情報処理学会論文誌　コンピューティングシステム　Vol.6　No.1　56–65 (Jan. 2013)

冗長な計算を伴わない 3次元FDTD法の時空間タイリング

南 武志[1]　岩下 武史[1,a)]　中島 浩[1]

受付日 2012年7月4日, 採録日 2012年10月10日

概要：本論文では高周波電磁場解析の一手法である 3 次元 FDTD 法におけるキャッシュメモリを考慮した性能改善手法の提案と性能評価を行う．3 次元 FDTD 法の計算カーネルは時間発展に関するループにより与えられ，各タイムステップにおいて電場と磁場の値が交互に更新される．3 次元 FDTD 法の計算カーネルは演算あたりのロード/ストア量が大きく，一般にメモリ帯域の影響を受けやすい計算である．キャッシュメモリのヒット率を向上させメインメモリへのアクセスによる性能の低下を軽減する性能改善手法として，解析領域をタイルと呼ぶ小領域に分割し各タイル内で複数のタイムステップに関する処理を連続して行うタイリングと呼ばれる手法が存在する．しかし，単純な固定タイルによる実装では，タイルの辺縁領域に対する冗長な計算がオーバヘッドとなっていた．そこで，本論文ではタイリング手法において，タイルの位置を時間ステップごとに変化させ計算量の増加を防ぐ手法を提案する．提案手法を評価した結果，AMD 製クアッドコア Opteron プロセッサよる数値実験において 4 スレッドによる並列処理を行った場合，一般的な 3 次元 FDTD 法の実装と比較して計算時間を約 50%短縮させることに成功した．

キーワード：FDTD 法，キャッシュチューニング，タイリング，マルチコアプロセッサ，スレッド並列処理

Temporal and Spatial Tiling Method without Redundant Calculations for Three-dimensional FDTD Method

Takeshi Minami[1]　Takeshi Iwashita[1,a)]　Hiroshi Nakashima[1]

Received: July 4, 2012, Accepted: October 10, 2012

Abstract: This paper deals with performance improvement of three dimensional FDTD kernel for high frequency electromagnetic field analyses. The FDTD method is one of explicit time stepping methods. The electric and magnetic fields are updated alternately in each time step. Since the calculation of the FDTD method has a large byte/flop ratio, its performance is limited by memory throughput. For a remedy of it, there is a technique called tiling, in which the analyzed domain is divided into multiple small domains, or tile. By updating electrical and magnetic fields in each tile in multiple time steps, we can utilize cache data efficiently. However, when we implement tiling based on simple fixed size tiles, redundant calculations are required for overlapped tile peripheries. In this paper, we propose a new tiling technique for three dimensional FDTD method without redundant calculations. This method prevents an increase in the amount of calculations by changing the position of the tile at each time step. Numerical tests on a quad-core AMD Opteron processor show that the proposed three dimensional FDTD method attains up to 50 percent reduction in the calculation time compared with an ordinary implementation of the three dimensional FDTD method.

Keywords: FDTD method, cache tuning, tiling, multicore processor, multithreading

1. はじめに

計算機による電磁場シミュレーションは重要な科学技術計算の 1 つであるが，解析対象の大規模化・複雑化にとも

[1] 京都大学
Kyoto University, Kyoto 606–8501, Japan
[a)] iwashita@media.kyoto-u.ac.jp

© 2013 Information Processing Society of Japan

ない計算時間が増大しており，その高速化が要望されている．計算機シミュレーションの高速化手法の 1 つとして，シミュレーションを行う計算機システムに合わせたチューニングを行うことによる性能改善があげられる．

本論文では，高周波電磁場解析において主要な解析手法の 1 つである FDTD（Finite Difference Time Domain）法 [1] の性能改善手法について述べる．FDTD 法は解析空間内の電場および磁場を与えられた初期値から時間ステップごとに陽的に解いていく手法であり，時間ステップを進めるごとに電磁場の値を更新する．しかしながら，通常のシミュレーションにおいて解析に必要なデータサイズは使用プロセッサのキャッシュ容量よりもはるかに大きく，1 演算あたりのメモリデータ参照・更新回数も大きいため，大量のメモリアクセスが必要となる．そのため，メモリの帯域が計算速度のボトルネックとなりプロセッサの演算性能と比較して十分な性能が得られない場合が多い．さらに，マルチコアプロセッサによる並列処理を行った場合，複数のコアによるメモリ帯域の競合が生じ，コアあたりの帯域はさらに制限され，本問題はより顕著となる．

一般に「反復型ステンシル計算」と呼ばれる時間に関するループの中に空間に関するステンシル演算のループを持つ計算手順において，複数のタイムステップを小領域（タイル）内で個別に行うことによりキャッシュのヒット率を向上させるタイリングと呼ばれる手法 [2] が存在する．これによりキャッシュメモリの利用効率を上げ，メモリ帯域による性能の低下を軽減し，プロセッサの演算能力を効果的に利用することが可能である．FDTD 法は反復型ステンシル計算の一種であり，著者らは文献 [3] において，同手法を 3 次元 FDTD 法に適用し，マルチコアプロセッサ上で性能評価を行った．その結果，コアあたりの実効的なメモリバンド幅が小さくなるマルチスレッド実行時においては，一般的な実装（Naive な実装）手法と比べて性能向上が得られた．一方，文献 [3] に述べた実装手法では，各タイムステップで固定のタイルを用いたため，複数のタイルの辺縁領域が重複することによる冗長な計算が必要となっていた．そこで，本論文では，3 次元 FDTD 法のタイリング手法において，タイルの位置を時間的に変化させるとともに，タイル内の電場・磁場更新の順序を適切に設定することで，冗長な計算を必要としない新たな実装手法を提案する．同手法を一般的な 3 次元 FDTD 法の実装（Naive な実装）と比較し，性能の向上が得られることを示す．

2. 3 次元 FDTD 法による電磁場シミュレーション

3 次元 FDTD（Finite Difference Time Domain）法は，直方体メッシュに区切られた空間内に離散的に配置された未知変数である電場 $\boldsymbol{E}$，磁場 $\boldsymbol{H}$ を，与えられた初期値から時間ステップごとに陽的に解いていく電磁場数値解析手法の 1 つである．一般に同手法では，離散電磁場変数の配置に Yee のメッシュを用い，電磁場計算を Leap-flog アルゴリズムにより行う [4]．

電磁場の振舞いを表す支配方程式は，Maxwell の方程式より次のように表される．

$$\nabla \times \boldsymbol{E} = -\mu \frac{\partial \boldsymbol{H}}{\partial t}, \tag{1}$$

$$\nabla \times \boldsymbol{H} = \epsilon \frac{\partial \boldsymbol{E}}{\partial t} + \sigma \boldsymbol{E}. \tag{2}$$

