第１０回 自動チューニング技術の現状と応用に関するシンポジウム(ATTA2018)
●日時： 2018年12月25日（火）10:30～17:30
●場所：東京大学 弥生講堂 一条ホール
16:15～16:45 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

科研基盤B採択課題（2016年度～2018年度）

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

名古屋大学情報基盤センター　片桐孝洋

背景

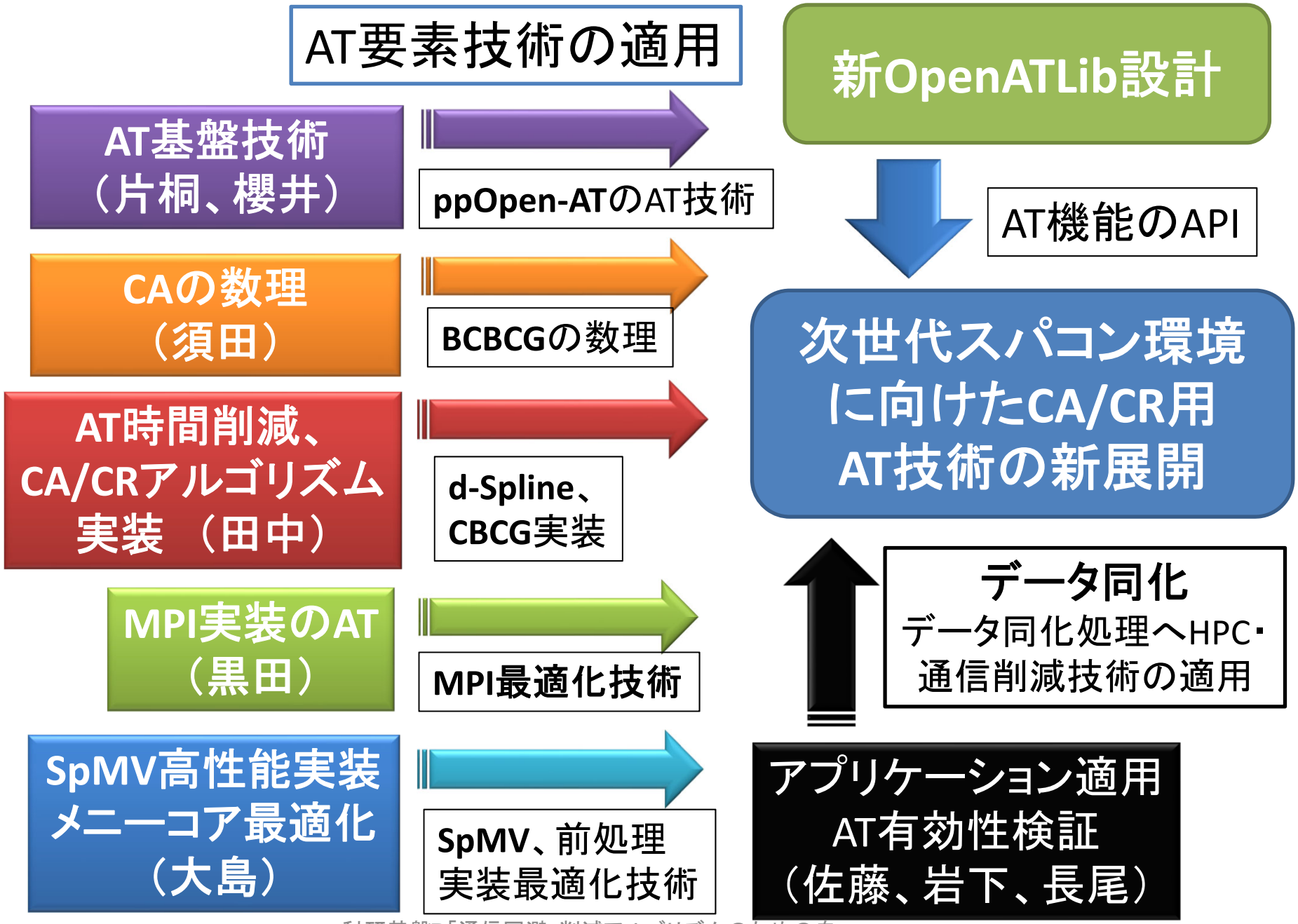
- 次世代スパコンでは、ノード数が10 万～100 万ノード級と想定され、高い並列性の確保と通信時間の削減が強く要請される。
- 特に通信時間については、通信レイテンシの削減がハードウェア上困難であることが知られている。
- そのため、通信回避（Communication Avoiding (CA)）、および、通信削減（Communication Reducing (CR)）アルゴリズムによる数値計算ライブラリの再構築が必須となる。
- 利便性の向上と多様な入力や計算機環境での高性能化のため、アプリケーションの性能パラメタを対象計算機のキャッシュサイズ、コア数、通信性能などの計算機アーキテクチャの特性に加えて、数値アルゴリズム選択に至る広範な要因を自動的にチューニングするソフトウェア自動チューニング技術（AT 技術）が、国内外から注目を集めている。

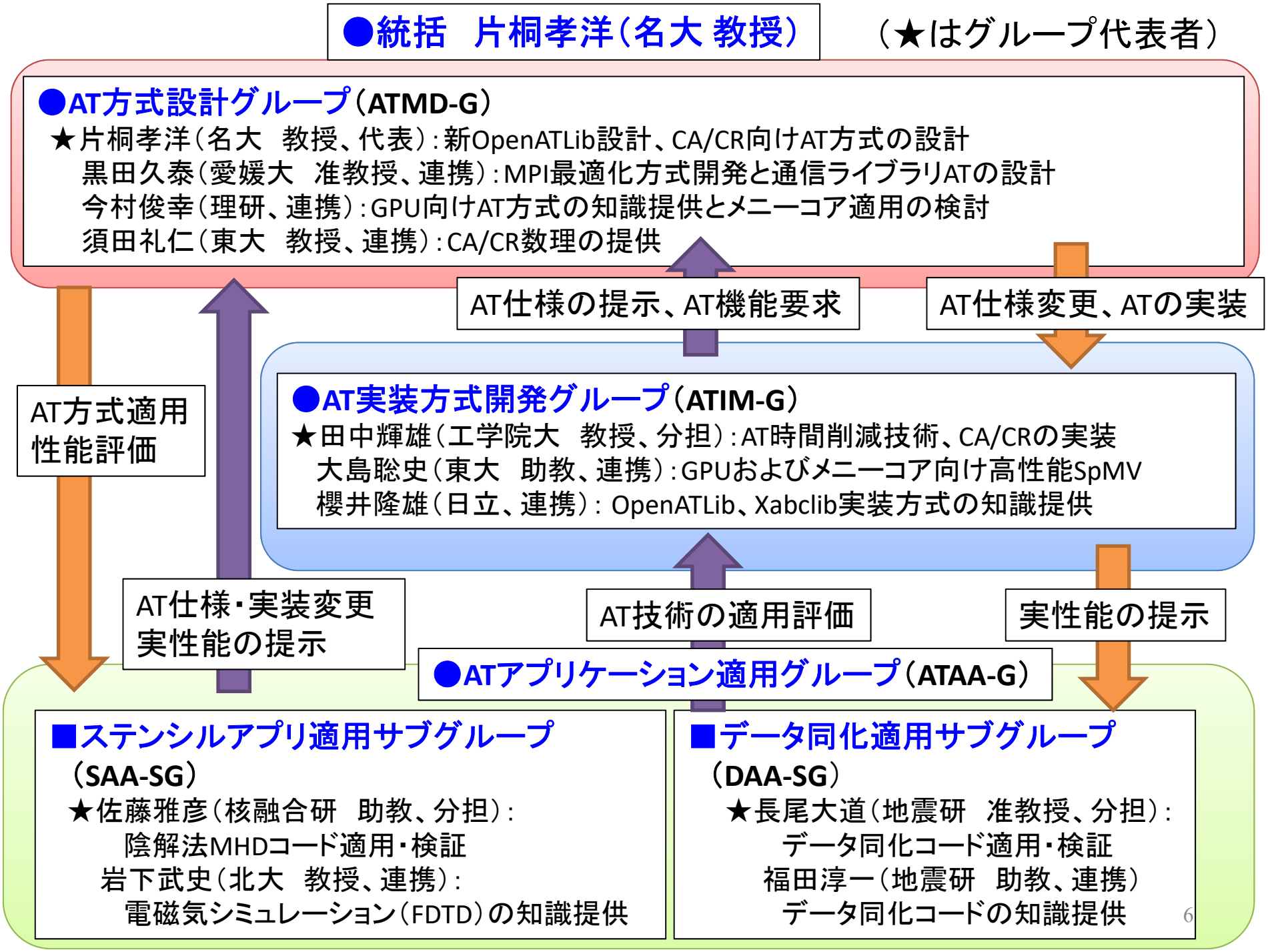
研究期間内に何をどこまで明らかにしようとするのか

1. CA/CR アルゴリズムで現れる性能パラメタと、そのAT 実装のフレームワーク
2. CA/CR アルゴリズムのためのハイブリッドMPI/OpenMP 実行における高性能実装方式
3. 1、2 の処理で利用可能なAT 時間削減方式
4. プラズマシミュレーションおよびデータ同化処理の「実用コード」において、開発したAT を用いて達成できる高性能化の原理

研究計画・方法

- <AT 方式設計グループ>、<AT 実装方式開発グループ>、<AT アプリケーション適用グループ>の3グループを編成し、CA/CR のためのAT 機能のアルゴリズム、API 仕様策定、実装、および、性能評価を機能的に実施する。
- 開発工程を5つのフェーズに分け、CA/CR のためのAT 方式を開発する。


AT要素技術の適用
新OpenATLib設計
AT基盤技術
（片桐、櫻井）
ppOpen-ATのAT技術
AT機能のAPI
CAの数理
（須田）
BCBCGの数理
次世代スパコン環境
に向けたCA/CR用
AT技術の新展開
AT時間削減、
CA/CRアルゴリズム
実装 （田中）
d-Spline、
CBCG実装
データ同化
データ同化処理へHPC・
通信削減技術の適用
MPI実装のAT
（黒田）
MPI最適化技術
SpMV高性能実装
メニーコア最適化
（大島）
SpMV、前処理
実装最適化技術
アプリケーション適用
AT有効性検証
（佐藤、岩下、長尾）


