割り振られ並列に実行される．ループの分割数（＝スレッド数）は実行環境に応じて最適な分割数に自動的に設定されるが，omp_set_num_threads( ) 関数で設定することも可能である．デュアルコアのノートパソコンでテストしたところ，シングルスレッドの場合は1953 msであった実行時間が，2スレッド並列処理の場合は

```
1  #include <stdio.h>
2  #include <time.h>
3  #include <stdlib.h>
4  #include<omp.h>
5
6  int main(){
7
8          int LoopNum = 65535*512;
9          double *Data = new double[LoopNum];
10         double Max = 1,Min = 0;
11
12         clock_t Start = clock();
13 #pragma omp parallel for
14         for(int i=0;i<LoopNum;i++)
15                 Data[i]=(double)rand()/RAND_MAX;
16
17 #pragma omp parallel for
18         for(int i=0;i <LoopNum;i++){
19                 Data[i]=(int)(255.0*(Data[i]-Min)/(Max-Min));
20                 //printf("%lf,",Data[i]);
21         }
22         printf("¥nCPU time:%d ms¥n",clock()-Start);
23         delete Data;
24 }
```

リスト 1　サンプルプログラム 1.

1110 ms となり，約 1.8 倍高速化された．このように，OpenMP を用いると，非常に簡単にループの並列化が可能である．

サンプルプログラム 1 は，分割された各 for ループの間に依存性がないため，単純に #pragma 指示節を追加するだけで並列化が可能であったが，ループ間に依存性がある場合は並列化が困難であったり，不可能であったりする．例えば実数配列から最大値と最小値を求める場合，複数のループで同時に最大最小判定と値の書き換えが行われると，本来最大値となるはずだった値が，別のループで処理されていた値で上書きされてしまうことがある．リスト 2 に，並列処理に対応した，最大値と最小値を計算するプログラムを示す．本コードをサンプルプログラム 1 の 16 行目に挿入すると，データの画像化に際して画像のコントラストを最大化できる．リスト 2 のプログラムでは，スレッドごとに最大値と最小値を求め，最後にスレッドごとに排他的に最大値と最小値の判定と値の書き換えの処理を行っている．2 行目の「#pragma omp parallel」節は，「次の行を並列実行せよ」という指示であり，3〜15 行目のブロックが並列化されて実行される．4 行目のように並列化された領域の中で宣言された変数は，プライベート

```
1  Max=0,Min=1;
2  #pragma omp parallel
3  {
4          double TempMax=0,TempMin=1;
5  #pragma omp for
6          for(int i=0;i<LoopNum;i++) {
7                  if(TempMax<Data[i]) TempMax=Data[i];
8                  if(TempMin>Data[i]) TempMin=Data[i];
9          }
10 #pragma omp critical
11         {
12                 if(Max<TempMax) Max = TempMax;
13                 if(Min>TempMin) Min = TempMin;
14         }
15 }
```

リスト 2　サンプルプログラム 2.

変数とよばれ，スレッドごとに独立した変数となる．10 行目の「#pragma omp critical」節は，「次の行を排他的に処理せよ」という指示であり，11〜14 行目のブロックは，同時に実行されるのは 1 つのスレッドのみであることが保証される．このようなプライベート変数の扱い方や排他処理といった考え方は，並列プログラミングで必須となる考え方である．OpenMP は，逐次駆動型から並列駆動型へステップアップする形でプログラミングを学べるため，並列プログラミングの入門にも適している．

本稿では，OpenMP の特徴と基本的な並列化について解説した．なお，OpenMP では，上述のような for ループの並列化のみではなく，複数のスレッドにそれぞれ異なる処理を割り当てることも可能である．詳細は文献 2 を参照されたい．

（宇都宮大学　田北啓洋）

文　献

1）Windows SDK for Windows Server 2008 and .NET Framework 3.5: http://www.microsoft.com/downloads/details.aspx?FamilyId=E6E1C3DF-A74F-4207-8586-711EBE331CDC&displaylang=en
2）北山洋幸：“OpenMP 入門―マルチコア CPU 時代の並列プログラミング―”（秀和システム，2009）．