ここで，諸変数は $\boldsymbol{E}$：電場，$\boldsymbol{H}$：磁場，誘電率 ϵ，透磁率 μ，導電率 σ である．

FDTD 法の電磁場計算においては，以下の Leap-flog アルゴリズムが用いられる．このアルゴリズムでは，離散的な時間発展計算を電場と磁場で半ステップずらし，また時間微分項の差分近似に中心差分を用いることで，あるタイムステップの電場から半ステップ後の磁場を求め，またその磁場からさらに半ステップ後の電場を求めるという繰返し計算により，電磁場計算を行う．

式 (1)，式 (2) に対して，Leap-flog アルゴリズムを用いた時間方向の離散化は以下のように行われる．時間ステップ Δt に対して，時間方向に関して時刻 $n\Delta t$ の電場 $\boldsymbol{E}^n$，時刻 $(n + (1/2))\Delta t$ の磁場 $\boldsymbol{H}^{n+1/2}$ は

$$\frac{\boldsymbol{E}^n - \boldsymbol{E}^{n-1}}{\Delta t} = \frac{1}{\epsilon}\left(\nabla \times \boldsymbol{H}^{n-1/2}\right) - \frac{\sigma}{\epsilon}\boldsymbol{E}^{n-1/2}, \tag{3}$$

$$\frac{\boldsymbol{H}^{n+1/2} - \boldsymbol{H}^{n-1/2}}{\Delta t} = -\frac{1}{\mu}\left(\nabla \times \boldsymbol{E}^n\right), \tag{4}$$

となる．ここで $\boldsymbol{E}^{n-1/2}$ を $\left(\boldsymbol{E}^n + \boldsymbol{E}^{n-1}\right)/2$ で近似すると，離散化されたマクスウェルの方程式

$$\boldsymbol{E}^n = \frac{1 - \{\sigma\Delta t/(2\epsilon)\}}{1 + \{\sigma\Delta t/(2\epsilon)\}}\boldsymbol{E}^{n-1} + \frac{\Delta t/\epsilon}{1 + \{\sigma\Delta t/(2\epsilon)\}}\left(\nabla \times \boldsymbol{H}^{n-1/2}\right), \tag{5}$$

$$\boldsymbol{H}^{n+1/2} = \boldsymbol{H}^{n-1/2} - \frac{\Delta t}{\mu}\left(\nabla \times \boldsymbol{E}^n\right), \tag{6}$$

が得られる．

解析空間内に x 軸，y 軸，z 軸を持つ直交座標系を考え，格子座標 (i, j, k) を導入することにより，各格子点の電場を $\boldsymbol{E}(i, j, k)$，磁場を $\boldsymbol{H}(i, j, k)$ と表すものとする．**図 1** に示すように，解析空間を離散化しセルの辺上に電場 E_x，E_y，E_z，磁場 H_x，H_y，H_z を交互に配置することより，式 (5)，式 (6) は，本論文では Naive な実装法と呼ぶ**図 2** に示すコードにより実現される．ここで，本論文では，解析空間は複数の媒質により構成されるものとし，媒質ごとに定まる ϵ，μ，σ からなる式 (5)，式 (6) 中のスカラ係数については，各媒質に対するこれらの値を配列に保持し，各格子点が属する媒質の種類を保持する整数配列 $id(i, j, k)$ を介して参照するものとする．

© 2013 Information Processing Society of Japan

```
for(t=0;t<nt;t++){
  for(i=1;i<=nx;i++){
    for(j=1;j<=ny;j++){
      for(k=1;k<=nz;k++){
        m=id[i][j][k];
        Ex[i][j][k] = Ce[m] * Ex[i][j][k]
                    + Cery[m] * (Hz[i][j][k] - Hz[i][j-1][k])
                    + Cerz[m] * (Hy[i][j][k] - Hy[i][j][k-1]);
        Ey[i][j][k] = Ce[m] * Ey[i][j][k]
                    + Cerz[m] * (Hx[i][j][k] - Hx[i][j][k-1])
                    + Cerx[m] * (Hz[i][j][k] - Hz[i-1][j][k]);
        Ez[i][j][k] = Ce[m] * Ez[i][j][k]
                    + Cerx[m] * (Hy[i][j][k] - Hy[i-1][j][k])
                    + Cery[m] * (Hx[i][j][k] - Hx[i][j-1][k]);
  }}}
  for(i=1;i<=nx;i++){
    for(j=1;j<=ny;j++){
      for(k=1;k<=nz;k++){
        m=id[i][j][k];
        Hx[i][j][k] = Hx[i][j][k]
                    + Chry[m] * (Ez[i][j+1][k] - Ez[i][j][k])
                    + Chrz[m] * (Ey[i][j][k+1] - Ey[i][j][k]);
        Hy[i][j][k] = Hy[i][j][k]
                    + Chrz[m] * (Ex[i][j][k+1] - Ex[i][j][k])
                    + Chrx[m] * (Ez[i+1][j][k] - Ez[i][j][k]);
        Hz[i][j][k] = Hz[i][j][k]
                    + Chrx[m] * (Ey[i+1][j][k] - Ey[i][j][k])
                    + Chry[m] * (Ex[i][j+1][k] - Ex[i][j][k]);
  }}}
}
```

図 2　3 次元 FDTD 法の Naive な実装法におけるループ構造と電磁場計算
Fig. 2　Loop structure of three-dimensional FDTD kernel based on naive implementation method.

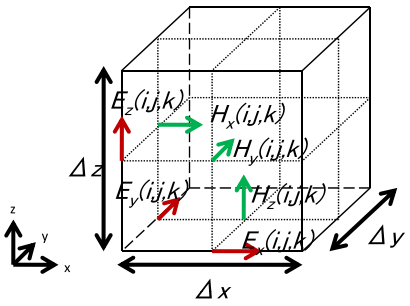


図 1　Yee のメッシュ
Fig. 1　Yee-cell.

図 2 に示されるように，Naive な実装法では，1 格子点・タイムステップあたりの浮動小数点演算数は 39 であるのに対し，データのロード/ストア回数は媒質に関するデータを除いても 30 となる．したがって，解析領域が十分に大きい場合には，メモリの帯域に計算が律速される．

3.　キャッシュメモリの有効利用による 3 次元 FDTD 法の性能改善手法

前章で述べたように，3 次元 FDTD 法の実装においては解析領域が十分に大きい場合メモリの帯域に計算が律速される．そこで，空間に関するループである電場・磁場の複数回の更新を解析領域を分割した小領域（タイル）を単位として行うことで，配列要素の参照間隔を短くして時間的局所性を向上させ，要求メモリ帯域を低減する手法（タイリング）が考えられる．著者らは文献 [3] において，複数タイムステップの計算を固定されたタイル上で行いタイムステップ間でタイルを再利用することでキャッシュメモリのヒット率の向上を図る性能改善手法を提案した．しかし，FDTD 法ではあるタイルの計算を行うためにはタイル外の隣接する格子点上の値も必要となるため，1 つのタイル上で複数タイムステップの更新を行う場合，そのタイルに含まれる格子点のうち，まだ更新していないタイルの計算に必要な格子点のデータを重複して保存しておかなければならない．

