B-022

統合陸域モデル ILS の Code Modernization

Code modernization on the integrated land simulator ILS

荒川隆†　原山卓也†　新田友子ǂ　竹島滉§　芳村圭‡

Takashi Arakawa　Takaya Harayama　Tomoko Nitta　Akira Takeshima Kei Yoshimura

1. はじめに

現代の高性能計算機にとって並列性能は演算性能を考える上で極めて重要なファクターである。計算機の並列性はノードやコアといった複数の階層に分けられるが、本研究では最下層に位置する SIMD（あるいはベクトル演算）に着目する。インテルアーキテクチャでは SIMD は拡張命令 AVX で利用可能なようになっており、AVX のヴァージョンが上がるにつれてベクトル長が長くなっている。アプリケーションソフトウェアを SIMD に対応させることをインテルでは Code Modernization と呼び、プロセッサの能力をフルに発揮させるために Code Modernization を推奨している。そこで本研究では具体的なアプリケーションソフトウェアを SIMD に対応させ性能を評価した。対象としたソフトウェアは統合陸域モデル ILS である。以下、ILS のプログラム構造、改良の詳細、性能評価について述べる。

2. ILS のプログラム構造

先述の通り、対象としたソフトウェアは統合陸域モデル ILS である。ILS については FIT2017 で全体構造と単体性能について[(1)]、FIT2018 で大気モデルとの結合について[(2)]報告した。今回改良の対象としたのは ILS のコンポーネントモデルのうち、陸面モデル MATSIRO[(3)]である。気象・気候シミュレーションにおいて陸面を表現する方程式系は空間軸として鉛直方向のみを扱い水平依存性を持たないのが一般的であり、MATSIRO の方程式系も基本的に鉛直軸のみで構成されている。ILS は方程式系の表現に適合するように、鉛直方向の計算を行う物理モジュールと鉛直カラムを水平面に広げるドライバモジュールの2つの階層からなる。この様子を模式的に表したのが図1である。上図はドライバ部であり、ここではグローバル変数 x は（鉛直、水平面）の次元をもつ。ドライバ部で水平ループ do i = 1, NXY が回り、このループの中で MATSIRO のサブルーチン matsiro_sub がコールされる。変数は x(:,i)のように鉛直1次元の引数として渡される。下図に示した計算部分では k = 1, NZ のループの中で物理的な計算が実行される。鉛直層数 NZ の数は計算条件によって異なるが、大気モデルの下部境界として用いられる場合は5程度である。また陸面の各プロセスのうち鉛直ループを有するプロセスはむしろ少数であり、MATSIRO の多くのサブルーチンはループ構造を持たないスカラ演算を行っている。

```
subroutine ils_driver
 real(kind=8) :: x(NZ, NXY)
 do i = 1, NXY
    call matsiro_sub(X(:,i))
 end do
end subroutine
```

```
subroutine matsiro_sub(x)
 real(kind=8) :: x(NZ)
 do k = 1, NZ
   x(k) = f(x,k)
 end do
end subroutine
```

図 1　MATSIRO のプログラム構造

3. ILS の改良

前項で述べたように、ILS の MATSIRO では鉛直ループが最内周となるが、この計算は依存性があるため SIMD が効かず、このことが計算性能の阻害要因となっていた。そこで図2に示すように、汎用的に用いることのできる次元 ud を新たに設け、このループを最内周とするようにした。

```
subroutine ils_driver
 real(kind=8) :: x(UDMAX, NZ, NXY)
 do i = 1, NXY
    call matsiro_sub(X(:,:,i))
 end do
end subroutine
```

```
subroutine matsiro_sub(x)
 real(kind=8) :: x(UDMAX, NZ)
 do k = 1, NZ
  do ud = 1, UDMAX
    x(ud,k) = f(x,ud,k)
  end do
 end do
end subroutine
```