●統括　片桐孝洋（名大 教授）
（★はグループ代表者）
●AT方式設計グループ（ATMD-G）
★片桐孝洋（名大　教授、代表）：新OpenATLib設計、CA/CR向けAT方式の設計
黒田久泰（愛媛大　准教授、連携）：MPI最適化方式開発と通信ライブラリATの設計
今村俊幸（理研、連携）：GPU向けAT方式の知識提供とメニーコア適用の検討
須田礼仁（東大　教授、連携）：CA/CR数理の提供
AT仕様の提示、AT機能要求
AT仕様変更、ATの実装
AT方式適用
性能評価
●AT実装方式開発グループ（ATIM-G）
★田中輝雄（工学院大　教授、分担）：AT時間削減技術、CA/CRの実装
大島聡史（東大　助教、連携）：GPUおよびメニーコア向け高性能SpMV
櫻井隆雄（日立、連携）：OpenATLib、Xabclib実装方式の知識提供
AT仕様・実装変更
実性能の提示
AT技術の適用評価
実性能の提示
●ATアプリケーション適用グループ（ATAA-G）
■ステンシルアプリ適用サブグループ
（SAA-SG）
★佐藤雅彦（核融合研　助教、分担）：
陰解法MHDコード適用・検証
岩下武史（北大　教授、連携）：
電磁気シミュレーション（FDTD）の知識提供
■データ同化適用サブグループ
（DAA-SG）
★長尾大道（地震研　准教授、分担）：
データ同化コード適用・検証
福田淳一（地震研　助教、連携）
データ同化コードの知識提供

ディープラーニングを用いた数値計算ライブラリにおける反復解法の前処理選択の検討

2018年9月3日(月)～5日(水)
日本応用数理学会 2018年度 年会、名古屋大学

November 28th, 2018, CANDAR'18@Hida Takayama,
K. Yamada (M2, Nagoya U.), T. Katagiri (Nagoya U.),
H. Takizawa (Tohoku U.), K. Minami (RIKEN), M. Yokokawa (Kobe U.),
T. Nagai, M. Ogino (Nagoya U.)
（**Best Workshop Award受賞論文**）

※本成果は滝沢基盤Bの成果でもあります。
※AIによるAT方式のAT言語への展開が片桐基盤Bの成果です。

アウトライン

- 研究背景
- 研究目標
- 実験手順
- 実験結果
- まとめ・今後の予定

研究背景

- 疎行列が対象の数値計算ライブラリは様々な分野で利用
  - ✓実装選択に関するチューニングパラメタは多く存在

➢実行を介さず最適化できれば，多くの分野における高性能化のコストが大きく削減可能
  - ✓疎行列ベクトル積(SpMV)を対象とする人工知能による最適実装選択(Cuiら ’16**)

✓ 数値計算ライブラリそのものを対象とした人工知能を用いた最適実装選択にはあまり前例がない

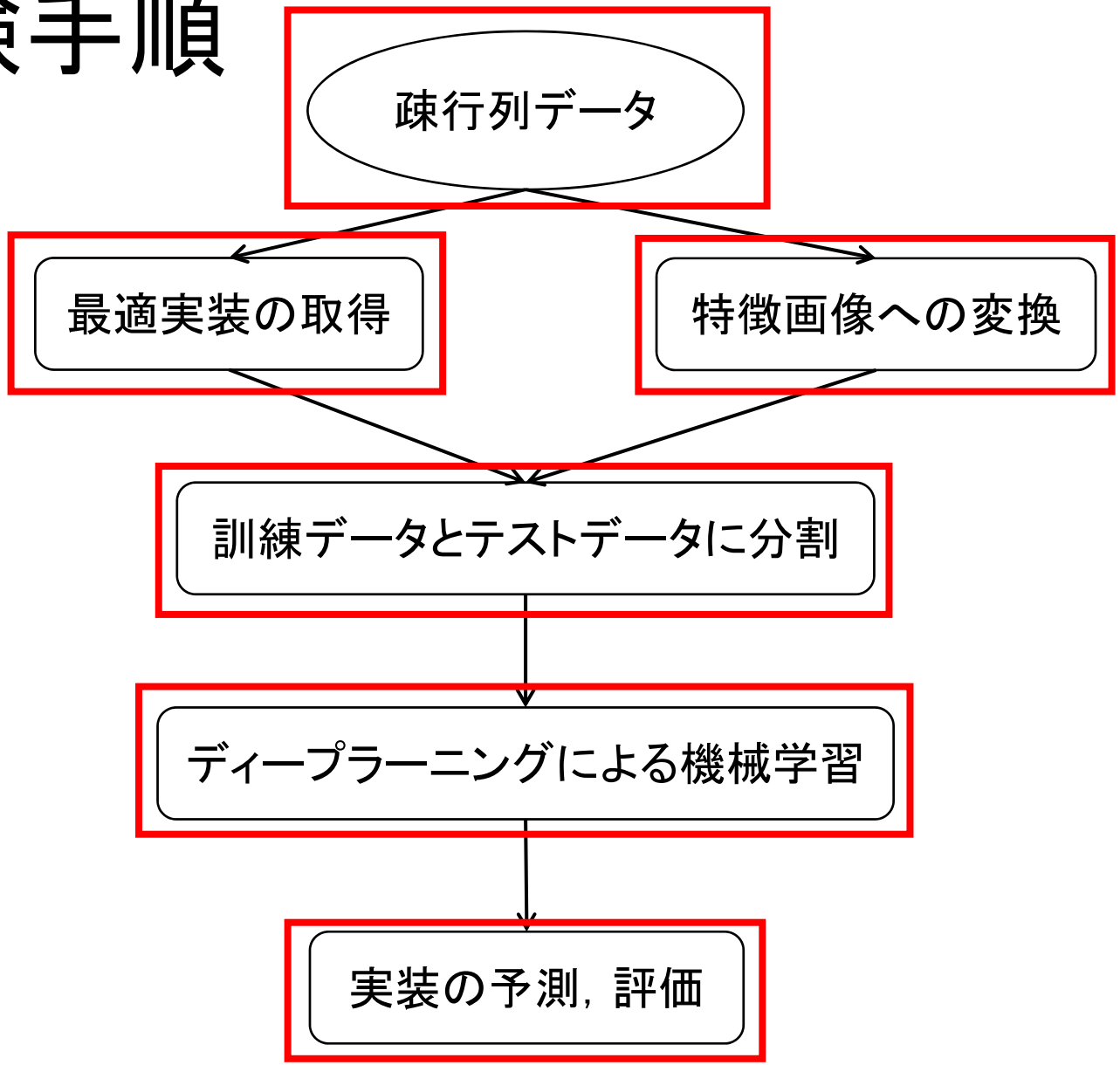
**Hang CUI, Shoichi HIRASAWA, Hiroyuki TAKIZAWA and Hiroaki KOBAYASHI, A Code Selection Mechanism Using Deep Learning, 2016 IEEE 10th International Symposium on Embedded Multicore/Many-core Systems-on-Chip, pp. 385-392.

研究目標

- 疎行列ベクトル積や反復解法の専門家でなくても最適な実装方式が自動指定できる数値計算ライブラリの構築を目指す
  - ✓提案手法：Cuiらの手法(既存手法)を拡張して数値計算ライブラリに適用する

➢専門家による実装選択と同等か許容範囲内の実装が自動選択可能かどうか実験を通して検討を行う
  - ✓対象数値計算ライブラリ:Xabclib_GMRES
  - ✓対象行列データ:フロリダ大学疎行列コレクション内の正方行列

実験手順


疎行列データ
最適実装の取得
特徴画像への変換
訓練データとテストデータに分割
ディープラーニングによる機械学習
実装の予測，評価

最適実装の取得

- 各疎行列(約4000個)に対してXabclib_GMRESを実行**
  - ✓収束解が得られなかった疎行列は使用データから除外
  - ➢最終的に1030個の行列を使用

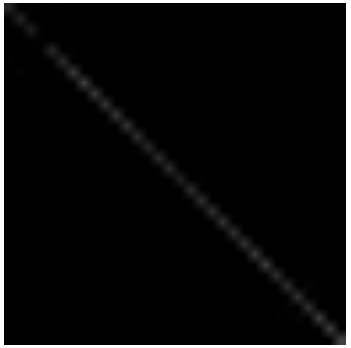
予測対象実装	前処理	6種類
実行設定	SpMV実装	Row Decomposition Method
	最大実行時間	3500秒
	最大リスタート回数	1000回
	リスタート周期	30(最大反復数：30000)
	実行スレッド数	32

前処理		
None	Gauss-Seidel	Jacobi
ILU(0)	ILU(0)_ Diagonal	ILUT ($p = 5$, $\tau = 1.0 \times 10^{-8}$)

**実行環境:Fujitsu PRIMEHPC FX100(名大情報基盤センター)

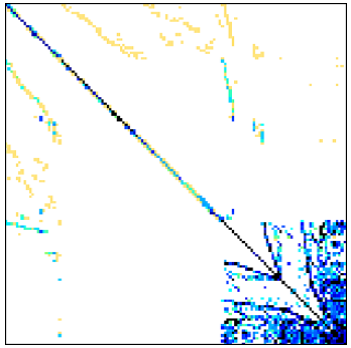
既存手法による特徴画像生成

- 特徴画像：ディープラーニングに使用するための行列画像
- 非ゼロ要素の代わりに行列サイズを画素値として挿入し，グレースケールの行列画像を生成
  ←消失するサイズ情報が保持される
- 問題点
  1. 元行列が大きい場合，構造情報がリサイズによりかなり消失する
  2. 最適実装予測の対象がSpMVであり非ゼロ要素の値を必要としていないため情報として残らない