そこで，本論文では各タイル上で複数タイムステップの更新を行う際に更新する領域をタイムステップごとに変化（移動）させることにより，各格子点のデータの重複を必要とすることなく計算を行うことのできるタイリング手法を提案する．

3.1　提案手法の概要

タイリングによる 3 次元 FDTD 法の実装では，解析領域をまず小領域（タイル）に分割し，各タイル上で複数ステップの電磁場更新を行う．ここで，タイルがキャッシュメモリ（主に 2，3 次キャッシュ）に収まるサイズの場合，タイル上での電磁場更新操作においてすべての配列要素の参照・ストアがキャッシュメモリにヒットするため，キャッシュヒット率の向上が期待できる．

以下に提案手法について述べる．解析領域全体のサイズ

© 2013 Information Processing Society of Japan

を $n_x \times n_y \times n_z$ とし，これを大きさ $t_x \times t_y \times t_z$ のタイルに分割する．ここで，以下のようにタイルに関する座標 (I, J, K) を導入する．

$$(I, J, K) = (\lceil i/t_x \rceil, \lceil j/t_y \rceil, \lceil k/t_k \rceil). \tag{7}$$

このとき，タイル座標 (I, J, K) に位置するタイル $T_{I,J,K}$ は，

$$\begin{aligned} T_{I,J,K} = \{(i,j,k) | &(I-1)t_x + 1 \le i \le I \cdot t_x, \\ &(J-1)t_y + 1 \le j \le J \cdot t_y, \\ &(K-1)t_z + 1 \le k \le K \cdot t_z\}, \end{aligned} \tag{8}$$

のような格子点の集合として定義される．次に，タイル内で連続して計算を行うタイルステップを s_t と表す．ここでは，すべての格子点の電場・磁場を時刻 t から $t + s_t$ に更新することについて考える．ここで，各格子点 (i, j, k) について，電場・磁場の更新状況を示す 2 つの要素を持つ配列 $\boldsymbol{s}_{i,j,k}$，

$$\boldsymbol{s}_{i,j,k} = (s_e, s_h), \quad s_e, s_h \in \{0, \ldots, s_t\}, \tag{9}$$

を導入する．

ここで，3 次元 FDTD 法の各格子点の電場 $\boldsymbol{E}$ および磁場 $\boldsymbol{H}$ の更新には，それぞれ解析領域上で隣接する格子点の $\boldsymbol{H}$，$\boldsymbol{E}$ が必要である．すなわち，状態 $\boldsymbol{s}_{i,j,k} = (u, u), u \in \{0, \ldots, s_t\}$ を電場更新して $\boldsymbol{s}_{i,j,k} = (u+1, u)$ にするためには更新に関係する格子点の磁場の状態が u である必要がある．ここで，FDTD 法の計算において各格子点上では電場，磁場は交互に更新されることから，これは $\boldsymbol{s}_{i-1,j,k}, \boldsymbol{s}_{i,j-1,k}, \boldsymbol{s}_{i,j,k-1}$ が (u, u) または $(u+1, u)$ である必要性を意味する．同様に $\boldsymbol{s}_{i,j,k} = (u+1, u)$ を $\boldsymbol{s}_{i,j,k} = (u+1, u+1)$ に磁場更新するためには関係する電場の状態が $u+1$，すなわち $\boldsymbol{s}_{i+1,j,k}, \boldsymbol{s}_{i,j+1,k}, \boldsymbol{s}_{i,j,k+1}$ は $(u+1, u)$ または $(u+1, u+1)$ である必要がある．ただし，解析領域の外周の格子点に関しては，適当な境界条件が定義されていて，内部の格子点からの影響のみを考えればよいものとする．

以上を考慮して，提案手法では時刻 t から $t + s_t$ への電磁場更新を以下のように行う．まず，タイルの計算順序として辞書式順序付け（z，y，x 方向の順に優先）を使用し，この順序に従って各タイルの計算を行う．すなわち，(I, J, K) のタイルは $K + (J-1)K_m + (I-1)J_m K_m$ 番目に処理される．ただし J_m，K_m は y，z 方向のタイルの数である．次に，各タイル上で s_t タイムステップ分の電磁場更新を行うが，このとき**図 3** に示されるように，1 タイムステップごとにタイルを x，y，z 方向に -1 格子点分移動させて計算を行う．すなわち，タイル $T_{I,J,K}$ に対して，その位置を 3 方向に $-u$ 格子点分移動させた領域 $T^u_{I,J,K}$ を

$$\begin{aligned} T^u_{I,J,K} = \{(i,j,k) | &\max((I-1)t_x + 1 - u,\ 1) \\ &\le i \le \min(I \cdot t_x - u,\ n_x), \\ &\max((J-1)t_y + 1 - u,\ 1) \\ &\le j \le \min(J \cdot t_y - u,\ n_y), \\ &\max((K-1)t_z + 1 - u,\ 1) \\ &\le k \le \min(K \cdot t_z - u,\ n_z)\}. \end{aligned} \tag{10}$$

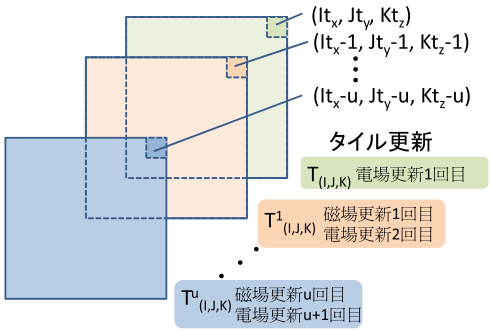


図 3　時間ステップごとのタイルの移動
Fig. 3　Movement of tile in time steps.

のように定義し，各タイル (I, J, K) に関する時刻 $t + u$ の計算を，(u, u) から $(u+1, u)$ への電場更新では領域 $T^u_{I,J,K}$ の更新を，$(u+1, u)$ から $(u+1, u+1)$ への磁場更新では領域 $T^{u+1}_{I,J,K}$ の更新を行うことで，解析領域のすべての格子点について時刻 $t + s_t$ の値を得る．

3.2　提案手法の正当性の検証

本節では，前節で述べた提案手法により各格子点の電場・磁場が正しく更新されることを示す．

z，y，x 方向の順に優先する辞書式順序において (I, J, K) よりも前に位置するタイル座標の集合を

$$\begin{aligned} S_{I,J,K} = \{(L,M,N) | &(L{<}I) \vee (L{=}I \wedge M{<}J) \\ &\vee (L{=}I \wedge M{=}J \wedge N{<}K)\}, \end{aligned} \tag{11}$$

(I, J, K) よりも後に位置するタイル座標の集合を

$$\begin{aligned} N_{I,J,K} = \{(L,M,N) | &(L{>}I) \vee (L{=}I \wedge M{>}J) \\ &\vee (L{=}I \wedge M{=}J \wedge N{>}K)\}, \end{aligned} \tag{12}$$

