RRI モデルの最適化による計算実行時間短縮に関する検討

国土交通省中国地方整備局企画部　正会員　○村岡　和満
山口大学大学院　正会員　朝位　孝二

１．はじめに

RRI（Rainfall Runoff Inundation model）は，流出と氾濫を一体的に解析でき，「流域治水」を実現するための「水系・流域が一体となった洪水予測モデル」の候補として注目されている．近年の大規模水害を踏まえ，さらなる予測精度向上が求められている．精度向上には計算負荷が伴う一方，我が国の洪水予測は 10 分間隔に予測値が更新されており，データ通信や表示処理にかかる時間を考慮すると，少なくとも 5 分以内に計算を完了させなければならない. RRI は，氾濫が発生しない程度の洪水の予測計算実行は数分で完了するが，平坦な地形において大規模氾濫が発生する場合，解が不安定となるため，数値解析のステップサイズと許容誤差を小さく設定する必要があり，計算実行時間が極端に長くなる問題がある．

本研究では，RRI の高速化を目的として，ソースコードの簡単な修正，コンパイラオプション設定の有無, 共有メモリ並列化手法の OpenMP コア数変更，プログラムの GPU 化による実行時間短縮の効果について，3 つの異なるシステムを用いて比較検証した．

２．最適化手法と検証ケース

2.1 ソースコードの修正

RRI は河道の解析に一次元拡散近似方程式，斜面（氾濫原）の解析に二次元拡散近似方程式を採用しており，それぞれの方程式について 4-5 次の Cash-Karp Runge-Kutta 法を用いて初期値問題を数値による近似で解いている．繰り返し計算が多発することから，演算時間を短縮する比較的簡単な方法として，do ループ内部にある 1 回の演算に時間のかかる除算を乗算に変換するとともに，固定値の演算については, do ループの外側に移動する修正を行った．

また，最も実行時間の長い斜面の解析コードを GPU 化したケースも作成した．

2.2 コンパイラオプション

ソースコードのコンパイルには Intel Fortran コンパイラを用いた．デフォルトの最適化オプション(`-O3 -qopenmp`)を基本とした実行ファイルと，インライン展開を行うオプション(`-ipo`)，およびプロセッサで利用可能な最上位の命令セット向けのコードを生成するオプション(`-xHost`)を追加した実行ファイルをそれぞれ作成した．

2.3 OpenMP コア数の変更

RRI には，プログラムの並列化として OpenMP が実装されている．そこで，並列化コア数を 1 コアからそれぞれのシステムが持つ CPU に実装された最大コア数まで変化させて，実行時間の短縮効果について検証した．

2.4 比較検証モデルの選定

比較検証に用いたシステムを**表-1**，モデルの諸元とケースを**表-2** に示す．システム A は九州大学のスーパーコンピュータ，システム B は一般的な Linux サーバ，システム C は Macbook Pro でアップルシリコンを搭載している．RRI のモデルとしてサンプルで梱包されているインドネシア Solo 川モデル Solo30s，大規模氾濫を有するモデルとしてバングラデシュの Meghna 川モデル Meghna15s を用いた．

表-1 計算システムの諸元

	システムA	システムB	システムC
CPU	Intel Xeon Gold 6154 (Skylake-SP) 3.0 GHz (Turbo 3.7 GHz) 18 core × 2 / node	Intel Xeon E5-1620 v3 3.5GHz (Turbo 3.6 GHz) 4 core	Apple M1 3.2GHz, 高性能4 core 高効率4 core
Memory	192GB Memory / node	48GB Memory	16GB Unified Memory
メモリ帯域幅	255.9 GB/sec / node	68 GB/s	68.2GB/s（推定値）
GPU	Nvidia Tesla P100	Nvidia Quadro K2200	Apple独自設計 8 core
実行モデル	Solo30s Meghna15s	Solo30s	Solo30s
実行ケース	Case 1 ～ 5	Case 1 ～ 5	Case 1 ～ 4
OMPコア数	1, 2, 4, 6, 8, 12,16, 32	1, 2, 4, 6, 8	1, 2, 4, 6, 8

表-2 モデルの諸元と比較ケース

検証モデル	Solo30s	Meghna15s
メッシュサイズ	271 x 166 (約900m)	1154 x 720 (約450m)
再現計算期間	2007/12/24 ～ （360時間）	2007/7/7 ～ （24時間）
許容誤差（eps）	0.01	0.0001
Case1	オリジナルソースコード	
Case2	ソースコードの修正	
Case3	Case1 にコンパイラオプション（-xHost -ipo）追加	
Case4	Case2 にコンパイラオプション（-xHost -ipo）追加	
Case5	CUDA FortranでGPUコード化	

キーワード　RRI，最適化，並列化，OpenMP，Xeon，Apple M1，GPU
連絡先　〒730-8530 広島市中区上八丁堀 6-30　TEL: 082-221-9231　E-mail: muraoka-k87km@mlit.go.jp