←既存手法特徴画像

提案手法特徴画像→



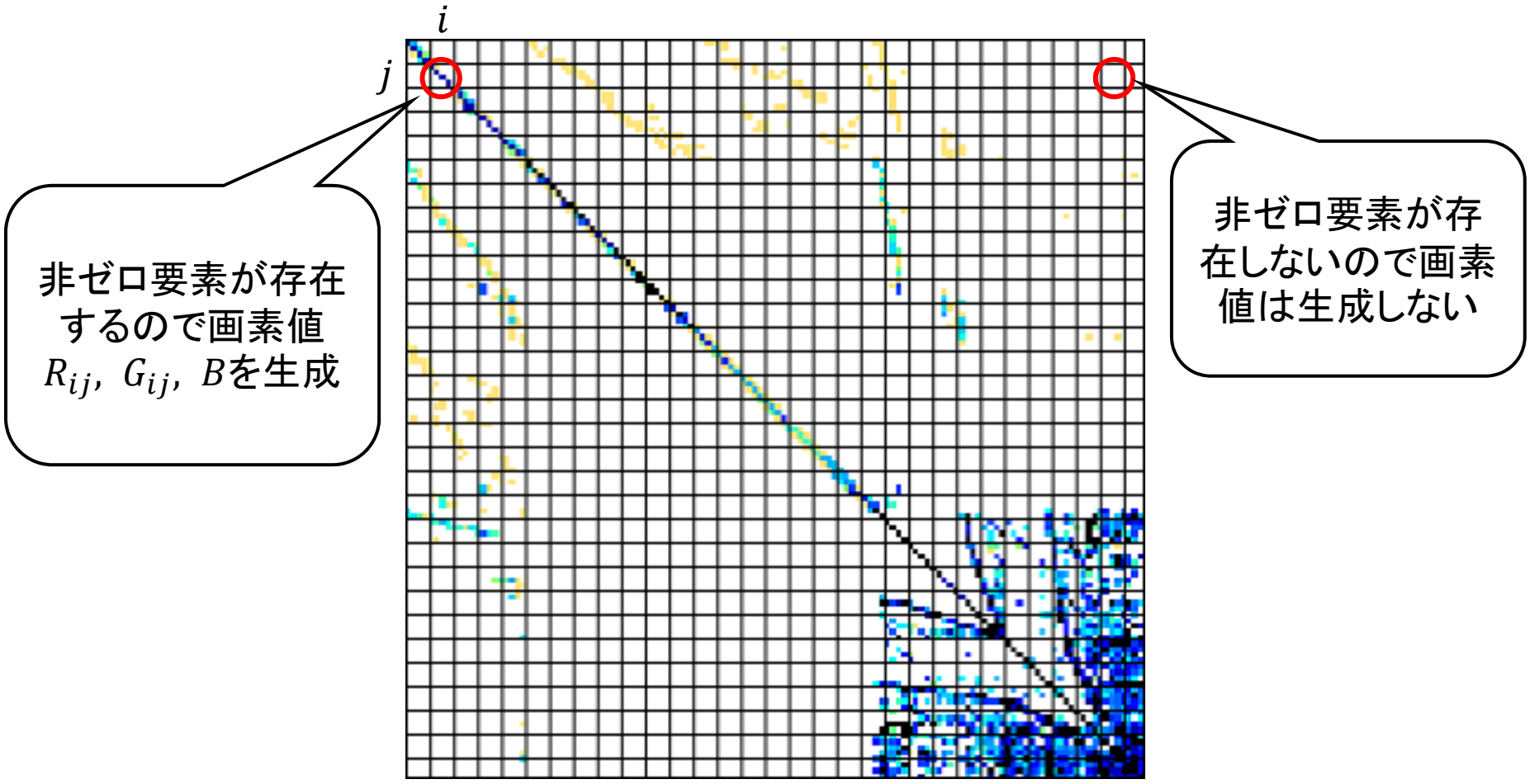
提案手法による特徴画像生成

- 3色カラー画像を用いることで3種類の情報を残す

1. 元行列を64×64等分に分割し非ゼロ要素が存在する区画を(i, j)とし画素値R_{ij}, G_{ij}, Bを生成
2. 64×64カラー画像の座標(i, j)へ画素値を挿入
   - ✓非ゼロ要素が存在しない区画ではいずれの画素値も生成せず，画素値の挿入も行わない
   - ←構造情報が維持される

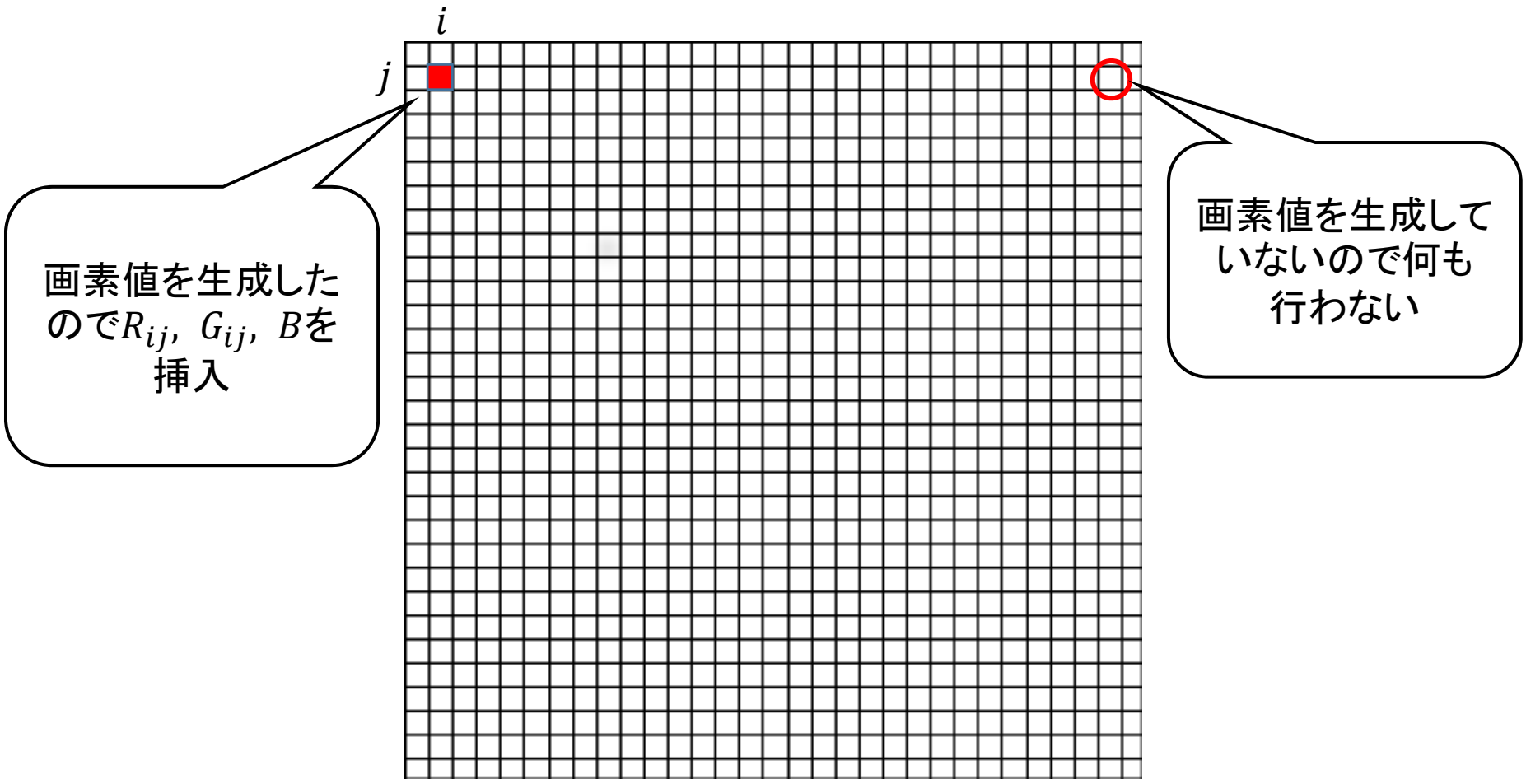
各色が示す情報(全て0～255の整数)
R_{ij}(赤):区画(i, j)における非ゼロ要素の値
G_{ij}(緑):区画(i, j)における疎度
B(青):対象行列のサイズ(全画素で固定値)

