

並列化手法を用いた流況シミュレーションの高速化

解析事業部 防災情報部

菊野 拓人

原井光一郎

1. はじめに

科学技術計算において計算機性能の向上は解析モデリングの精緻化を果たす上で重要な要素となっており、その性能が飛躍的な向上をしてきたことは周知のことと思われる。例えば、マイクロプロセッサの性能は約 40 年の間に 32 万倍の性能向上が得られたという報告もある。¹⁾

この性能向上に比例して数値シミュレーションにかかる計算時間も自動的に短縮化が図れてきたことは容易に想像できる。しかしながら、このマイクロプロセッサの性能は、消費電力の問題等により、これまでの性能向上と比べると頭打ちの状態となっている。

一方、数値シミュレーションを扱うような実務面においては、計算領域の拡大、空間分解能(Δx)の細分化(小メッシュ、多層空間)、 Δx の細分化に伴う時間刻み(Δt)の細分化など計算容量および計算時間が増大するような要求事項が増えてきており、今後もその傾向は続くものと思われる。

これらの背景を踏まえると、効率よく計算を行うためには、ハード面での整備だけでなく数値シミュレーションに用いる計算プログラムを並列化していく等、ソフト面での整備が必要であると考えられる。

この並列化手法はこれまでに様々な方法が紹介されている(表 1)。代表的な手法として、PC クラスタのような分散メモリ型の計算機環境で実行可

能な MPI(Message Passing Interface)が挙げられる。また、CPU のマルチコア化が進み、共有メモリ型の計算機環境で普及した並列化手法として OpenMP がある。更に近年注目される手法として画像処理のハードウェアである GPU (Graphics Processing Unit)を数値計算に利用する GPGPU (General-Purpose computing on GPUs)が挙げられる。

以上のような背景を踏まえて、本稿では社内の計算機環境のように身近な計算資源で数値計算の高速化を図ることを目的とし、MPI および OpenMP を用いて既存の流況シミュレーションモデルの並列化を試みた。

表 1 並列化手法の例

メモリアーキテクチャ	名称
分散メモリ型	MPI
	HPF
	XcalableMP
共有メモリ型	OpenMP
	Pthreads
—	GPGPU

2. 計算モデル

本稿で並列化を試みる流況シミュレーションモデルは非圧縮性流体に対する Navier-Stokes の運動方程式と流体の連続式を基礎式とした二次元

多層非定常モデルによる海域の潮流計算モデルである。また、流入河川を考慮するため、水温、塩分に関しては Fick の拡散方程式を基礎式とする平面二次元拡散モデルを用いて同時に解析している。なお、解析モデルに用いられているプログラミング言語は fortran である。

モデル化の対象は東京湾全域とし、格子分割数は水平方向に(東西)152×(南北)208 格子、鉛直層分割数は 5 層とした。座標系は直角座標とし、格子幅(□x)は 300mである。図 1 に格子分割図を示す。

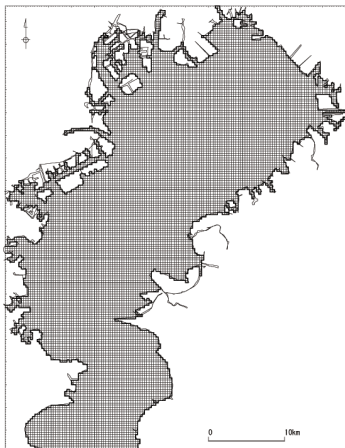


図 1 格子分割図

3. 並列化の特徴

並列計算は実行する計算機環境によって「プロセス並列」、「スレッド並列」の 2 種類に大別できる。プロセス並列は、クラスタなどの分散メモリ環境で処理が可能であり、代表的な手法として MPI がこれにあたる。一方、スレッド並列は共有メモリ環境で処理が可能であり、代表的な手法として OpenMP がこちらにあたる。

本稿では、これら MPI および OpenMP を用いて、

2 章で述べたシミュレーションモデルの並列化を試みた。なお、並列計算は並列実行するプロセッサ数に計算領域を分割すること(データ並列)で行った。以下に、各並列化手法について特徴およびプログラミングの留意点について述べ、その後に並列化の概要について説明する。