とし，

$$\begin{aligned} S^u_{I,J,K} = \{(i,j,k) | &(i,j,k) \in T^u_{L,M,N}, \\ &(L,M,N) \in S_{I,J,K}\}, \end{aligned} \tag{13}$$

$$\begin{aligned} N^u_{I,J,K} = \{(i,j,k) | &(i,j,k) \in T^u_{L,M,N}, \\ &(L,M,N) \in N_{I,J,K}\}, \end{aligned} \tag{14}$$

とする．このとき，式 (10) より，

$$\begin{aligned} T^u_{I,J,K} \cap S^u_{I,J,K} &= T^u_{I,J,K} \cap N^u_{I,J,K} \\ &= S^u_{I,J,K} \cap N^u_{I,J,K} = \emptyset, \end{aligned} \tag{15}$$

が成り立つ．

ここで，領域 $T^u_{I,J,K}$ の電場を更新する際に必要な磁場の

© 2013 Information Processing Society of Japan

値を持つ格子点座標の集合 $A^u_{I,J,K}$ とすると，

$$\begin{aligned} A^u_{I,J,K} \subseteq \{(i,j,k) | & \max((I-1)t_x - u,\ 1) \\ & \le i \le \min(I \cdot t_x - u,\ n_x), \\ & \max((J-1)t_y - u,\ 1) \\ & \le j \le \min(J \cdot t_y - u,\ n_y), \\ & \max((K-1)t_z - u,\ 1) \\ & \le k \le \min(K \cdot t_z - u,\ n_z)\}, \end{aligned} \quad (16)$$

となる．解析領域全体の格子点の集合に対する $S^u_{I,J,K}$ の補集合を $\overline{S^u_{I,J,K}}$ とすると，式 (10)，式 (15) より，

$$\begin{aligned} A^u_{I,J,K} &\subseteq \bigcup_{L \ge I, M \ge J, N \ge K} T^{u+1}_{L,M,N} \\ &\subseteq T^{u+1}_{I,J,K} \cup N^{u+1}_{I,J,K} \subseteq \overline{S^{u+1}_{I,J,K}}, \end{aligned} \quad (17)$$

が成り立つ．同様に，

$$\begin{aligned} A^u_{I,J,K} &\subseteq \bigcup_{L \le I, M \le J, N \le K} T^{u}_{L,M,N} \\ &\subseteq T^{u}_{I,J,K} \cup S^{u}_{I,J,K} \end{aligned} \quad (18)$$

となる．ここで，式 (17)，式 (18) より，

$$\begin{aligned} A^u_{I,J,K} &\subseteq (T^u_{I,J,K} \cup S^u_{I,J,K}) - S^{u+1}_{I,J,K} \\ &\subseteq T^u_{I,J,K} \cup (S^u_{I,J,K} - S^{u+1}_{I,J,K}). \end{aligned} \quad (19)$$

また，領域 $T^u_{I,J,K}$ の磁場を更新する際に必要な電場の値を持つ格子点座標の集合 $B^u_{I,J,K}$ とすると，

$$\begin{aligned} B^u_{I,J,K} &\subseteq \bigcup_{L \ge I, M \ge J, N \ge K} T^{u}_{L,M,N} \\ &\subseteq T^{u}_{I,J,K} \cup N^{u}_{I,J,K} \subseteq \overline{S^{u}_{I,J,K}}, \end{aligned} \quad (20)$$

$$\begin{aligned} B^u_{I,J,K} &\subseteq \bigcup_{L \le I, M \le J, N \le K} T^{u-1}_{L,M,N} \\ &\subseteq T^{u-1}_{I,J,K} \cup S^{u-1}_{I,J,K}, \end{aligned} \quad (21)$$

となり，式 (20)，式 (21) より，

$$\begin{aligned} B^u_{I,J,K} &\subseteq (T^{u-1}_{I,J,K} \cup S^{u-1}_{I,J,K}) - S^{u}_{I,J,K} \\ &\subseteq T^{u-1}_{I,J,K} \cup (S^{u-1}_{I,J,K} - S^{u}_{I,J,K}). \end{aligned} \quad (22)$$

ここで，提案手法の実行において，タイル $T_{I,J,K}$ の時刻 $t+v$ から $t+v+1$ への更新開始時の各格子点の更新状況 $\boldsymbol{s}_{i,j,k}$ が，

$$s_{i,j,k} = \begin{cases} (s_t, s_t) & (i,j,k) \in S^{s_t}_{I,J,K} \\ (u+1, u) & (i,j,k) \in S^u_{I,J,K} - S^{u+1}_{I,J,K}, \\ & v \le u \le s_t - 1 \\ (v, v) & (i,j,k) \in T^v_{I,J,K} \\ (u+1, u) & (i,j,k) \in (S^u_{I,J,K} \cup T^u_{I,J,K}) \\ & -(S^{u+1}_{I,J,K} \cup T^{u+1}_{I,J,K}), \\ & 0 \le u \le v-1 \\ (0,0) & (i,j,k) \notin S^0_{I,J,K} \cup T^0_{I,J,K} \end{cases}, \quad (23)$$

であると仮定する．

タイル $T_{1,1,1}$ の計算の開始時刻 t では，$S_{1,1,1} = \emptyset$ より式 (23) は，

$$\boldsymbol{s}_{i,j,k} = (0,0) \quad (\forall (i,j,k)), \quad (24)$$

となり，明らかに式 (23) が成り立つ．

タイル $T_{I,J,K}$ の計算中に時刻 $t+v$ において式 (23) が正しいとすると，時刻 $t+v+1$ への更新は以下のように行われる．まず，電場更新が領域 $T^v_{I,J,K}$ で行われる．このとき，式 (19) および式 (23) より，

$$\boldsymbol{s}_{i,j,k} = (v+1, v) \quad ((i,j,k) \in A^v_{I,J,K} \cap S^v_{I,J,K}), \quad (25)$$

$$\boldsymbol{s}_{i,j,k} = (v, v) \quad ((i,j,k) \in T^v_{I,J,K}), \quad (26)$$

となり，領域 $T^v_{I,J,K}$ の電場は更新可能である．更新により $\boldsymbol{s}_{i,j,k} = (v+1, v)$ $((i,j,k) \in T^v_{I,J,K})$ となる．続いて領域 $T^{v+1}_{I,J,K}$ の磁場が更新される．このとき式 (22) および電場更新の結果より，

$$\boldsymbol{s}_{i,j,k} = (v+1, v) \quad ((i,j,k) \in B^{v+1}_{I,J,K}), \quad (27)$$

となり，領域 $T^{v+1}_{I,J,K}$ の磁場は更新可能である．更新により $\boldsymbol{s}_{i,j,k} = (v+1, v+1)$ $((i,j,k) \in T^{v+1}_{I,J,K})$ となる．