提案手法による特徴画像生成



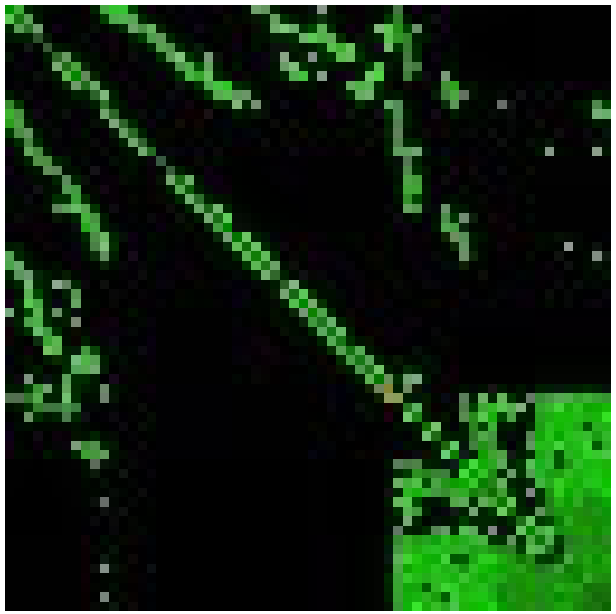
変換前元行列

提案手法による特徴画像生成

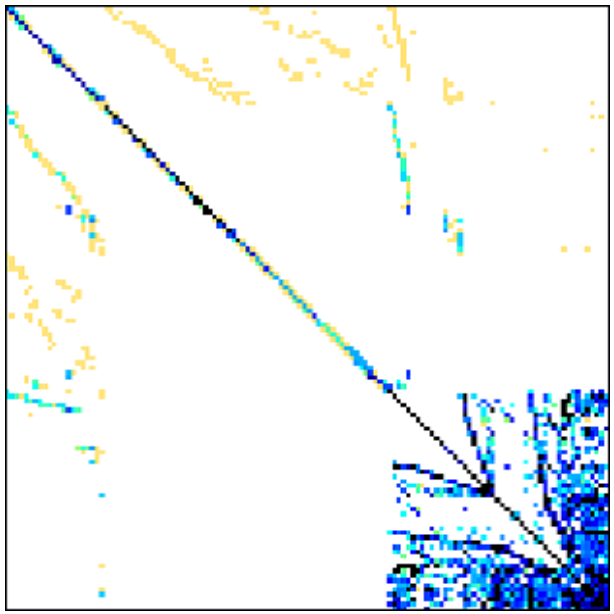


変換中カラー画像

提案手法による特徴画像生成



提案手法特徴画像



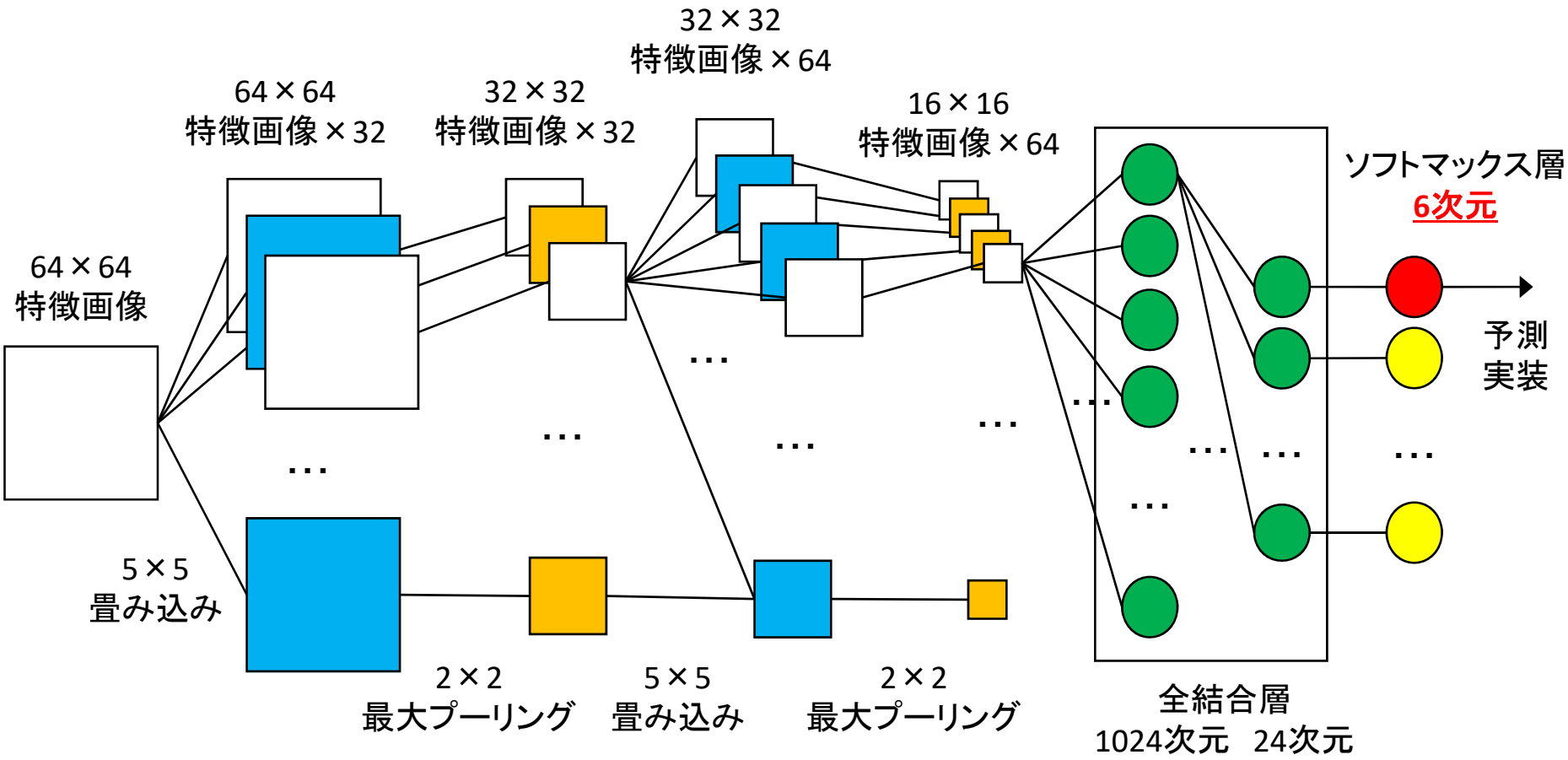
元行列画像

学習設定

使用フレームワーク：TensorFlow
使用マシン：Core i7-4790 3.60GHz 32.0GB

項目	値
総データ(行列)数	1030
訓練データ:テストデータ	4:1
ミニバッチサイズ	80
ステップ数	1250
活性化関数	ReLU
最適化	Adam法
学習率	1.0×10^{-4}
ドロップアウト確率	0.5
学習所要時間	3時間

畳み込みニューラルネット構造
(CNN)



評価結果

- 最適な前処理の予測精度は79.5%

前処理予測精度

前処理	None	Jacobi	Gauss-Seidel	ILU(0)_Diagonal	ILU(0)	ILUT
精度(%)	85.8	80.7	30.0	82.3	52.7	81.1
最適実装数	316	218	20	79	74	323
最適予測数	271	176	6	65	39	262

- 前処理毎に精度を見ると最大で85.8%
- 前処理予測においてはある程度の精度が期待できる
  ←データ数が多い前処理はCNNで特徴をつかめている
  可能性はある

評価結果

前処理予測の分布

		予測実装											
		None		Jacobi		Gauss		ILU(0)_D		ILU(0)		ILUT	
最適実装	None[316]	[271]		14	4	0	0	3	1	8	5	20	13
	Jacobi[218]	15	1	[176]		1	0	2	0	8	0	16	1
	Gauss[20]	2	0	3	3	[6]		3	0	3	1	4	1
	ILU(0)_D[80]	4	2	1	1	1	0	[65]		3	0	5	0
	ILU(0)[74]	12	8	7	6	0	0	2	0	[39]		14	6
	ILUT[323]	28	13	18	8	2	0	4	2	9	3	[255]	

- データ分布の偏りが予測に影響を与えている可能性

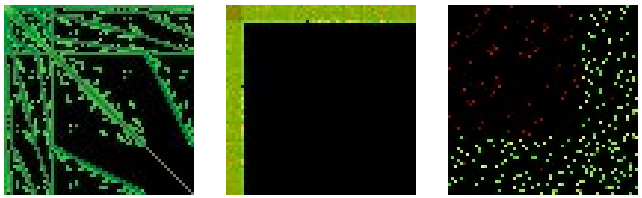
まとめ

- 実験の結果、前処理選択において79.5%の精度で最適なものを特定できた
- 前処理毎の観点では最大で85.8%の精度が得られた
  ←入力データの特性から、この結果が必ずしもモデルの性能を適切に表現できているとは限らない

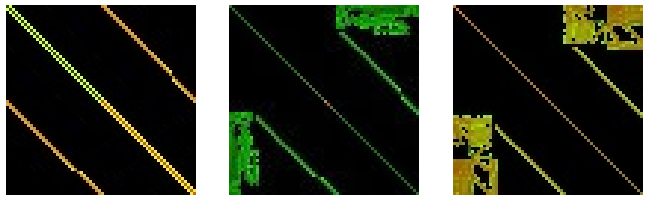
今後の予定

- 精度以外の観点からも評価を行う
- 予測の正否と特徴画像を比較して、特徴画像の生成法の改善点を探る

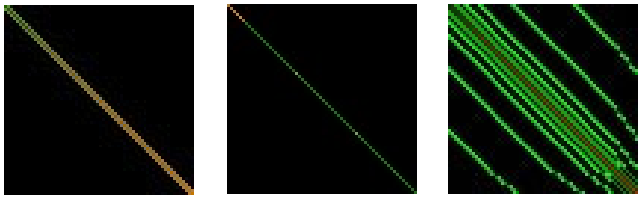
前処理正答行列例



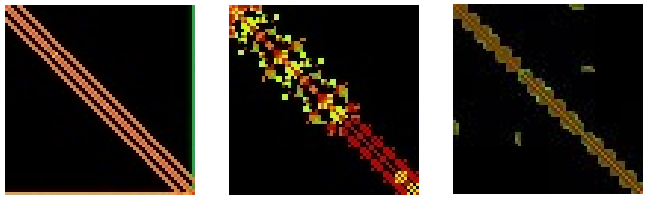
None



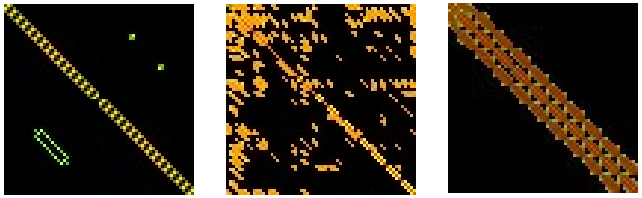
ILU(0)_Diagonal



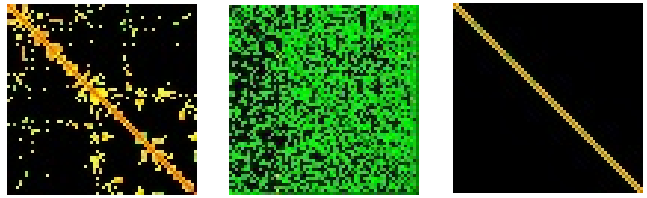
Jacobi



ILU(0)