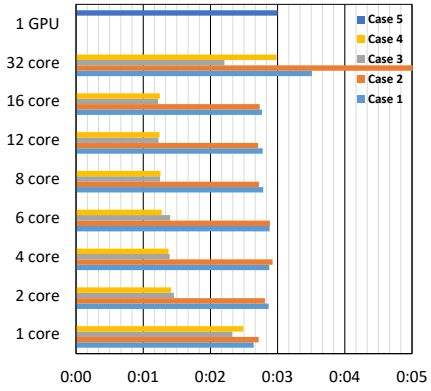

図-1 Solo30s (システム A)の実行結果

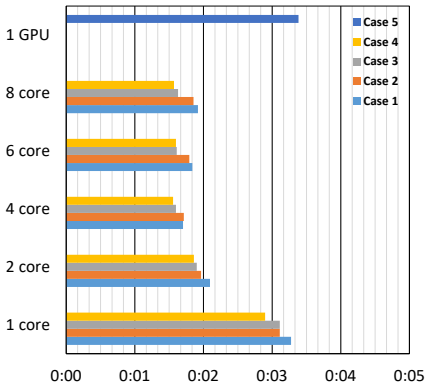

図-2 Solo30s (システム B)の実行結果

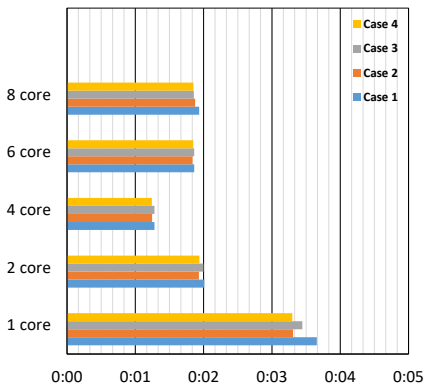

図-3 Solo30s (システム C)の実行結果

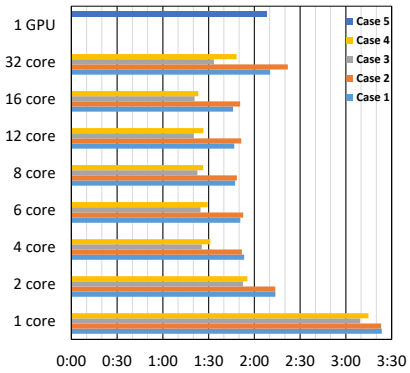

図-4 Meghna15s (システム A)実行結果

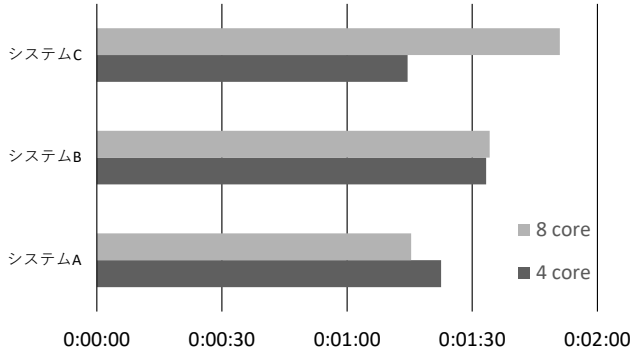

図-5 Solo30s のシステム A,B,C の性能比較

３．実行結果比較と考察

3.1 コード修正による最適化効果

図-1～図-4 に各システムでの実行結果を示す．実行時間計測は time コマンドを使用した．どのシステムにおいても数秒程度の短縮で，除算を乗算に変更しても顕著な改善は得られなかった．これは，-no-prec-div オプションがデフォルトで，コンパイラが自動的に除算を乗算に変換していることが原因と考えられる．GPU コード化した case5 についても顕著な改善は得られず，OpenMP のシングルコアとほぼ同程度の実行時間であった．

3.2 コンパイラオプションによる最適化効果

インライン展開を行うオプション(-ipo)を追加した場合，システム A 及び B で実行時間短縮の効果が得られた．特にシステム A での効果が顕著であり，Solo30s では 2.0～2.3 倍，Meghna15s では 1.1～1.3 倍高速化した．なお，システム C は MacOS であり，コンパイラに-ipo オプションがサポートされていないため，実行時間短縮の効果は得られなかった．

3.3 OpenMP 並列化コア数による最適化効果

OpenMP による並列化はどのシステムにおいても 4 コアで頭打ちとなり，1.7～2.9 倍程度高速化した．6 コア以上ではほとんど変化が無く，または計算速度が低下する副作用が発生し，16 コアが限界であると考えられる．図-5 にシステム A, B, C の性能比較を示す．システム A の 8 コアとシステム C の 4 コアでの性能がほぼ同じであることは特筆すべき結果である．

４．まとめ

RRI の OpenMP による高速化は 4 並列以上で頭打ちとなり，システム A と C で計算時間が最も短くなり，大規模氾濫を有するモデルでも同様であった．また，インライン展開を行うと計算実行速度が向上し，特にシステム A で効果が顕著であった．今後，さらなるチューニングを実施していく予定である．

謝辞

RRI の計算は，九州大学情報基盤研究開発センターのスーパーコンピュータ ITO を利用しました．

参考文献

1) 佐山・建部・藤岡・牛山・萬矢・田中：2011年タイ洪水を対象にした緊急対応の降雨流出氾濫予測，土木学会論文集B1（水工学），2013.
2) 村岡和満：南アジア・東南アジアの気象水文量の長期変化と水災害リスクに関する研究，山口大学博士論文，2021.