図2　改良した MATSIRO のプログラム構造

† 高度情報科学技術研究機構
§ 東京大学大学院工学系研究科
ǂ 東京大学生産技術研究所

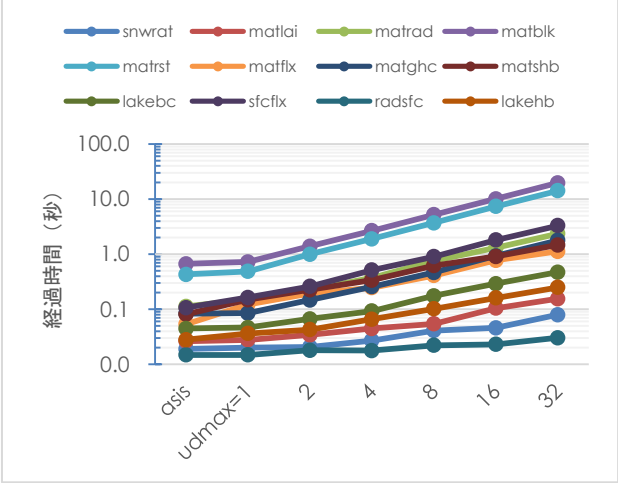
4. 性能評価

4. 1 Intel プロセッサでの性能評価

はじめに RIST 所有の Intel マシンで性能を評価した。評価は udmax を1から増加させ実行時間を測定する方法で行った。用いた命令セットは AVX2、コンパイラはインテルコンパイラの 2018 年以降のものである。CPU は Intel Xeon CPU E5-2697 である。この CPU の倍精度実数のベクトル長は4であるため、理想的な条件では実行時間は udmax = 4 までは udmax = 1 と同程度になる。MATSIRO の計算は matsiro_flux と matsiro_step の2つのサブルーチンコールに大別され、それぞれのサブルーチンが更に複数のサブルーチン群をコールしている。そこで、matsiro_flux と matsiro_step の各々について各部の実行時間を図3に示す。実行時間の短いサブルーチンでは udmax に対する時間増加が緩やかで性能向上が見られるが、実行時間の長いサブルーチンでは udmax に対して時間が直線的に増加しており SIMD 化の効果が得られていない。コンパイル情報を見ると、これらの効果のなかったサブルーチンでは、べき乗演算や関数呼び出しがあるためベクトル化できなかったというメッセージがあり、本来速度向上を要求される複雑な演算コードで Code Modernization の効果が得られていないことがわかる。

4. 2 地球シミュレータでの性能評価

地球シミュレータは最大ベクトル長が 256 と大きく、Intel コンパイラでの評価方法であるベクトル長 4 程度ではオーバーヘッドが大きいためベクトル化されない。つまり Intel プロセッサの場合と同じ実験はできない。そこで、UDMAX = NXY, NXY = 1 とし、最内周の ud ループで水平格子の計算を行うようにした。水平格子の大きさは領域分割された各プロセスで 6000 程度であり、十分なベクトル長が得られている。ベクトル化前後の各部の実行時間を図4に示す。ベクトル化前最も実行時間の長かった gwfpar は 53 秒から 0.3 秒と 170 倍高速化された。他の計算ルーチンも大幅に高速化されており、ベクトル化の効果が得られたことがわかる。



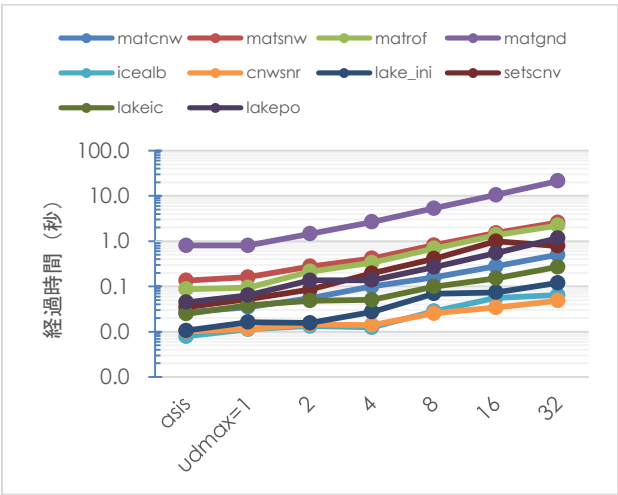


図3 Intel プロセッサでの時間測定結果

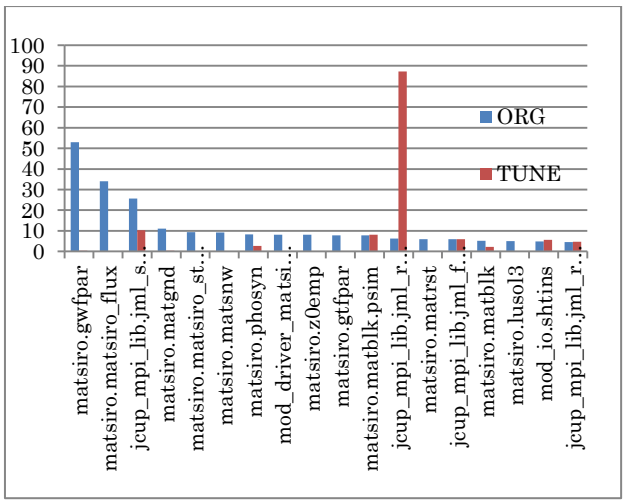


図4 地球シミュレータでの時間測定結果

(1)荒川隆、新田友子、鳩野美佐子、芳村圭：陸域統合モデル ILS とその結合について、第 16 回情報科学技術フォーラム、東京大学、2017 年 9 月 14 日
(2)荒川隆、新田友子、鳩野美佐子、竹島滉、芳村圭：統合陸域モデル ILS と大気モデルの結合、第 17 回情報科学技術フォーラム、福岡工業大学、2018 年 9 月 19 日
(3)Takata, K., S. Emori, and T. Watanabe: Development of minimal advanced treatments of surface interaction and runoff. Global Planet. Change, 38, 209–222, doi: 10.1016/S0921-8181(03)00030-4.