Gauss-Seidel



ILUT

第167回ハイパフォーマンスコンピューティング研究会
2018/12/17 沖縄産業支援センター

時空間ブロッキングを用いた アジョイント法の性能モデル構築の試み

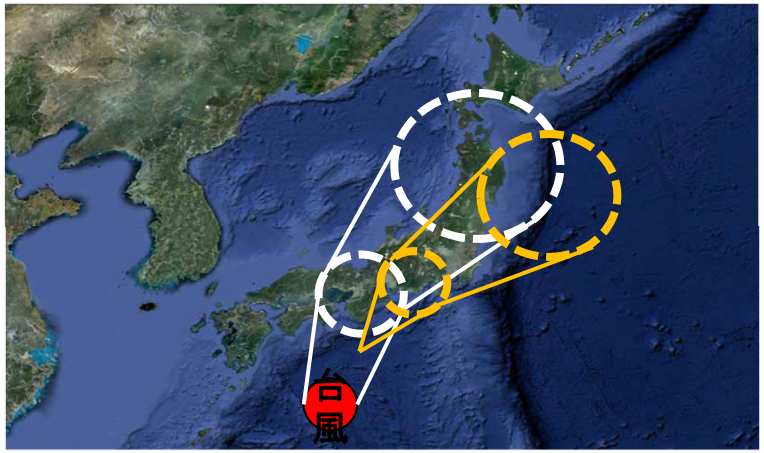
名古屋大学大学院 情報学研究科
・藤川隼人

名古屋大学情報基盤センター
・片桐孝洋
・永井亨
・荻野正雄

研究背景

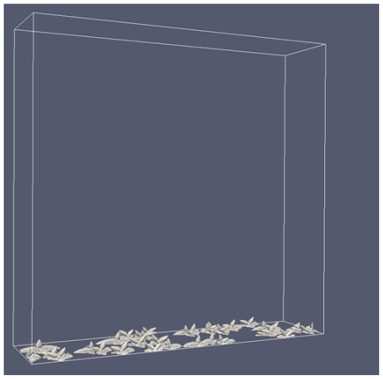
- 数値シミュレーションと観測データを融合する手法としてデータ同化があり，気象学，海洋学などから構造材料分野まで利用されている．
- フェーズフィールドモデルなどの大規模なデータを扱う場合のデータ同化の手法として，アジョイント法がある．

気象予報



物性予測

フェーズフィールド法による予測

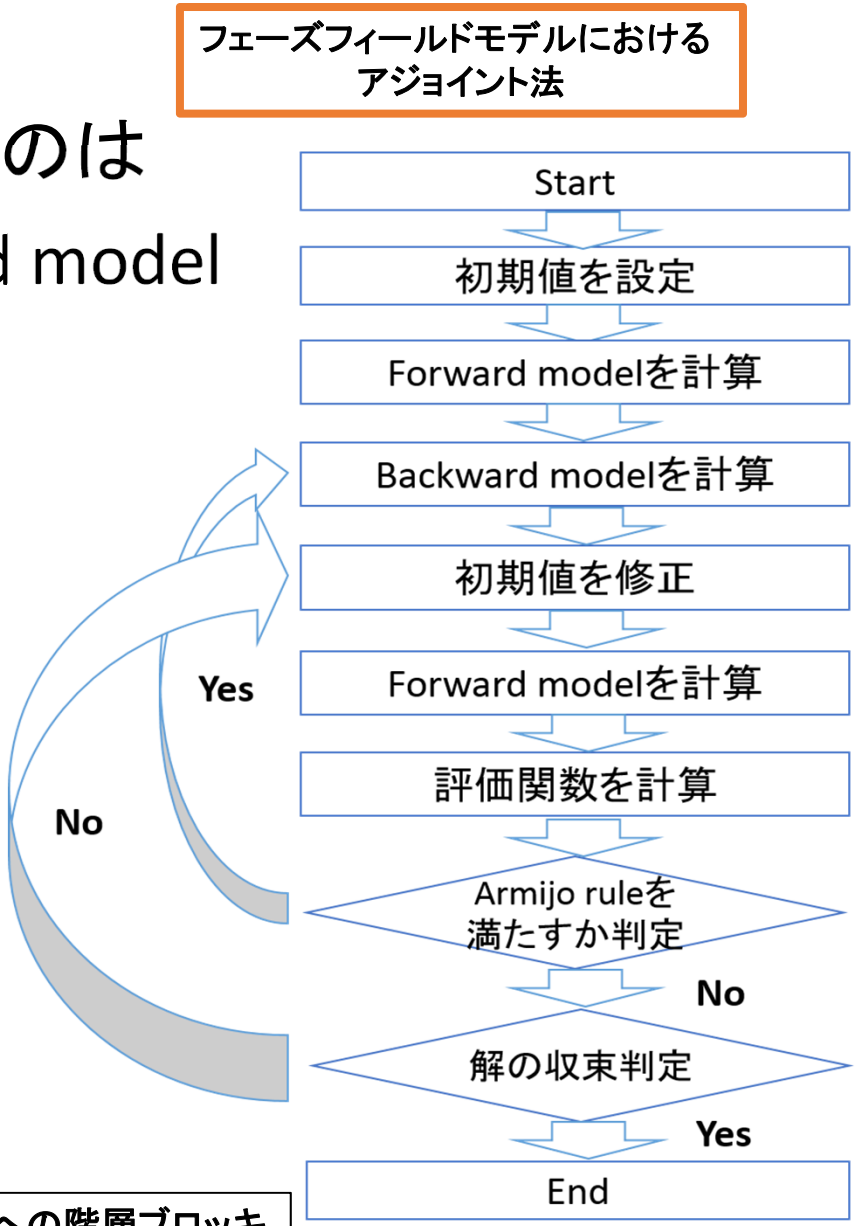


出典 : http://web.tuat.ac.jp/~yamanaka/research.html

アジョイント法

- 計算時間の大半を占めるのは Forward modelとBackward model

- 時空間ブロッキングを用いたForward modelとBackward modelの速度の向上を試みる手法が提案されている

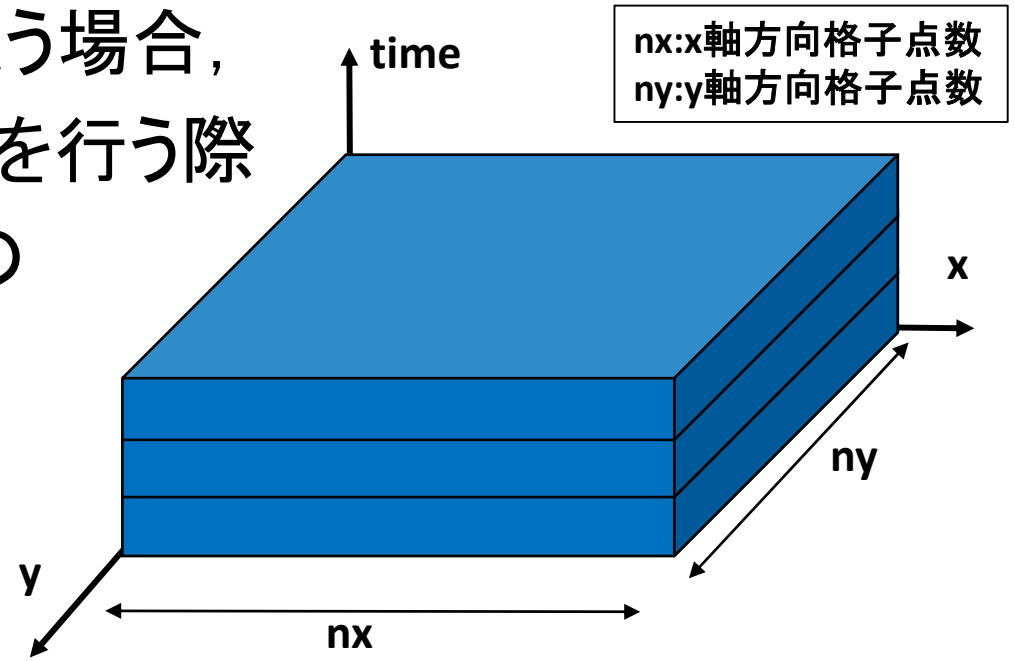


出典:池田朋哉 ほか, アジョイント法におけるForward model への階層ブロッキング適用による高性能化, 情報処理学会研究報告, Vol. 2016-HPC-157

Forward modelの計算(naïveな実装)

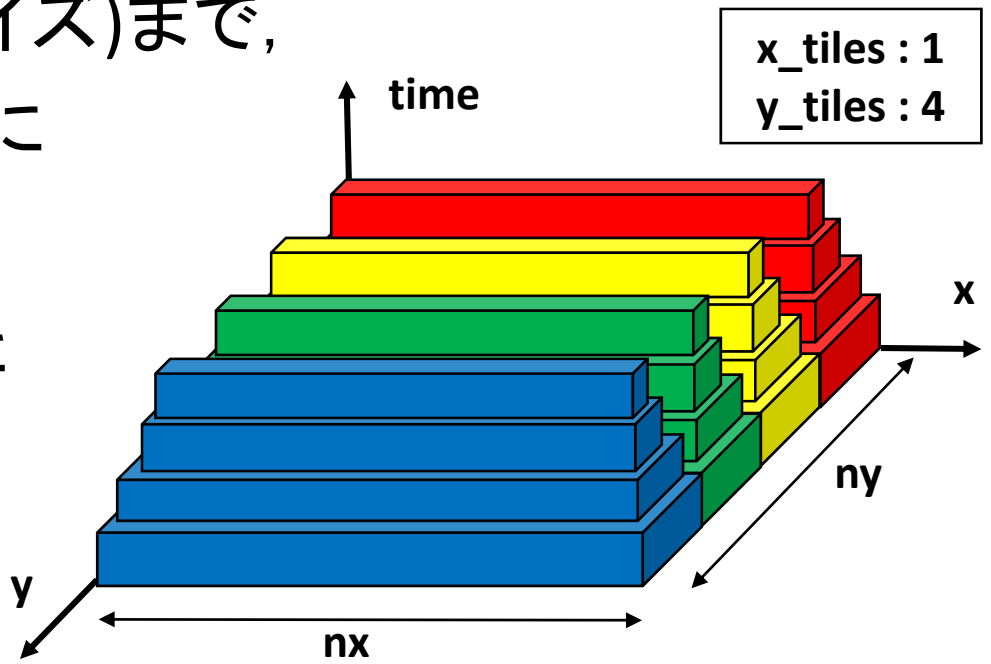
- 全ての格子点の値に対して，
  次の時間ステップまで計算する.