以上を式 (23) に適用すると，タイル $T_{I,J,K}$ の計算中の時刻 $t+v+1$ のとき各格子点の更新状況 $\boldsymbol{s}_{i,j,k}$ は，

$$\boldsymbol{s}_{i,j,k} = \begin{cases} (s_t, s_t) & (i,j,k) \in S^{s_t}_{I,J,K} \\ (u+1, u) & (i,j,k) \in S^u_{I,J,K} - S^{u+1}_{I,J,K}, \\ & v+1 \le u \le s_t - 1 \\ (v+1, v+1) & (i,j,k) \in T^{v+1}_{I,J,K} \\ (u+1, u) & (i,j,k) \in (S^u_{I,J,K} \cup T^u_{I,J,K}) \\ & -(S^{u+1}_{I,J,K} \cup T^{u+1}_{I,J,K}), \\ & 0 \le u \le v \\ (0,0) & (i,j,k) \notin S^0_{I,J,K} \cup T^0_{I,J,K} \end{cases}, \quad (28)$$

となり，時刻 $t+v+1$ においても式 (23) の仮定が成り立つ．

タイル $T_{I,J,K}$ の計算の終了時，時刻 $t+s_t$ において式 (23) は，

$$\boldsymbol{s}_{i,j,k} = \begin{cases} (s_t, s_t) & (i,j,k) \in S^{s_t}_{I,J,K} \cup T^{s_t}_{I,J,K} \\ (u+1, u) & (i,j,k) \in (S^u_{I,J,K} \cup T^u_{I,J,K}) \\ & -(S^{u+1}_{I,J,K} \cup T^{u+1}_{I,J,K}), \\ & 0 \le u \le s_t - 1 \\ (0,0) & (i,j,k) \notin S^0_{I,J,K} \cup T^0_{I,J,K} \end{cases}, \quad (29)$$

となる．ここで，辞書式順序において (I,J,K) の次に更新するタイル座標を (I',J',K') とすると，$S^u_{I,J,K} \cup T^u_{I,J,K} = S^u_{I',J',K'}, u = \{0, \cdots s_t\}$ であり，これを用いると式 (29) は，

$$\boldsymbol{s}_{i,j,k} = \begin{cases} (s_t, s_t) & (i,j,k) \in S^{s_t}_{I',J',K'} \\ (u+1, u) & (i,j,k) \in S^u_{I',J',K'} - S^{u+1}_{I',J',K'}, \\ & 0 \le u \le s_t - 1 \\ (0,0) & (i,j,k) \notin S^0_{I',J',K'} \end{cases}, \quad (30)$$

© 2013 Information Processing Society of Japan

と書ける．これはタイル $T_{I',J',K'}$ の計算の開始時，時刻 $t+0$ の式 (23) と等しい．よって，帰納的に式 (23) はすべてのタイル，$v=\{0,\cdots s_t\}$ で成り立つ．

式 (23) を用いてすべてのタイルを計算し終えた時点で，

$$\boldsymbol{s}_{i,j,k}=(s_t,s_t)\quad(\forall(i,j,k)), \tag{31}$$

となり，解析領域全体が時刻 $t+s_t$ まで正しく更新される．

以上より，提案手法は式 (23) で表される更新状況に従い正しく動作することが示された，

3.3　提案手法の電磁場更新手順

前節での議論をふまえ，提案手法では以下のような手順により解析領域内の電磁場を更新する．なお，以下の記述において可読性のために「解析領域」あるいは「領域」という表現を使用するが，これはプログラム上ではいずれも電場・磁場に対応する配列である．

Step 1.　解析領域に初期値を入力する．

Step 2.　$I_m=\lceil(n_x+s_t)/t_x\rceil, J_m=\lceil(n_y+s_t)/t_y\rceil, K_m=\lceil(n_z+s_t)/t_z\rceil$ とし，タイル $T_{1,1,1},\cdots,T_{I_m,J_m,K_m}$ を定義する．

Step 3.　解析領域に対して式 (10) に従って領域分割を行い，$T^u_{I,J,K}, u\in\{0,\cdots s_t\}$ を定義する．

Step 4.　$(I,J,K)=(1,1,1)$ とする．

Step 5.　タイル $T_{I,J,K}$ に対して s_t タイムステップの電磁場更新を行う．ここで，タイル $T_{I,J,K}$ の更新において v 回目の電場更新は領域 $T^{v-1}_{I,J,K}$ を，v 回目の磁場更新は領域 $T^v_{I,J,K}$ を対象とする．

Step 6.　$(I,J,K)=(I_m,J_m,K_m)$ ならば **Step 7.** に進む．それ以外の場合は，3.1 節で示した辞書式順序に従ってタイル座標 (I,J,K) を更新し，**Step 5.** に戻る．

Step 7.　時刻を s_t 進める．計算を続ける場合，**Step 4.** に戻る．

ここで，**Step 2.** における I_m，J_m，K_m の設定に関して補足する．提案手法において，1 回のタイリングは解析領域すべてを s_t ステップ分更新する作業と等価となる必要がある．したがって，$T^0_{I,J,K}$ から $T^{s_t}_{I,J,K}$ のいずれかが解析領域を含むようなタイル (I,J,K) については，そのすべてが定義されている必要がある．あるタイル $T_{I,J,K}$ の各ステップでの更新範囲は，ステップが進むごとに x，y，z 方向に -1 格子点分移動していく形となるため，$T^0_{I,J,K}$ から $T^{s_t}_{I,J,K}$ は s_t 格子点分だけずれることとなり，計算に必要なすべてのタイルを定義するためにはタイル生成時に解析領域外に s_t 格子点分だけ広げて I_m，J_m，K_m を設定する必要がある．

3.4　提案手法により期待できる効果

本節では，解析格子がキャッシュ容量に比べて十分に大きい場合について考えるものとする．3 次元 FDTD 法において，提案手法と Naive な実装法を比較した場合，前者の性能改善の要因はキャッシュデータを有効活用し，メモリからのデータのロード/ストア量を軽減することにある．したがって，キャッシュ容量内のデータに対して 3 次元 FDTD カネールを実行した場合の 1 格子点・タイムステップあたりの計算時間を τ_c とした場合，提案手法における同計算時間の下限は τ_c で与えられ，これが提案手法により得られる性能改善の限界値となる．