3. 1 MPI(Message Passing Interface)²⁾

プロセス並列による計算の流れのイメージを図 2 に示す。分散メモリ環境は各プロセッサ(CPU)がそれぞれ独立したメモリを保有し(この状態をノードという)、ネットワークを介してつながっている。この環境で並列計算を行うと、各ノードは割り当てられたデータ量について計算を実行するが、あるプロセスが別のプロセスのデータを参照するような場合、ノード間でデータを通信する必要がある(図3)。このようなノード間でデータ(メッセージ)の通信を行うための関数(ルーチン)をライブラリとして提供している代表的な規格が MPI である。

領域分割(データ並列)によって MPI 並列計算を行う場合、各ノードが担当する計算量は

$$(\text{全体のデータ量}) \div (\text{領域分割数})$$

となる。したがって、並列化により各ノードの計算量が節約され、かつそれらが並列に実行されるため計算の効率化が図れる。ただし、MPI プログラムの注意点として以下のことが挙げられる。上述のようにあるプロセスが別のプロセスのデータを参照する場合、MPI 通信関数を指定する必要があるが、この関数を正しく指定しなければ誤った解を算出するおそれがある。また、MPI によって効率よく計算を行うためには、このデータの通信量を減らすようなプログラミングが必要である。したがって、対象

とする計算の規模(格子数、タイムステップ等)、地形形状などによってデータの分割方法を考慮する必要がある。さらに、各ノードに割り当てるデータ量に差があると、ロードバランスが不均等になり効率的な計算が行えない。したがって、各ノードに割り当てるデータにできるだけ偏りが無くなるような領域分割が必要である。

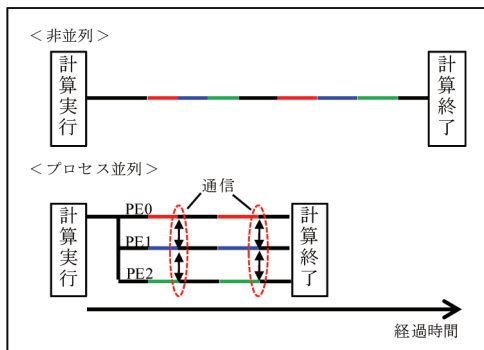


図2 プロセス並列のイメージ

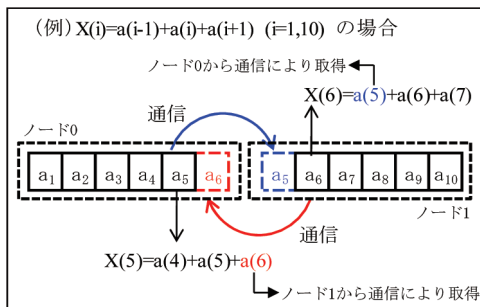


図3 データ通信の例

3.2 OpenMP³⁾

OpenMP はプログラム中にコンパイラへの並列化指示文を挿入(スレーブスレッドの生成)することで並列計算が可能になる。また共有メモリ上で計算を行うため、各プロセスは同じメモリ上のデータにアクセスでき、MPI のようにノード間のデータ通信を意識したプログラミングの必要がない。ただし、

スレーブスレッドを生成することで、すべての変数は各スレッド内だけでアクセスできる変数(プライベート変数)もしくはすべてのスレッドで共有な変数に分類されるため、各変数がどちらにあたるかを定義する必要がある。

OpenMP プログラム(スレッド並列プログラム)の計算の流れを図4に示す。計算を実行するとマスタースレッドによりファイルの入力など逐次計算が行われる。そして、OpenMP 指示文を挿入した部分に達すると、予め設定した数のスレッドが立ち上がり並列計算を実行する。並列計算が終了するとマスタースレッドに戻り逐次計算に戻る。したがって、計算負荷の大きい do ループのみを並列化するなど、ユーザーによって意図的な並列化が可能である。ただし、do ループのたびに指示文を挿入するなど計算粒度が小さくなる並列化は計算効率を悪くするおそれがある。これは、スレッドの立ち上げが計算上のオーバーヘッドとなるため、粒度が小さければオーバーヘッドも増大するためである。したがって、OpenMP による並列化は、計算粒度をなるべく大きくするプログラミングが必要となる。(ただし、計算粒度が粗すぎると、計算負荷の不均衡により計算性能が低下するおそれがある。)