- 大規模なモデルを扱う場合，
  次のステップの計算を行う際
  キャッシュのデータの
  再利用が出来ない.



Forward modelの計算 (時空間ブロッキング)

- まず，指定された時間ステップ(時空間ブロッキングサイズ)まで，格子点値をピラミッド型に計算する.
- 残りの計算できなかった領域の格子点値はその後計算.



キャッシュのデータを再利用できる

出典:池田朋哉 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告, Vol. 2016-HPC-157

計算コスト(先行研究より)

- 先行研究から，以下のモデルを想定

E ＝問題サイズ
W＝ 空間ブロッキングサイズ

C_{first} ＝ 最初のキャッシュアクセスの時間
$C_{intermediate}$ ＝ プリフェッチエンジンの立ち上がり時の時間
C_{stream} ＝ ストリーム計算の時間

$$C_{total}[s] = C_{first} + k * C_{intermediate} + (\lceil E/W \rceil - k - 1) * C_{stream}$$

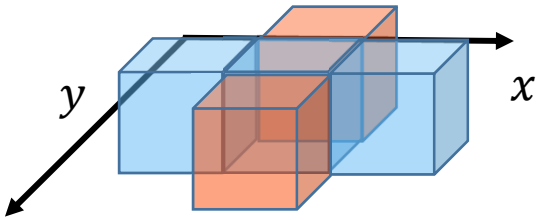
kは小さい

$$C_{total}[s] = C_{first} + (\lceil E/W \rceil - 1) * C_{stream}$$

出典:OPTIMIZATION AND PERFORMANCE MODELING OF STENCIL COMPUTATIONS ON MODERN MICROPROCESSORS, Proc. of IEEE SC2008 (2008)
KAUSHIK DATTAら、ほか６人

計算時間の上限と下限

- 5点ステンシル計算の場合は以下の赤い部分をキャッシュミスするかどうかで考える



計算方向はx軸方向

- 計算時間が下限の時は赤色のキャッシュミスが無い時で，上限は赤色を二つともキャッシュミスする時.
- 下限 $2C_{total}$[s](読み込み1回，書き込み1回)
- 上限 $4C_{total}$[s](読み込み3回，書き込み1回)

ベンチマーク

- ベンチマークとしてstream triadベンチマークにOpenMP構文を加えたもの使用.

```
!$omp parallel private(i)
    do j=1,STEP_NUMBER,1
      !$omp do
          do i=1,X_ARRAY_LENGTH*Y_ARRAY_LENGTH,1
              A(i) = SCALAR*B(i) + C(i)
           enddo
       !$omp end do
    enddo
!$omp end parallel
```

- 我々が開発したプログラムの仕様上，E=Wであるため，現段階では以下のモデルで評価を行った

$$C_{total}[s] = C_{first}$$

出典:https://github.com/jeffhammond/STREAM/blob/master/stream.f

実験環境

Fujitsu PRIMEHPC FX100

CPU	Fujitsu SPARK64 Xlfx (2.2GHz)
1ノード当たりのコア数	32(1ソケットあたり16コアの2ソケットNUMA構成)
ノード当たりの理論演算性能	1.126GFLOPS
メモリ容量	32GiB
L1キャッシュ容量	64KB(コア毎)
L2キャッシュ容量	24MB(共有)
コンパイラ	frtpx : Fujitsu Fortran Driver Version 2.0.0 P-id: T01776-01
コンパイラオプション	-Kfast –Kopenmp
メモリアクセス性能	240GB/s (読出し、書込みごと) 合計：480GB/s

Fujitsu PRIMERGY CX400

CPU	Intel Haswell(2.6GHz) Intel Xeon E5-2600 v3 processor family
1ノード当たりのコア数	28(1ソケットあたり14コアの2ソケットNUMA構成)
ノード当たりの理論演算性能	1,164GFLOPS
メモリ容量	128GiB
L1キャッシュ容量	64KB(コア毎)
L2キャッシュ容量	256KB(コア毎)
L3キャッシュ容量	35MB(共有)
コンパイラ	frt: Fujitsu Fortran Driver Version 1.2.0 P-id: T01778-01
コンパイラオプション	-Kfast –Kopenmp
メモリアクセス性能	136GB/s

両マシンとも，名古屋大学情報基盤センター設置

テストモデル: フェーズフィールドモデル

単純界面移動フェーズフィールドモデル

R. Kobayashi (1993)

$$\tau\frac{\partial\phi}{\partial t}=\epsilon^2\Delta\phi+\phi(1-\phi)\left(\phi-\frac{1}{2}+m\right) \qquad |m|<\frac{1}{2}$$

既知のパラメータ: τ, ϵ

未知のパラメータ: m

パラメータ

τ	時間の単位
ϵ	空間の単位
m	発展速度

(全空間で一定値)

$$\begin{cases} m=const. \\ \phi(-\infty,t)=1, \phi(\infty,t)=0 \end{cases}$$

$$\phi(x,t)=\frac{1}{2}[1-tanh(\frac{x}{2\sqrt{2}\epsilon}-\frac{mt}{2\tau})]$$

mの時間発展方程式を導入

$$\frac{\partial m}{\partial t}=0$$

拘束条件

$$|m|<\frac{1}{2} \qquad 0<\phi(x,0)<1$$

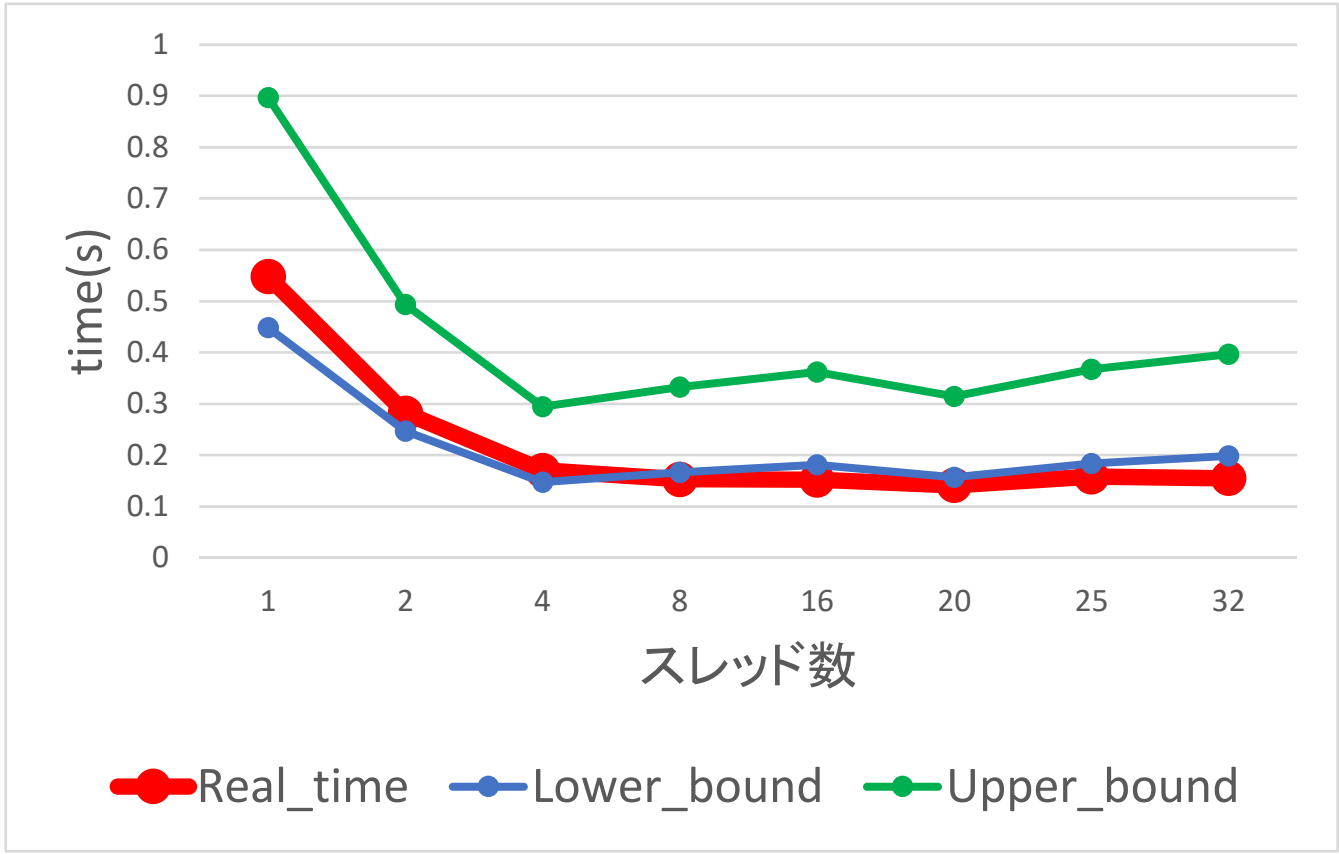
問題

初期状態$\phi(x,0)$とパラメータmはどのようなものか?