次に，上記の性能限界値に対して，提案手法におけるオーバヘッドについて考える．提案手法において，あるタイルの計算を行う際にメインメモリからのデータロードが必要になるのは，タイル上の最初のステップの計算に必要な格子点データと，各タイムステップでタイル位置が変化した際に新たに必要となる領域のデータである．ここで，タイル内での 1 回目のタイムステップの影響は s_t が増加するに従って相対的に減少する．したがって，s_t をなるべく大きくすることが本観点から有利となるが，過度に大きな s_t の設定は，$T^{s_t}_{I,J,K}$ の計算が終了した後に $T^0_{I,J,K+1}$ の計算を行う際において，$T^0_{I,J,K+1}$ と $T^0_{I,J,K}$ の境界面の格子点に関するデータがキャッシュ上に存在しない状況を招き，結果的にキャッシュデータの再利用性を低下させるため，それを避ける必要がある．また，各タイムステップごとに新たにデータロードを行う格子点数はタイルのサイズ $t_x\times t_y\times t_z$ に対して $t_x\times t_y+t_y\times t_z+t_z\times t_x$ で与えられる．したがって，たとえば $n_t=t_x=t_y=t_z$ の立方体タイルを考えた場合，両者の比は $n_t/3$ で与えられ，タイルサイズがなるべく大きい方がこれらのデータロードにともなうオーバヘッドの影響は低減される．そこで，タイルサイズはキャッシュ容量以下でなるべく大きくする方が有利になると考えられる．ただし，実際の解析では，キャッシュの構成（ウェイ数等）の影響により，キャッシュ容量のすべてをタイル内の格子点データで使用することは困難なため，キャッシュ容量の数十%程度を目途にその範囲内でなるべく大きなタイルを用いるのが有効であると考えられる．

4.　性能評価

4.1　数値実験環境

提案手法の有効性の評価のため，数値実験を行った．実験に使用した計算機は，京都大学学術情報メディアセンターの T2K オープンスーパーコンピュータの計算ノードである富士通 HX600 [5] である．HX600 は 4 個の AMD 社製クアッドコア Opteron (8356) プロセッサと 32 GB（DDR2-667）のメモリを有している．本プロセッサの性能は，理論ピーク演算性能が 36.8 GFlops，コアあたり 9.2 GFlops，キャッシュメモリ容量は L2 キャッシュがコアあたり 512 KB，L3 キャッシュがプロセッサあたり 2 MB となっている．また，メインメモリの帯域はプロセッサ（ソケット）あたり

© 2013 Information Processing Society of Japan

10.6 GB/s である．

本数値実験では，金属壁に囲まれた立方体形状の解析領域を使用する．金属壁内では電場 E と磁場 H はともに 0 となるとし，演算は行わない．このため，解析領域中の 1 方向の格子点数を n と表した場合，金属壁を含む領域全体は $(n+2)\times(n+2)\times(n+2)$ の格子に分割され，このうち計算の対象となる格子は $n\times n\times n$ となる．実応用上の 3 次元 FDTD 解析では，解析の精度を高めるために解析に用いるデータサイズ（格子点数）はキャッシュ容量と比べて十分に大きな値をとることが一般的であり，たとえば文献 [6] では，4,000 万格子点を超える解析が報告されている．こうした背景や実験環境のメモリ容量を考慮し，本解析では 1 方向の格子点数 n は 250 とする．また，本数値実験ではマルチコアプロセッサによるスレッド並列処理を利用する．HX600 の 1 ノード上の 1 つのプロセッサを用い，最大 4 スレッドによる並列実行を行う．本実験の実装では，最外ループを x 方向，最内ループを z 方向とし，最内ループである z 方向においてメモリアクセスが連続となるものとする．スレッド並列処理を行う場合，本実験ではタイルを最外ループである x 方向にスレッド数だけ分割し，各スレッドが分割されたタイル内の格子点を担当する．なお，本実験では各スレッドを 1 つのプロセッサ（ソケット）上に集中して配置し，解析プログラム中の各種配列は，使用するプロセッサ（ソケット）に接続されたメモリ上に配置するものとする．

数値実験により提案手法における 1 格子点・タイムステップあたりの計算時間を求め，Naive な実装および文献 [3] の実装と比較する．なお，本研究ではタイル形状として立方体タイルを用いるものとする．タイルサイズ n_t を 5〜50 の範囲，タイムステップ s_t を 1，5，10，20，25 と変化させ，提案手法の性能を評価する．

4.2　数値実験結果

図 4，**図 5**，**図 6** に，逐次実行，2 および 4 スレッド実行時における測定結果を示す．横軸がタイルのサイズ n_t，縦軸が 1 格子点・タイムステップあたりの計算時間 τ_t を示している．比較のため，同じ条件で実行した Naive な実装，文献 [3] による実装および 3.4 節で示した性能改善の限界値の計算時間も図中に示す．図中において “Old” は文献 [3] で述べられた既存手法の結果を表し，“Limit” は各スレッド数に対して Naive な実装法を使用し，キャッシュ容量に比べて十分に小さい範囲内で領域サイズを変化させて得られた τ_t の最小値で，実効的な τ_c の値である．

まず，既存手法である文献 [3] で述べた実装法との比較について述べる．図 4，5，6 より，逐次，2，4 スレッド実行時のいずれの場合においても，提案手法が既存手法を上回る性能を有していることが分かる．既存手法では，スレッドあたりの実効メモリバンド幅が比較的大きい逐次，

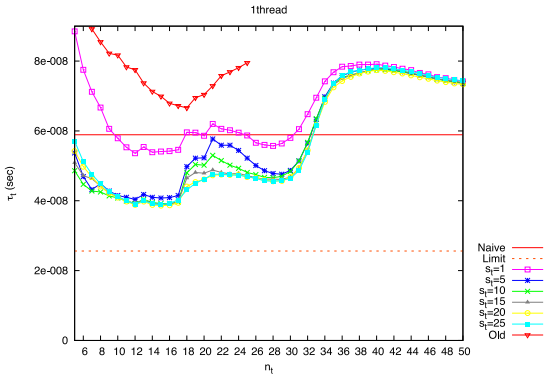


図 4　逐次実行時における 1 格子点・タイムステップあたりの計算時間

Fig. 4　Computational time per grid point in one time step in serial computing.

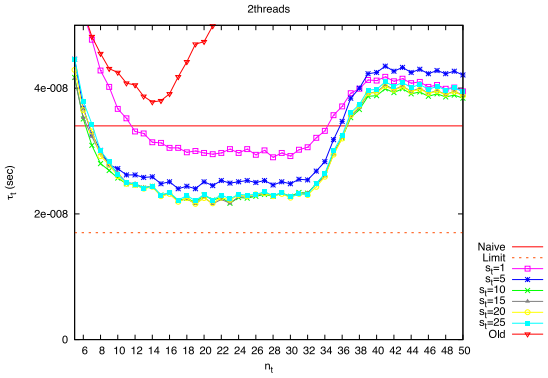


図 5　2 スレッド並列実行時における 1 格子点・タイムステップあたりの計算時間

Fig. 5　Computational time per grid point in one time step in parallel computing with 2 threads.

