# メニーコア計算機環境でのアジョイント法による データ同化における時空間ブロッキングの性能評価

藤川 隼人† 池田 朋哉‡ 片桐 孝洋†3 永井 亨†3 荻野 正雄†3
名古屋大学大学院情報学研究科情報システム学専攻†
名古屋大学大学院情報科学研究科情報システム学専攻‡
名古屋大学 情報基盤センター†3

## 1. はじめに

大規模数値シミュレーションと大容量観測データの融合を狙う計算技術としてデータ同化が注目されている．このデータ同化の手法としてアジョイント法があるが，フェーズフィールドモデルのような大規模で自由度の高いデータを扱う際のアジョイント法適用の問題点の 1 つとして，メモリアクセスの増大による演算効率の低下がある．この問題に対し，アジョイント法のステンシル計算に対し時空間ブロッキング[1]を行うことで高性能化する手法が提案されている[2]．

本研究では，時空間ブロッキングをメニーコア計算機上で動作させて性能パラメータを変化させることによるチューニングの効果を調べる．

## 2. 時空間ブロッキング

アジョイント法における時空間ブロッキングとは，空間方向だけでなく時間方向にもブロッキングを施すことにより，キャッシュのデータを再利用しつつ，次のステップの計算を可能にする考え方である．

大規模で自由度が高いモデルでは，近傍の格子点値のアクセス範囲がキャッシュ容量を超え，演算効率が低下する問題がある．この問題が生じる原因は，空間方向への計算を先に行うため，次のステップの計算を行うときにはキャッシュにデータが残っていないことによる．そこで，時間方向への計算を先行的に行い，キャッシュにデータをできるだけ保持するように計算を進めることで，キャッシュ上のデータを再利用することができる．

## 3. 性能評価

性能評価は名古屋大学情報基盤センター設置の Fujitsu PRIMEHPC FX100, Fujitsu PRIMERGY CX400, および東京大学情報基盤センター設置の Oakforest-PACS（メニーコア計算機）を用いる．このことで，メニーコア計算機環境の性能の特徴を明らかにする．フェーズフィールドモデルにおいて，初期値を推定する問題をベンチマークとし，以下の問題設定で行った．また，時空間ブロッキングサイズを変化させて実行時間を測定した．

表 1. 問題設定

| | |
|---|---|
| 格子点数 | 1600 × 1600 |
| 初期値 | すべて 0.2 |
| 真の値 | 乱数[0,1]の 0.1 刻み |
| 時間ステップ数 | 128 |

変化させたスレッド数は，FX100 では 1,4,16,25,32, CX400 では 1,4,14,16,25,28 であり，ソケットを跨ぐ NUMA の影響を考慮したスレッド数を設定した．また，Oakforest-PACS では，1，68，136，204，272 と物理コア数（68 コア）を超えるスレッド数を指定した．

FX100 では，計測を行ったすべてのスレッド数に対して高速化を得ることができた．ブロッキングサイズを変化させた結果，最大で 1.5 倍の速度向上が得られた．最高速のスレッド数は 16 であったため，スレッド数を増やすほど速度向上が得られたわけではない．

CX400 では，時空間ブロッキングサイズを変化させると最大で 1.6 倍の速度向上が得られた．しかし，スレッド数が 25 と 28 の際は時空間ブロッキングを行うと逆に速度が遅くなるという結果が得られた．ラストレベルキャッシュミス率を測定した結果，時空間ブロッキングを施すとキャッシュミス率が高くなっていた．そのため，1 ソケット 14 コアを超えるスレッド実行では，NUMA 構成によるメモリアクセスが影響していると考えられる．

一方，Oakforest-PACS では，測定したスレッド数全てに対して時空間ブロッキングによる速度向上が得られた．時空間ブロッキングによる速度向上は最大で 1.3 倍だった．スレッド数を増やすほど高速になるため，最大のスレッド数 272 の際は naïve な実装と比較して 28.9 倍の速度向上を得ることができた．

## 4. まとめ

実験の結果，FX100 と Oakforest-PACS では時空間ブロッキングによる高性能化が得られることが明らかになった．一方，CX400 では，ソケットを跨ぐことにより時空間ブロッキングの効果が無くなった．詳しい性能分析の結果は当日発表する．

**謝辞** 本研究の一部は，科学技術研究費補助金基盤研究（B）「通信回避・削減アルゴリズムのための自動チューニング技術の新展開」（課題番号：16H02823），および JSPS 二国間交流事業共同研究（オープンパートナーシップ）「国際交流による自動チューニングのための性能モデルの深化」による．

## 参考文献

[1] K. Datta, et.al., Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures, Proc. of the 2008 ACM/IEEE conference on Supercomputing (2008)

[2] 池田ほか，アジョイント法における Forward model への階層ブロッキング適用による高性能化，情報処理学会研究報，Vol.2016-HPC-157, No.17 (2016)