出典：伊藤 伸一, 長尾 大道 「大自由度フェーズフィールドモデルのための2nd-order-adjoint法に基づくデータ同化」

FX100での評価

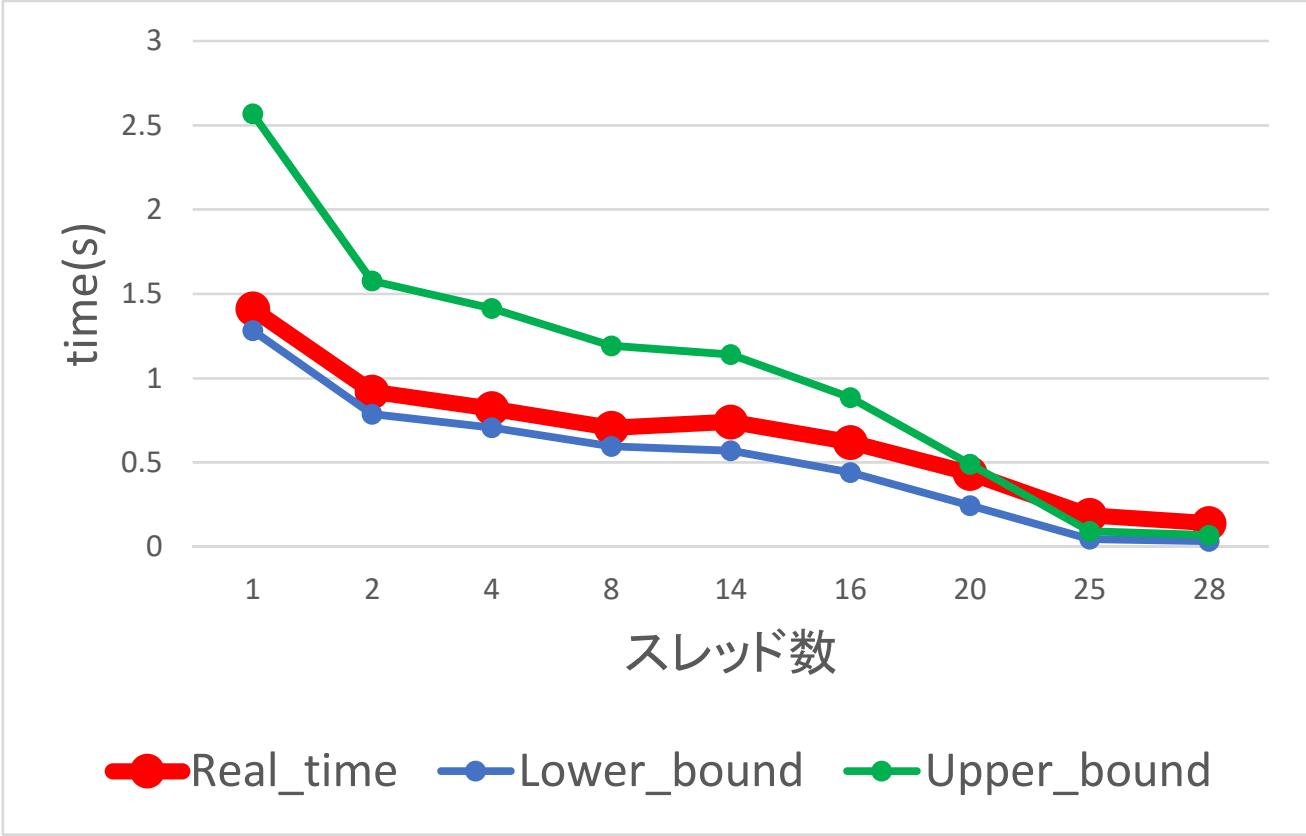
- スレッド数が8を超えると実時間が下限を下回る.
  計算時間が最速となるスレッド数が，実時間では16スレッドであるが，性能モデルでは4スレッドであり異なる.



FX100でのモデル評価

CX400での評価

- スレッド数が25，28の時に実時間が上限を上回る．計算時間が最速となるスレッドは，実時間でも性能モデルでも28スレッドであり，最速時間の予測の観点では性能モデルはよい精度を示している．



CX400でのモデル評価

時間ブロッキングモデル化の手法

- 格子点によって計算コストを変化させるため，1つの格子点にかかる計算時間が必要となる

nx	問題領域のx軸方向の格子点数
ny	問題領域のy軸方向の格子点数
STEP_NUMBER	時間ステップ数
N_THREAD	並列数

- 1つの格子点の計算時間$T_{stencil}$は

$$T_{stencil} = C_{total}/(nx * ny * STEP\_NUMBER/N\_THREAD)$$

と表される

時間ブロッキングモデル化の手法

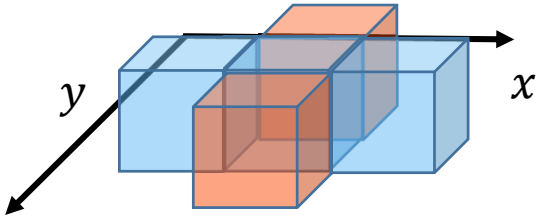
- ブロッキングサイズを*iblt*, *n*を0以上の整数として，時間ブロッキングモデルの上限と下限を設定する

上限

- *n*iblt+1*及び袖領域の格子点は，

キャッシュが外れると考えるので

$$4T_{stencil}$$



計算方向はx軸方向

- それ以外の格子点はすべてキャッシュが当たると考えるので

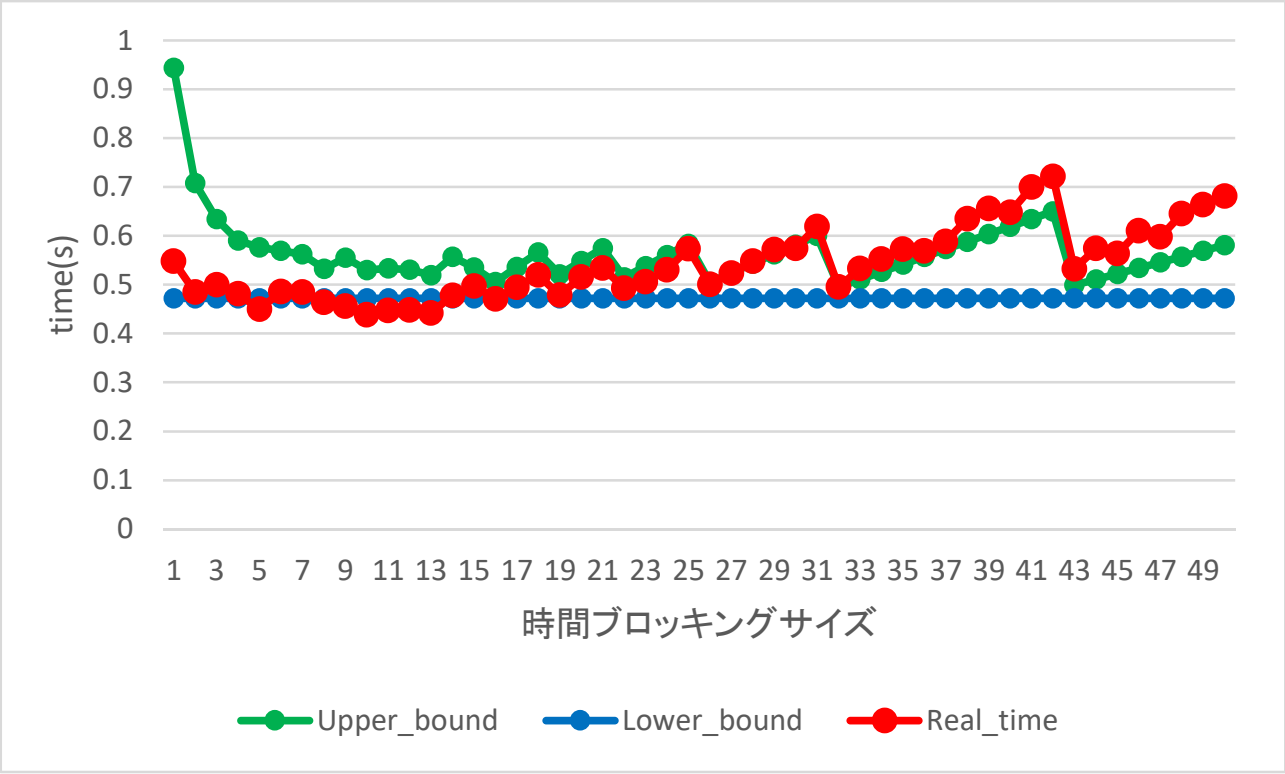
$$2T_{stencil}$$

下限

- 全ての格子点値でキャッシュが当たるものと考えるので

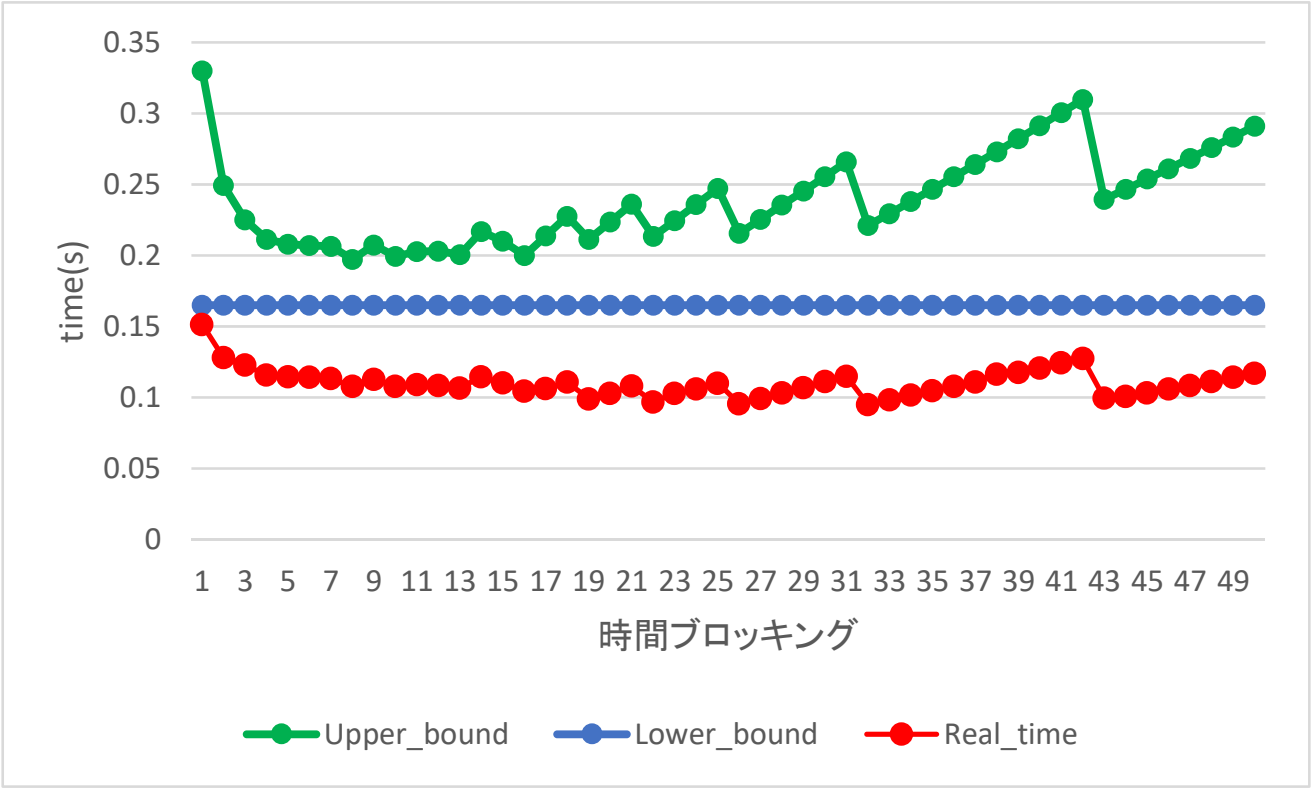
$$2T_{stencil}$$

FX100での結果(スレッド数1)