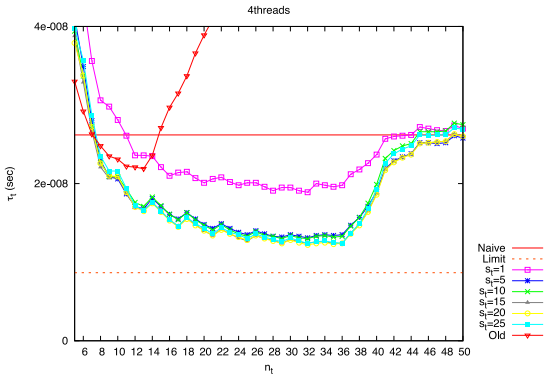


図 6　4 スレッド並列実行時における 1 格子点・タイムステップあたりの計算時間

Fig. 6　Computational time per grid point in one time step in parallel computing with 4 threads.

2 スレッド実行時には Naive な実装法を上回る性能を得ることができなかったが，提案手法では適切なタイルサイズとステップ数 s_t を用いることで Naive な実装による性能を改善することが可能である．本数値実験条件下で 4 スレッドを用いた実験において，提案手法は n_t=32，s_t=20 のときに最良の結果を得ており，このとき Naive な実装と

© 2013 Information Processing Society of Japan

比べて 53%，既存手法における最良の結果と比べて 44%の計算時間の短縮を実現している．本性能改善の主たる要因は提案手法で導入した時空間タイリングにおいて，格子点に関する冗長計算が必要とされないためと考えられる．その結果，逐次，2 スレッド実行時においても，Naive な実装と比べてそれぞれ 34%（n_t=15，s_t=20），36%（n_t=20，s_t=19）の計算時間の短縮が実現された．

次に，提案手法におけるタイルサイズ n_t およびステップ数 s_t の影響について考える．図 4，5，6 によると，逐次実行時で s_t の値が比較的小さい場合を除いて，1 格子点・タイムステップあたりの計算時間 t_t はタイルサイズに対して下に凸のグラフを描く挙動を示している．本挙動は以下のように説明される．すなわち，タイルサイズが小さすぎる場合には，タイリングを行うために必要なタイル情報の取得や，小さなタイルでタイリングを行うことでデータアクセスの連続性が下がることによるによるオーバヘッドが存在するために良好な性能が得られない．一方，タイルサイズがキャッシュメモリの容量を超過した場合にはメモリアクセスが発生し，著しく性能が劣化する．これらの挙動は文献 [3] における既存手法においても見られるが，同手法の場合には同一のタイル内時間ステップ数に対してタイルサイズを増加させた場合，冗長計算される領域が拡大するオーバヘッドが存在し，図中のグラフの概形は V 字型となる．一方，冗長計算を必要としない提案手法では，特に並列実行時で十分に大きな s_t を用いた場合，比較的良好な性能が得られるタイルサイズの範囲が大きい．このことから，大域的最適化手法によりタイルサイズ，形状を最適化する場合，提案手法は既存手法と比べて少ない試行回数で性能の良いタイルを定めることができると考えられる．

次に，ステップ数 s_t については，逐次実行時，並列実行時のいずれの場合においてもその値を増大させることにより性能が改善される．しかし，s_t が 10 以上の場合，s_t の性能に与える影響は小さくなる．3.4 節で述べたように，s_t を増加させることによる性能改善は，あるタイル上の処理について最初に必要となるメモリアクセスの影響が次第に減少することによると考えられる．したがって，s_t を増加させるにつれてその効果は減少することとなり，本数値実験の場合では，s_t を 10 以上とすれば上記のメモリアクセスの影響は十分に軽減されているといえる．

次に，**図 7** に実効メモリバンド幅を計測するベンチマークである Stream benchmark（Triad），Naive な実装法，提案手法の各々について逐次実行時の計算時間を基準とした台数効果を示す．図 7 より，Naive な実装法の場合，並列台数効果はほぼ実効メモリバンド幅の台数効果と同様の振舞いを示すことが分かる．一方，時空間タイリングを用いた提案手法では，キャッシュメモリを有効に活用することにより，2 から 4 へのスレッド数の増加に対して実効メモリバンド幅の拡大率以上の性能改善を実現している．現在，

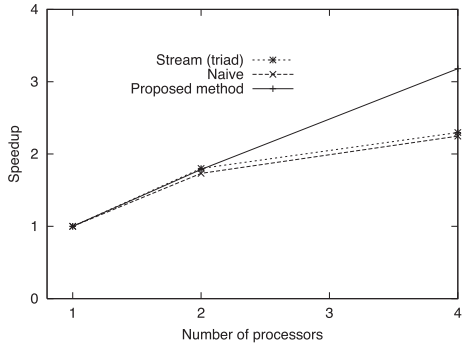


図 7　並列台数効果（実効メモリバンド幅，Naive な実装，提案手法）
Fig. 7　Parallel speedup (effective memory bandwidth, naive implementation, proposed method).

マルチコアプロセッサに関する技術動向では，コアあたりのメモリバンド幅は低下する傾向にあるため，メモリ帯域の影響を受けやすい 3 次元 FDTD 法では時空間タイリングによる性能改善は今後も重要性を持つと考えられる．

5. 関連研究

1 章で述べたように，本研究で対象とする 3 次元 FDTD 法は反復型ステンシル計算の 1 つと考えることができる．反復型ステンシル計算は一般に演算量と比べてデータのロード/ストア量が多いため，その計算速度は使用計算機の実効メモリ帯域に大きく影響される．そこで，著者らの先行論文 [3] で言及したように，ステンシル計算の対象となる解析領域をキャッシュサイズを考慮して小領域に分割し，各小領域で複数の（時間に関する）反復を行う時空間タイリング手法による研究 [2], [7] が行われている．しかし文献 [3] において，FDTD 法への時空間タイリングの適用に関しては，差分解析において定常反復法を利用した場合に見られるような一般的な反復型ステンシル計算の形式，すなわち時間ループ内に空間に関するループが 1 つしかない形式によって記述可能な 2 次元 FDTD 法に関する報告は見られるが，時間ループ中に 2 つの空間に関するループを有し，またその各々でステンシルの形状が異なる 3 次元 FDTD 法に関する報告例がほとんど見当たらないことを述べた．

本章では文献 [3] 以降の（3 次元）FDTD 法の高速化に関連する研究について述べる．実応用上において 3 次元 FDTD 法は重要な計算手法であり，たとえば文献 [8] では実用的な応用モデルを用いた解析例を示しつつ，その高速化について論じている．このような 3 次元 FDTD 法の高速化に関する研究では，同手法が陽的な解法で SIMD 計算への親和性が高いことから GPU に関連した研究が多く報告されている．たとえば，先の文献 [8] のほかに，文献 [9]，[10] 等があげられる．一方，時空間タイリングによる FDTD 法の高速化については，文献 [11]，[12] の報告があり，文献 [11] では冗長計算をともなわない手法について

© 2013 Information Processing Society of Japan

も述べられているが，いずれの文献も 1 次元もしくは 2 次元 FDTD 法を対象とした記述にとどまっており，3 次元 FDTD 法において冗長計算をともなわない時空間タイリングが可能であることを示した本論文とは内容が異なっている．