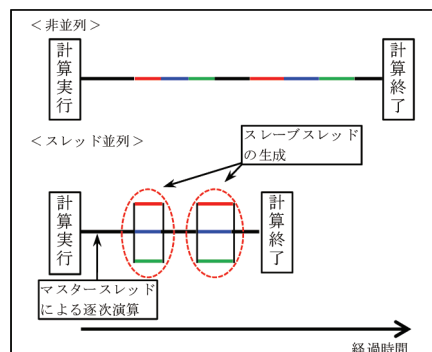


図4 スレッド並列のイメージ

3.3 並列化概要

並列計算の方法は、並列プログラミングが比較的容易であること、かつ効果的な計算速度の向上を期待してデータ並列による計算を採用した。データ並列による計算は、各プロセス(スレッド)に担当するデータを割り当てることで行う。本稿では、計算領域を並列実行するプロセス(スレッド)数に分割し、分割した領域のデータを各プロセス(スレッド)に割り当てることで並列計算を行うようにした。ここで計算領域の分割は、図 1 に示した計算範囲を y 方向(南北方向)に分割することで行った。これは、MPI は 3.1 節で述べたように、隣り合う領域と接続する境界位置のデータを互いに通信する必要があり、本稿のモデル領域が南北方向に格子数が多い配置となっているため、このように分割することで通信量を抑えることができると考えられるためである。ここで領域を x, y 方向それぞれ 2 分割した場合の通信量のイメージを図 5 に示す。

OpenMP では、プログラム内の y 方向に関する `do` ループを並列に実行するよう指示文を挿入し、計算を行った。

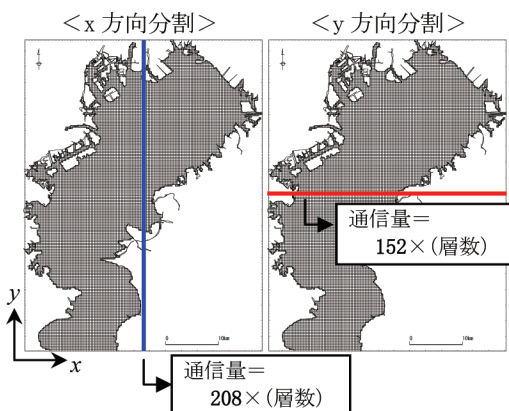


図 5 データ通信量

4. 並列化による性能検討

4.1 検討条件

計算を実行するにあたり、使用した計算機環境を表 2 に示す。計算はマルチコア PC(1 台)の環境下で実施した。まず、基準となる計算結果算出のために 1 プロセスのみを使用した計算を実行した。並列化による性能検討のために実施した並列計算のプロセス数(スレッド数)は MPI、OpenMP 共に 2,4,8 プロセス(スレッド)とした。

表 2 計算環境一覧

OS	Windows Vista Ultimate SP1
CPU	Core i7 CPU 870 2.93GHz
RAM	8.00GB
コンパイラ	Intel Fortran 11.0
MPIライブラリ	Intel MPI Library 4.0

並列化に伴い、各手法の計算精度の検証を行い、計算時間の変化について検討した。計算精度は、MPI はデータの通信が不十分な場合、OpenMP は変数の扱い方(プライベート/共有変数の指示)などに誤りがある場合、逐次計算を進めると誤差(1 プロセスの結果との差)が蓄積し、最終的に過った解を得ることになる。したがって、基準となる 1 プロセスの計算結果と並列化によって得られる計算結果との差を確認した。並列化によって得られた結果に問題が無いことを確認した上で、計算速度の比較による並列化の性能検討を行った。流況シミュレーションの計算条件を表 3 に示す。

表 3 計算条件

対象潮汐	平均大潮 (M2+S2分潮) (=1周期あたり12時間)
計算期間	助走計算期間：12時間 (1周期) 周期計算期間：60時間 (5周期) 計 72時間
タイムステップ	2.0 秒