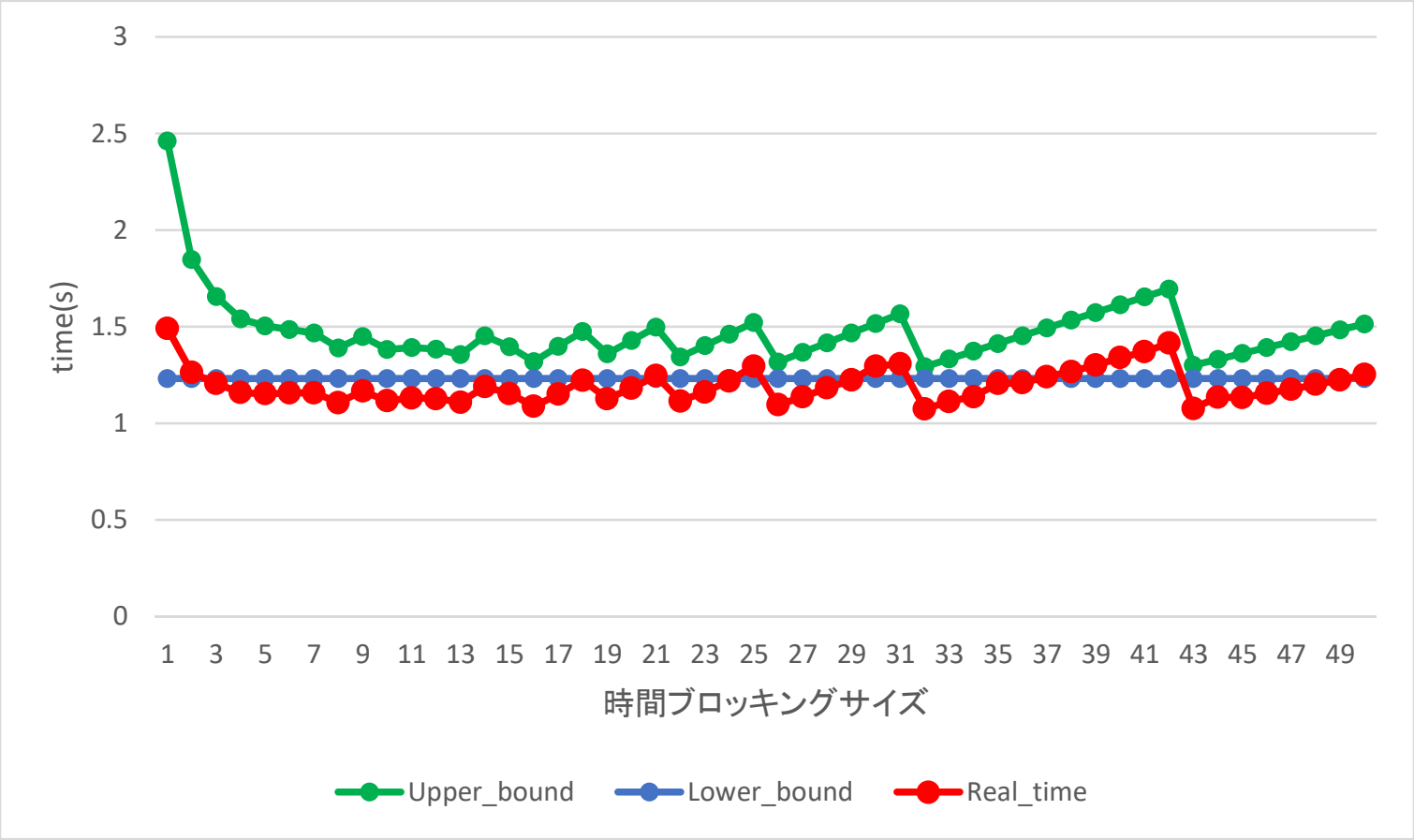
FX100(スレッド数1)でのモデル評価

FX100での結果(スレッド数16)



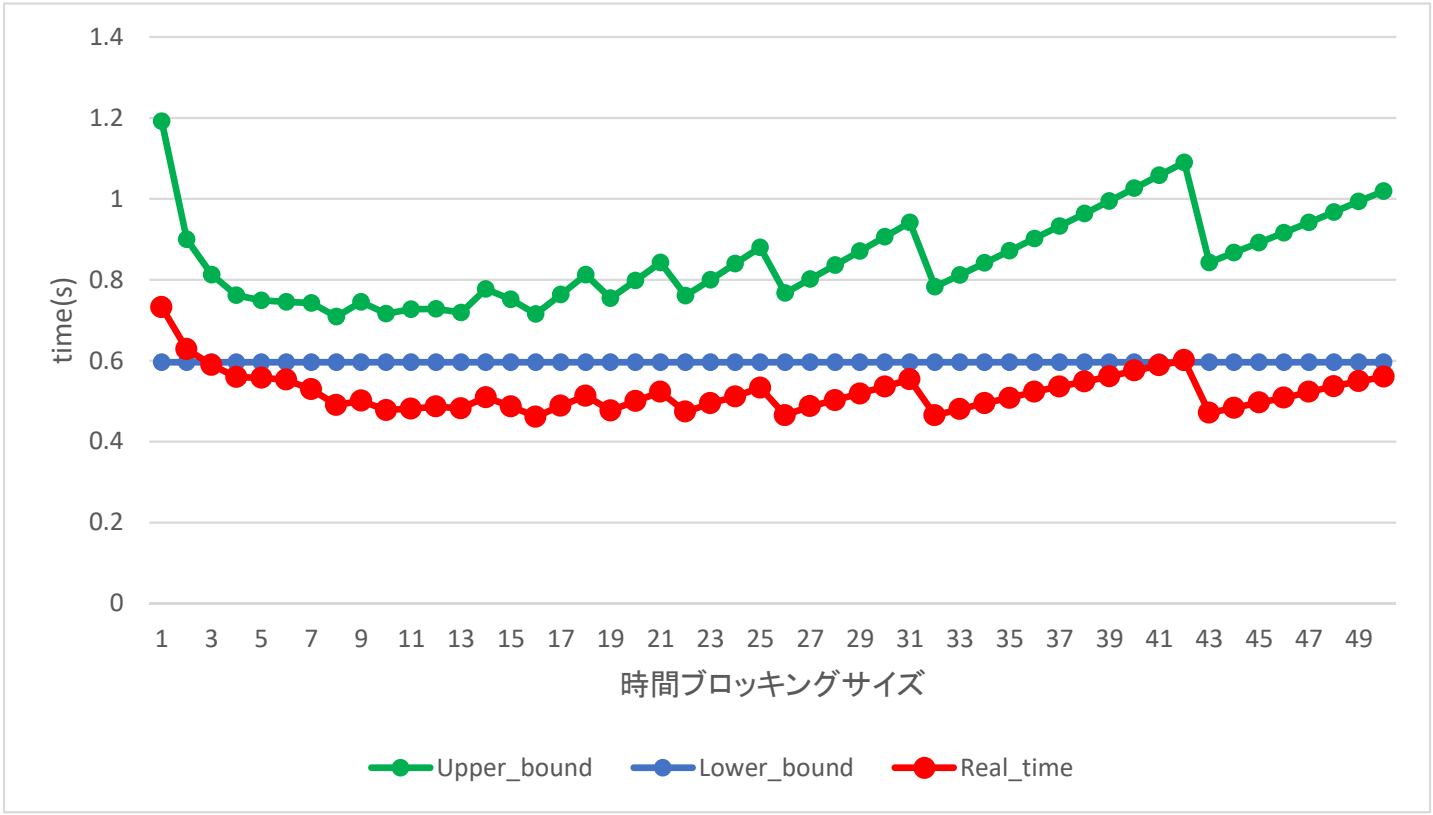
FX100(スレッド数16)でのモデル評価

CX400での結果(スレッド数1)



CX400(スレッド数1)でのモデル評価

CX400での結果(スレッド数14)



CX400(スレッド数14)でのモデル評価

片桐基盤Bの主な成果（2018年度）

- AT言語開発・AT性能モデル
  - 田中（工学院大）G：　D-SplineによるAT方式
  - 片桐（名大）G：
    1. ディープラーニングを用いたAT方式
    2. 時空間ブロッキング向け性能モデル
    3. AT言語（ppOpen-AT）言語開発
- アプリケーション適用
  - 長尾（東大）G：　データ同化アルゴリズム開発・AT適用
  - 佐藤（核融合研）G：
    プラズマシミュレーションアルゴリズム開発・AT適用
  - 片桐（名大）G：　医療画像処理高性能実装・AT適用検討
- 高性能実装
  - 片桐（名大）G：　ループ変換技法、動的スレッド数変更
  - 大島（九州）：　メニーコア・GPU実装