6. まとめ

本論文では，電磁場解析の一手法である 3 次元 FDTD（Finite Difference Time Domain）法を対象とし，その計算性能の改善に関する検討を行い，キャッシュメモリをより有効に活用し計算性能を向上させる方法について述べた．著者らは先行研究において，3 次元 FDTD 法において複数のタイムステップの計算をタイル（小領域）上で行いキャッシュメモリのヒット率を向上させる時空間タイリング手法 [3] を提案した．しかしながら，各タイムステップで固定のタイルを用いる同手法では，隣接するタイル間の影響を考慮するために冗長な計算を必要とした．そこで本論文ではタイルの位置を時間ステップごとに変化させ，タイルの計算順序を適切に設定することで冗長な計算を排除したより高性能な時空間タイリング手法を提案した．マルチコアプロセッサ上の数値実験の結果，提案手法においてタイムステップ数やタイルサイズを適切に設定することで，逐次，並列実行時のいずれの場合においても，Naive な実装法および先行研究での手法 [3] を上回る性能を得た．4 スレッド並列実行の場合，提案手法は Naive な実装と比べて約 53%の計算時間の短縮を実現した．

なお，本研究では単一プロセッサ上で性能評価を行ったが，計算環境として複数のプロセッサによる共有メモリ型並列計算環境や SMP クラスタ型の計算機も広く利用されている．それらの計算環境に向けては，マルチプロセス，マルチスレッドを併用したハイブリッド並列処理が有効であると考えられる．たとえば，プロセス並列化のために問題領域を適宜分割したうえで，個々の部分領域を担当するプロセスでは提案手法に基づくスレッド並列化を施し，隣接した部分領域間で厚さ s_t 分の境界領域の格子点データを s_t ステップごとに交換することで，ハイブリッド並列化を実現することができる．この方法は単純な領域分割による並列化に比べ，タイリングによる高速化に加えてプロセス間通信回数を削減する効果がある一方，厚さ s_t の境界領域について冗長な計算を必要とするため，最適な s_t を定めるトレードオフ要因が加わることになる．また境界通信コストの隠蔽を図る場合，境界領域の計算をそれ以外の領域の計算に先行して実施するのが通例であるが，提案手法ではタイルの計算順序を辞書式とする必要があるため，この隠蔽手法を単純に適用することはできず，新たな手法を見い出す必要がある．そのためハイブリッド並列による 3 次元 FDTD 法の実装と性能評価に関しては今後の課題としたい．

参考文献

[1] Yee, K.S.: Numerical solution of initial boundary value problems involving Maxwell's equations in isotropic media, *IEEE Trans. Antennas Propagat.*, Vol.AP-14, pp.302–307 (Aug. 1966).

[2] Wolf, M.: More iteration space tiling, *Proc. Supercomputing '89*, pp.655–664 (1989).

[3] 南　武志，高橋康人，岩下武史，中島　浩：キャッシュメモリを考慮した 3 次元 FDTD カーネルの性能改善，情報処理学会論文誌 コンピューティングシステム，Vol.4, No.2, pp.70–83 (2011).

[4] 宇野　亨：FDTD 法による電磁界およびアンテナ解析，コロナ社，東京 (1998).

[5] Nakashima, H.: T2K Open Supercomputer: Inter University and Inter-Disciplinary: Collaboration on the New Generation Supercomputer, *Proc. Intl. Conf. Informatics Education and Research for Knowledge-Circulating Society*, pp.137–142 (2008).

[6] 小林敬生，佐藤源之：3 次元 FDTD 法による地雷探査用地中レーダの解析，電子情報通信学会技術研究報告，A·P，アンテナ・伝播，103 (265), pp.7–11 (2003).

[7] Strzodka, R., Shaheen, M., Pajak, D. and Seidel, H.P.: Cache oblivious parallelograms in iterative stencil computations, *ICS '10: Proc. 24th ACM international conference on supercomputing*, pp.49–59, ACM (2010).

[8] Yu, W., Yang, X., Liu, Y., Mittra, R., Chang, D.C., Liao, C.H., Muto, A., Li, W. and Zhao, L.: New Development of Parallel Conformal FDTD Method in Computational Electromagnetics Engineering, *IEEE Antennas and Propagation Magazine*, Vol.53, No.3, pp.15–41 (2011).

[9] Kim, K., Kim, K. and Park, Q.: Performance analysis and optimization of three-dimensional FDTD on GPU using roofline model, *Computer Physics Communications*, Vol.182, pp.1201–1207 (2011).

[10] Chi, J., Liu, F., Weber, E., Li, Y. and Crozier, S.: GPU-Accelerated FDTD Modeling of Radio-Frequency Field. Tissue Interactions in High-Field MRI, *IEEE Trans. Biomedical Engineering*, Vol.58, No.6, pp.1789–1976 (2011).

[11] Orozco, D., Garcia, E. and Gao, G.: Locality Optimization of Stencil Applications Using Data Dependency Graphs, *Languages and Compilers for Parallel Computing*, Lecture Notes in Computer Science, Vol.6548, pp.77–91 (2011).

[12] Tavarageri, S., Pouchet, L.N., Ramanujam, J., Rountev, A. and Sadayappan, P.: Dynamic Selection of Tile Sizes, *Proc. 18th Intl. Conf. High Performance Computing* (*HiPC*), pp.1–10 (2011).

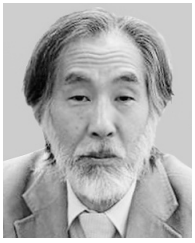
南 武志　（学生会員）

平成 22 年 4 月より京都大学大学院情報学研究科博士後期課程在籍．高性能計算，電磁界解析に関する研究に従事．

© 2013 Information Processing Society of Japan

岩下 武史　（正会員）

平成 10 年京都大学大学院工学研究科電気工学専攻博士課程修了．博士（工学）．平成 10 年京都大学大学院工学研究科リサーチ・アソシエイト（日本学術振興会未来開拓学術研究推進事業 PD），平成 12 年同大学大型計算機センター助手を経て，平成 15 年より同大学学術情報メディアセンター助教授（平成 19 年職名変更により同准教授），現在に至る．高性能計算，線形反復法，電磁界解析，並列処理に関する研究に従事．IEEE，SIAM，日本応用数理学会，電気学会，日本 AEM 学会，日本計算工学会各会員．



中島 浩　（正会員）

昭和 31 年生．昭和 56 年京都大学大学院工学研究科情報工学専攻修士課程修了．同年三菱電機（株）入社．推論マシンの研究開発に従事．平成 4 年より京都大学工学部助教授．平成 9 年より豊橋技術科学大学教授．平成 18 年より京都大学教授．並列計算機のアーキテクチャ，プログラミング言語の実装方式に関する研究に従事．工学博士．昭和 63 年元岡賞，平成 5 年坂井記念特別賞受賞．IEEE-CS，ACM，ALP，TUG 各会員．

© 2013 Information Processing Society of Japan